OLLSCOIL NA GAILLIMHE
UNIVERSITY OF GALWAY

Provided by the author(s) and University of Galway in accordance with publisher policies. Please cite the published version when available.

Title	Neural transfer learning for natural language processing
Author(s)	Ruder, Sebastian
Publication Date	2019-06-07
Publisher	NUI Galway
Item record	http://hdl.handle.net/10379/15463

Downloaded 2024-04-20T06:25:23Z

Some rights reserved. For more information, please see the item record link above.

NATIONAL UNIVERSITY OF IRELAND, GALWAY

Neural Transfer Learning for Natural Language Processing

by

Sebastian Ruder

A thesis submitted in partial fulfillment for the
degree of Doctor of Philosophy

in the
College of Engineering and Informatics
School of Engineering and Informatics

Supervisors: John G. Breslin, Parsa Ghaffari

February, 2019

Contents

Declaration of Authorship vi

Abstract viii

Acknowledgements ix

List of Figures x

List of Tables xii

Abbreviations xiv

Notation xvi

1 Introduction 1
- 1.1 Motivation . . . 1
- 1.2 Research objectives . . . 3
- 1.3 Contributions . . . 4
- 1.4 Thesis outline . . . 6
- 1.5 Publications . . . 7

2 Background 10
- 2.1 Probability and information theory . . . 10
  - 2.1.1 Probability basics . . . 10
  - 2.1.2 Distributions . . . 15
  - 2.1.3 Information theory . . . 16
- 2.2 Machine learning . . . 18
  - 2.2.1 Maximum likelihood estimation . . . 19
  - 2.2.2 Linear regression . . . 21
  - 2.2.3 Gradient descent . . . 24
  - 2.2.4 Generalization . . . 25
  - 2.2.5 Regularization . . . 28
- 2.3 Neural networks . . . 30
  - 2.3.1 Layers and models . . . 32
  - 2.3.2 Back-propagation . . . 35

2.4 Natural language processing . . . 37
2.5 Conclusions . . . 41

3 Transfer Learning 42
3.1 Introduction . . . 42
3.1.1 Definition . . . 44
3.1.2 Scenarios . . . 44
3.1.3 Taxonomy . . . 45
3.2 Multi-task learning . . . 47
3.2.1 Introduction . . . 47
3.2.2 Motivation . . . 47
3.2.3 Two methods for MTL in neural networks . . . 48
3.2.4 Why does MTL work? . . . 49
3.2.5 MTL in non-neural models . . . 50
3.2.5.1 Block-sparse regularization . . . 51
3.2.5.2 Learning task relationships . . . 52
3.2.6 Auxiliary tasks . . . 55
3.2.6.1 Common types of auxiliary tasks . . . 57
3.2.6.2 Related tasks used in NLP . . . 60
3.2.7 Summary . . . 62
3.3 Sequential transfer learning . . . 63
3.3.1 Introduction . . . 63
Sequential transfer learning stages . . . 64
3.3.2 Pretraining . . . 65
3.3.2.1 Distantly supervised pretraining . . . 66
3.3.2.2 Supervised pretraining . . . 67
3.3.2.3 Unsupervised pretraining . . . 67
3.3.2.4 Multi-task pretraining . . . 76
3.3.2.5 Architectures . . . 76
3.3.3 Adaptation . . . 77
3.3.3.1 Fine-tuning settings . . . 78
3.3.3.2 A framework for adaptation . . . 79
3.3.4 Lifelong learning . . . 80
3.3.5 Evaluation scenarios . . . 85
3.3.6 Summary . . . 85
3.4 Domain adaptation . . . 86
3.4.1 Introduction . . . 86
3.4.2 Representation approaches . . . 87
3.4.2.1 Distribution similarity approaches . . . 87
3.4.2.2 Latent feature learning . . . 90
3.4.3 Weighting and selecting data . . . 93
3.4.3.1 Instance weighting . . . 94
3.4.3.2 Instance selection . . . 95
3.4.4 Self-labelling approaches . . . 96
3.4.4.1 Self-training . . . 97
3.4.4.2 Multi-view training . . . 97
3.4.5 Multi-source domain adaptation . . . 99

3.4.6 Summary . . . 100
3.5 Cross-lingual learning . . . 100
3.5.1 Introduction . . . 101
3.5.2 Notation and terminology . . . 102
3.5.3 A typology for cross-lingual word embedding models . . . 104
3.5.4 Word-level alignment models . . . 106
3.5.4.1 Word alignment methods with parallel data . . . 106
3.5.4.2 Word alignment methods with comparable data . . . 115
3.5.5 Sentence-level alignment models . . . 116
3.5.5.1 Sentence alignment methods with parallel data . . . 117
3.5.5.2 Sentence alignment methods with comparable data . . . 123
3.5.6 Document-level alignment models . . . 123
3.5.6.1 Document alignment methods with comparable data . . . 124
3.5.7 Evaluation . . . 126
3.5.8 Summary . . . 128
3.6 Conclusions . . . 129

4 Selecting Data for Domain Adaptation 130
4.1 Learning to Select Data for Transfer Learning With Bayesian Optimization 131
4.1.1 Introduction . . . 131
4.1.2 Data selection model . . . 132
4.1.2.1 Bayesian Optimization for data selection . . . 133
4.1.2.2 Features . . . 133
4.1.3 Experiments . . . 135
4.1.4 Results . . . 137
4.1.5 Related work . . . 143
4.1.6 Summary . . . 144
4.2 Strong Baselines for Neural Semi-supervised Learning under Domain Shift 145
4.2.1 Introduction . . . 145
4.2.2 Neural bootstrapping methods . . . 146
4.2.2.1 Self-training . . . 147
4.2.2.2 Tri-training . . . 148
4.2.2.3 Multi-task tri-training . . . 149
4.2.3 Experiments . . . 152
4.2.3.1 POS tagging . . . 152
4.2.3.2 Sentiment analysis . . . 153
4.2.3.3 Baselines . . . 153
4.2.3.4 Results . . . 153
4.2.4 Related work . . . 158
4.2.5 Summary . . . 159
4.3 Conclusions . . . 159

5 Unsupervised and Weakly Supervised Cross-lingual Learning 160
5.1 The Limitations of Unsupervised Bilingual Dictionary Induction . . . 161
5.1.1 Introduction . . . 161
5.1.2 How similar are embeddings across languages? . . . 162
5.1.3 Unsupervised cross-lingual learning . . . 164

5.1.3.1 Learning scenarios . . . 164
5.1.3.2 Summary of Conneau et al. [2018a] . . . 165
5.1.3.3 A simple supervised method . . . 166
5.1.4 Experiments . . . 167
5.1.4.1 Experimental setup . . . 167
5.1.4.2 Impact of language similarity . . . 167
5.1.4.3 Impact of domain differences . . . 169
5.1.4.4 Impact of hyper-parameters . . . 171
5.1.4.5 Impact of dimensionality . . . 172
5.1.4.6 Impact of evaluation procedure . . . 173
5.1.4.7 Evaluating eigenvector similarity . . . 174
5.1.5 Related work . . . 174
5.1.6 Summary . . . 175
5.2 A Discriminative Latent-Variable Model for Bilingual Lexicon Induction . 176
5.2.1 Introduction . . . 176
5.2.1.1 Graph-theoretic formulation . . . 177
5.2.1.2 Word embeddings . . . 178
5.2.2 A latent-variable model . . . 178
5.2.3 Parameter estimation . . . 181
5.2.3.1 Viterbi E-Step . . . 181
5.2.3.2 M-Step . . . 182
5.2.4 Reinterpretation of Artetxe et al. [2017] as a latent-variable model 183
5.2.5 Experiments . . . 184
5.2.5.1 Experimental details . . . 185
5.2.5.2 Results . . . 186
5.2.6 Analysis . . . 187
5.2.7 Related work . . . 190
5.2.8 Summary . . . 191
5.3 Conclusions . . . 192

6 Improved Sharing in Multi-task Learning 193
6.1 Latent Multi-task Architecture Learning . . . 194
6.1.1 Introduction . . . 194
6.1.2 Multi-task architecture learning . . . 196
6.1.3 Prior work as instances of sluice networks . . . 199
6.1.4 Experiments . . . 201
6.1.5 Analysis . . . 205
6.1.6 Related work . . . 206
6.1.7 Summary . . . 208
6.2 Multi-task Learning of Pairwise Sequence Classification Tasks Over Disparate Label Spaces . . . 208
6.2.1 Introduction . . . 209
6.2.2 Related work . . . 210
6.2.3 Multi-task learning with disparate label spaces . . . 211
6.2.4 Experiments . . . 214
6.2.5 Analysis . . . 218
6.2.6 Summary . . . 222

6.3 Conclusions . . . 222

7 Adapting Universal Pretrained Representations 223
7.1 Universal Language Model Fine-tuning for Text Classification . . . 224
7.1.1 Introduction . . . 224
7.1.2 Related work . . . 225
7.1.3 Universal Language Model Fine-tuning . . . 226
7.1.3.1 General-domain LM pretraining . . . 228
7.1.3.2 Target task LM fine-tuning . . . 228
7.1.3.3 Target task classifier fine-tuning . . . 230
7.1.4 Experiments . . . 232
7.1.4.1 Experimental setup . . . 232
7.1.4.2 Results . . . 233
7.1.5 Analysis . . . 235
7.1.6 Summary . . . 239
7.2 To Tune or Not to Tune? . . . 239
7.2.1 Introduction . . . 239
7.2.2 Pretraining and adaptation . . . 240
7.2.3 Experimental setup . . . 241
7.2.3.1 Target tasks and datasets . . . 241
7.2.3.2 Feature extraction . . . 242
7.2.3.3 Fine-tuning . . . 243
7.2.4 Results . . . 244
7.2.5 Analyses . . . 245
7.2.6 Summary . . . 249
7.3 Conclusions . . . 249

8 Conclusion 250
8.1 Synopsis . . . 250
8.2 Summary of findings . . . 251
8.3 Future directions . . . 255

Bibliography 260

Declaration of Authorship

I, Sebastian Ruder, declare that this thesis titled, 'Neural Transfer Learning for Natural Language Processing' and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed: _____

Date: _____

"Science is not about building a body of known "facts". It is a method for asking awkward questions and subjecting them to a reality-check, thus avoiding the human tendency to believe whatever makes us feel good."

- Terry Pratchett, *The Science of Discworld*

NATIONAL UNIVERSITY OF IRELAND, GALWAY

Abstract

College of Engineering and Informatics
School of Engineering and Informatics

Doctor of Philosophy

by Sebastian Ruder

The current generation of neural network-based natural language processing models excels at learning from large amounts of labelled data. Given these capabilities, natural language processing is increasingly applied to new tasks, new domains, and new languages. Current models, however, are sensitive to noise and adversarial examples and prone to overfitting. This brittleness, together with the cost of attention, challenges the supervised learning paradigm.

Transfer learning allows us to leverage knowledge acquired from related data in order to improve performance on a target task. *Implicit* transfer learning in the form of pretrained word representations has been a common component in natural language processing. In this dissertation, we argue that more *explicit* transfer learning is key to deal with the dearth of training data and to improve downstream performance of natural language processing models. We show experimental results transferring knowledge from related domains, tasks, and languages that support this hypothesis.

We make several contributions to transfer learning for natural language processing: Firstly, we propose new methods to automatically select relevant data for supervised and unsupervised domain adaptation. Secondly, we propose two novel architectures that improve sharing in multi-task learning and outperform single-task learning as well as the state-of-the-art. Thirdly, we analyze the limitations of current models for unsupervised cross-lingual transfer and propose a method to mitigate them as well as a novel latent-variable cross-lingual word embedding model. Finally, we propose a framework based on fine-tuning language models for sequential transfer learning and analyze the adaptation phase.

Acknowledgements

A PhD is a long journey through unknown lands, over sky-high peaks, and through deep and dark troughs. I am thankful for the many people I got to meet and who accompanied me on the way. First, I'd like to thank my PhD supervisors John Breslin and Parsa Ghaffari. The freedom you gave me to pursue my interests and follow my curiosity has made all the difference. John, thank you for your sensible advice and patience. Parsa, thank you for your support and for fostering a productive research environment at Aylien.

I want to thank my partner Jéssica who has been my advisor, sounding board, and confidante for her limitless patience. I am also grateful for the support of my family and friends at home in Germany and abroad. Special thanks go to my dad Harald and my brother Jona.

I am very fortunate to have met mentors along the way who took a chance on me and helped me grow. I am particularly grateful to Barbara Plank and Anders Søgaard. Barbara, working with you has been one of the high points of my PhD. Anders, the energy of you and your lab has catalyzed my growth and many future collaborations.

I also want to thank my colleagues at Aylien. In particular, I am grateful to my research team mates John Glover, Chris Hokamp, and Demian Gholipour for regular reading group meetings, inspiring discussions, and exchange of ideas. Thanks also to Sven Nebel, Fabio Cibecchini, and Robson Montenegro for occasional video game sessions.

I am grateful for all the amazing people I had the chance to get to know at numerous conferences and summer schools. I am looking forward to seeing you all at future events.

I would to like thank John Glover, Barbara Plank, Thomas Wolf, and Anders Søgaard for feedback on drafts of this thesis. I am grateful to my examiners Ivan Titov and Paul Buitelaar for much constructive feedback. Lastly, I would like to express my gratitude to the Irish Research Council, Aylien, and the Insight Centre (grant SFI/12/RC/2289) for funding during my PhD.

List of Figures

1.1 Contributions in this work. 5

3.1 The traditional supervised learning setup in machine learning. 43
3.2 The transfer learning setup. 43
3.3 A taxonomy for transfer learning for NLP. 46
3.4 Hard parameter sharing 48
3.5 Soft parameter sharing 49
3.6 The inductive transfer learning spectrum 64
3.7 The skip-gram model 72
3.8 The continuous bag-of-words model 72
3.9 A shared embedding space between two languages 101
3.10 Examples for the nature and type of cross-lingual alignment 105
3.11 Similar geometric relations in a cross-lingual embedding space 108

4.1 Bayesian Optimization development accuracy curves for POS tagging . . . 140
4.2 The Multi-task Tri-training model 149
4.3 Domain adaptation results for sentiment analysis 154
4.4 Accuracy comparison for different word frequency bins 158

5.1 Nearest neighbor graphs of English words and German translations. . . . 163
5.2 Influence of language-pair and domain similarity on bilingual lexicon induction performance 171
5.3 Impact of the frequency of query words 173
5.4 Correlation between bilingual lexicon induction performance and the proposed graph similarity metric 175
5.5 Partial lexicons of German and English shown as a bipartite graph 178
5.6 Bilingual dictionary induction results across different vocabulary sizes . . 188
5.7 Bilingual dictionary induction results with different priors 188

6.1 A sluice meta-network for two tasks. 195
6.2 The relative importance of a synthethic auxiliary task 200
6.3 Heat maps of learned α parameters in trained sluice networks 207
6.4 Multi-task learning with a Label Embedding Layer and a Label Transfer Network 212
6.5 Label embeddings of all tasks. 219
6.6 Development learning curves for the Label Transfer Network 221

7.1 The three stages of Universal Language Model Fine-tuning 227
7.2 The slanted triangular learning rate schedule 230

7.3 Few-shot learning results 235
7.4 Fine-tuning validation error rate curves 238
7.5 Performance of diagnostic classifiers trained on BERT representations . . 248
7.6 Mutual information between BERT representations and labels 249

List of Tables

2.1 Example annotations for all tasks discussed in this thesis 38

3.1 Notation related to cross-lingual word embeddings 103
3.2 Nature and alignment level of bilingual data sources 105
3.3 Cross-lingual embedding models ordered by data requirements. 107
3.4 Similarities and differences of three bilingual skip-gram variants 122
3.5 Overview of monolingual objectives and regularization terms 127

4.1 Statistics for the Amazon Reviews and the SANCL 2012 dataset. 136
4.2 Bayesian optimization results for sentiment analysis 138
4.3 Bayesian optimization results for part-of-speech tagging and parsing . . . 139
4.4 Cross-model transfer results for POS tagging 141
4.5 Cross-domain transfer results for sentiment analysis 142
4.6 Cross-domain transfer results for POS tagging 142
4.7 Cross-task transfer results 143
4.8 Average scores for sentiment analysis 154
4.9 Domain adaptation results for POS tagging with reduced training data . . 155
4.10 Domain adaptation results for POS tagging on the full training data . . . 156
4.11 Impact of out-of-vocabulary words and unknown word-tag combinations . 157

5.1 Languages used in previous work and in our experiments 168
5.2 Bilingual dictionary induction results results 168
5.3 Impact of hyper-parameters 172
5.4 Impact of the part-of-speech tag of query words 173
5.5 Impact of homographs as query words 174
5.6 Bilingual lexicon induction results 186
5.7 Cross-lingual similarity results 187
5.8 Hubs in English–German cross-lingual embedding space 189
5.9 Bilingual dictionary induction results for low-resource languages 190
5.10 Comparison of example translations 191

6.1 Statistics of the OntoNotes 5.0 dataset 202
6.2 Example annotations for chunking, named entity recognition, semantic role labelling, and part-of-speech tagging 202
6.3 Results for chunking with POS tagging as auxiliary task 204
6.4 Results for NER and SRL with POS tagging as auxiliary task 204
6.5 Performance of a model trained on all four tasks 205
6.6 Ablation analysis of a sluice network 206
6.7 Training set statistics and evaluation metrics of every task. 215

6.8 Example instances from the employed datasets. 215
6.9 Multi-task learning model comparison 217
6.10 Best-performing auxiliary tasks 219
6.11 Ablation results 220
6.12 Error analysis for the Label Transfer Network 221

7.1 Statistics of the text classification datasets 231
7.2 Results on two text classification datasets 234
7.3 Results on four large text classification datasets 234
7.4 Impact of pretraining 236
7.5 Impact of the language model 236
7.6 Impact of language model fine-tuning 236
7.7 Impact of classifier fine-tuning 237
7.8 Guidelines for feature extraction and fine-tuning 240
7.9 Feature extraction and fine-tuning results with ELMo and BERT 244
7.10 Comparison of cross-sentence embeddings methods with ELMo fine-tuning 245
7.11 Comparison of cross-sentence embedding methods with BERT feature extraction 246
7.12 NER performance for ELMo with feature extraction and fine-tuning . . . 246
7.13 Feature extraction and fine-tuning accuracy with BERT on MNLI domains 247

Abbreviations

General notation

cf.	**c**onf**e**r/**c**onf**e**ratur (*en*: compare)
e.g.	**e**xemplum **g**ratia (*en*: for example)
et al.	**et al**ia (*en*: and others)
i.e.	**i**d **e**st (*en*: that is)

Probability and statistics

CCA	**C**anonical **C**orrelation **A**nalysis
KL	**K**ullback-**L**eibler
JS	**J**ensen-**S**hannon
PCA	**P**rincipal **C**omponent **A**nalysis
PDF	**P**robability **D**ensity **F**unction
PMF	**P**robability **M**ass **F**unction
SVD	**S**ingular **V**alue **D**ecomposition

Machine learning

CV	**C**omputer **V**ision
i.i.d.	**i**ndependent and **i**dentically **d**istributed
LASSO	**L**east **A**bsolute **S**hrinkage and **S**election **O**perator
ML	**M**achine **L**earning
MTL	**M**ulti-**T**ask **L**earning
MSE	**M**ean **S**quared **E**rror
SGD	**S**tochastic **G**radient **D**escent
SSL	**S**emi-**S**upervised **L**earning
SVM	**S**upport **V**ector **M**achine

Neural networks

BPTT	**B**ack-**P**ropagation **T**hrough **T**ime
CNN	**C**onvolutional **N**eural **N**etwork
GAN	**G**enerative **A**dversarial **N**etwork
LSTM	**L**ong **S**hort-**T**erm **M**emory
NN	**N**eural **N**etwork
ReLU	**Re**ctified **L**inear **U**nit
RNN	**R**ecurrent **N**eural **N**etwork

Natural language processing

BDI	**B**ilingual **D**ictionary **I**nduction
BOW	**B**ag-**O**f-**W**ords
CBOW	**C**ontinuous **B**ag-**O**f-**W**ords
CRF	**C**onditional **R**andom **F**ield
EM	**E**xpectation **M**aximization
HMM	**H**idden **M**arkov **M**odel
KB	**K**nowledge **B**ase
LDA	**L**atent **D**irichlet **A**llocation
LM	**L**anguage **M**odel
LSA	**L**atent **S**emantic **A**nalysis
MT	**M**achine **T**ranslation
NER	**N**amed **E**ntity **R**ecognition
NLP	**N**atural **L**anguage **P**rocessing
OOV	**O**ut-**O**f-**V**ocabulary
PCA	**P**rinciple **C**omponent **A**nalysis
POS	**P**art-**O**f-**S**peech
QA	**Q**uestion **A**nswering
SGNS	**S**kip-**G**ram with **N**egative **S**ampling
SRL	**S**emantic **R**ole **L**abelling
tf-idf	**t**erm **f**requency-**i**nverse **d**ocument **f**requency

Notation

Probability and statistics

P	Probability mass function
p	Probability density function
$\hat{P}$	Empirical distribution function
σ	Standard deviation
σ^2	Variance
ρ	Pearson's r
$\mathcal{N}$	Gaussian distribution

Machine learning

η	Learning rate
$\mathcal{T}$	Task
$\mathcal{D}$	Domain
$\mathbf{W}$	Weight matrix
X	Set of training examples
Y	Set of training labels
θ	Model parameters
$\mathcal{L}$	Loss function

We use the following conventions throughout this thesis: We use bold lower case letters ($\mathbf{x}$) to denote vectors, bold upper case letters ($\mathbf{X}$) to denote matrices, and standard weight letters (x) for scalars. We will use subscripts with bold letters ($\mathbf{x}_i$) to refer to entire rows or columns and subscripts with standard weight letters for specific elements (x_i). We also use subscripts to denote different variables ($\mathbf{W}_1$, $\mathbf{W}_2$). To avoid ambiguity, we may sometimes use bracketed superscripts for the latter purpose instead ($\mathbf{W}_i^{(1)}$, $\mathbf{W}_i^{(2)}$).

To my parents, Sibylle and Harald.

Chapter 1

Introduction

1.1 Motivation

Language is often regarded as the hallmark of human intelligence. Developing systems that can understand human language is thus one of the main obstacles on the quest towards artificial general intelligence. This objective has driven research in artificial intelligence and particularly in natural language processing and computational linguistics. As language permeates every part of the human existence, natural language processing is ultimately necessary for computers to achieve their full potential in augmenting human intelligence.

Early symbolic approaches towards this elusive goal tried to capture the meaning of text using rules written by humans. Such rule-based systems, however, were brittle and limited to the particular domains they had been designed for [Winograd, 1972]. They generally were unable to deal with unexpected or unseen inputs and ultimately proved too restrictive to capture the intricacy of natural language [National Research Council and Automatic Language Processing Advisory Committee, 1966].

Over the last 20 years, a statistical approach of natural language processing [Manning et al., 1999] has become commonplace, which uses mathematical models to automatically learn rules from data. Consequently, rather than writing rules, human effort was channelled into creating features that tell a model what connections and relationships in the data it should consider to make its prediction. Engineering features, however, is time-consuming as features are generally task-specific and require domain expertise.

In the span of the past five years, deep neural networks [Krizhevsky et al., 2012a, Goodfellow et al., 2016], a particular category of machine learning models, have become the model of choice when learning from data. These models automatically learn a

multi-layered hierarchy of features and thus reduce the need for feature engineering. Human energy consequently has focused on determining the most suitable architecture and training setting for each task.

In natural language processing as well as in many areas of machine learning, the standard way to train a model is to annotate a large number of examples that are then provided to the model, which learns a function that maps from inputs to outputs. This is known as supervised learning. For every task such as analyzing the syntactic structure of a text, disambiguating words, or translating a document, a new model is trained from scratch. Knowledge from related tasks or domains is never combined and models always start tabula rasa, from a random initialization.

Learning from such a blank state is antithetic to the way humans acquire language. Human language learning does not occur in isolation but in a rich sensory environment. Children learn language via interaction with their surroundings [Hayes et al., 2002] and through continuous feedback and reinforcement [Bruner, 1985].

Nevertheless, recent deep neural network-based approaches have achieved remarkable successes on a wide range of tasks by learning from hundreds of thousands to millions of input–output pairs such as in machine translation [Wu et al., 2016]. Given these successes, one might think that there is no need to stray from the paradigm of supervised learning, that it is unnecessary to create algorithms inspired by human language acquisition. After all, nature has served us well as an inspiration rather than a blueprint; for instance, artificial neural networks are only loosely inspired by human cognition [Rumelhart et al., 1986].

Recent studies [Jia and Liang, 2017, Belinkov and Bisk, 2018], however, show that current algorithms are brittle in a way similar to early rule-based systems: They do not generalize beyond the data they have seen during training. They conform to the characteristics of the data they have been trained on and are not able to adapt when conditions change.

The needs of humans are complex and language is diverse; new tasks—from identifying new precedents in legal documents, mining unseen drug interactions, to routing support emails—are therefore constantly required to be solved using natural language processing. Natural language processing also promises to help bridge the digital language divide[1], which leads to an inequality of information—and opportunity—online. To this end, models need to be applied not only to English but to the world's 6,000 languages.

In order to obtain a model that performs well on data that has not been seen before–whether from a new task, domain, or language—, supervised learning requires a sufficient

[1] `http://labs.theguardian.com/digital-language-divide/`

number of examples to be labelled for every new setting. Given the plethora of languages, tasks, and domains in the real world, manually annotating examples for every setting is simply infeasible. Standard supervised learning thus breaks down in light of these real-world challenges.

Transfer learning promises to ameliorate this failing by transferring knowledge from related domains, tasks, and languages to the target setting. Indeed, transfer learning has long been a latent part of many NLP systems. Many of the most fundamental advances in NLP such as latent semantic analysis [Deerwester et al., 1990], Brown clusters [Brown et al., 1993b], and pretrained word embeddings [Mikolov et al., 2013a] can be considered as particular forms of transfer learning, as a means of transferring knowledge from a general-purpose source task to a more specialized target task.

In this dissertation, we argue that framing the training of NLP models as transfer learning rather than supervised learning can help unlock new potential that will allow our models to generalize better. To this end, we develop novel models that transfer across domains, tasks, and languages for a variety of scenarios. We demonstrate that our models outperform both existing transfer learning methods as well as models that do not transfer.

1.2 Research objectives

This thesis studies the problem of automatically learning representations that transfer across tasks, domains, and languages with neural network-based methods for natural language processing. The main hypothesis of this thesis is the following:

Deep neural networks in natural language processing that leverage existing relevant information from related domains, tasks, and languages outperform models not using this information across a wide range of tasks.

In other words, we argue that in most settings, transfer learning outperforms supervised learning, with two caveats:

1. Transfer learning may be less helpful when already a sufficient number of training examples are available.
2. Transfer learning may be less useful if no relevant information is available.

To address the first aspect, we analyze the few-shot capabilities of transfer learning (§7.1.5). The second points hints at a recurring theme in this thesis: The success of transfer learning is dependent on the similarity of the source setting to the target setting.

Overall, we lay out five desiderata that will be addressed by the approaches proposed in this thesis:

1. **Overcoming a discrepancy between source and target setting**: The method should overcome a discrepancy between the source and target settings. Many existing methods only work well when source and target settings are similar. To overcome this challenge, we propose methods that select relevant examples (§4.1), leverage weak supervision (§5), flexibly share parameters across tasks (§6), learn general-purpose representations (§7.1), and analyze task similarity (§7.2).
2. **Inducing an inductive bias**: The model should induce an inductive bias that improves its ability to generalize. Inductive biases that we employ include semi-supervised learning (§4.2, §6.2), multi-task learning (§4.2, §6) an orthogonality constraint (§4.2, §6.1), weak supervision (§5), a matching prior (§5.2), hierarchical relations (§6.1), and pretrained representations (§7).
3. **Combining traditional and current approaches**: The model should take inspiration from classic work to overcome the limitations of state-of-the-art approaches. We propose two models that explicitly combine the best of both worlds, the strengths of traditional and neural methods (§4.2, §5.2).
4. **Transfer across the hierarchy of NLP tasks**: The approach should transfer knowledge across the hierarchy of NLP tasks. This includes sharing across low-level and high-level tasks (§6.1), sharing between coarse-grained and fine-grained sentiment tasks (§6.2), and transfer from a general-purpose task to a diverse set of tasks (§7).
5. **Generalization across many settings**: The method should enable generalization to many different settings. To test this, we evaluate each method across a wide range of tasks (§4, §6, §7), domains (§4, §5.1, §6, §7), and languages (§5).

1.3 Contributions

Throughout this thesis, we will focus on the three main dimensions of transfer learning for NLP: transfer across domains, transfer across tasks, and transfer across languages. These three dimensions can be naturally separated into four different transfer learning settings based on the nature of the source and target tasks and domains and the order of learning: domain adaptation, cross-lingual learning, multi-task learning, and sequential transfer learning. We show how the contributions in this thesis relate to these four settings in Table 1.1.

The contributions in this thesis can be categorized based on their theoretical, practical, and empirical impact. In terms of theoretical contributions,

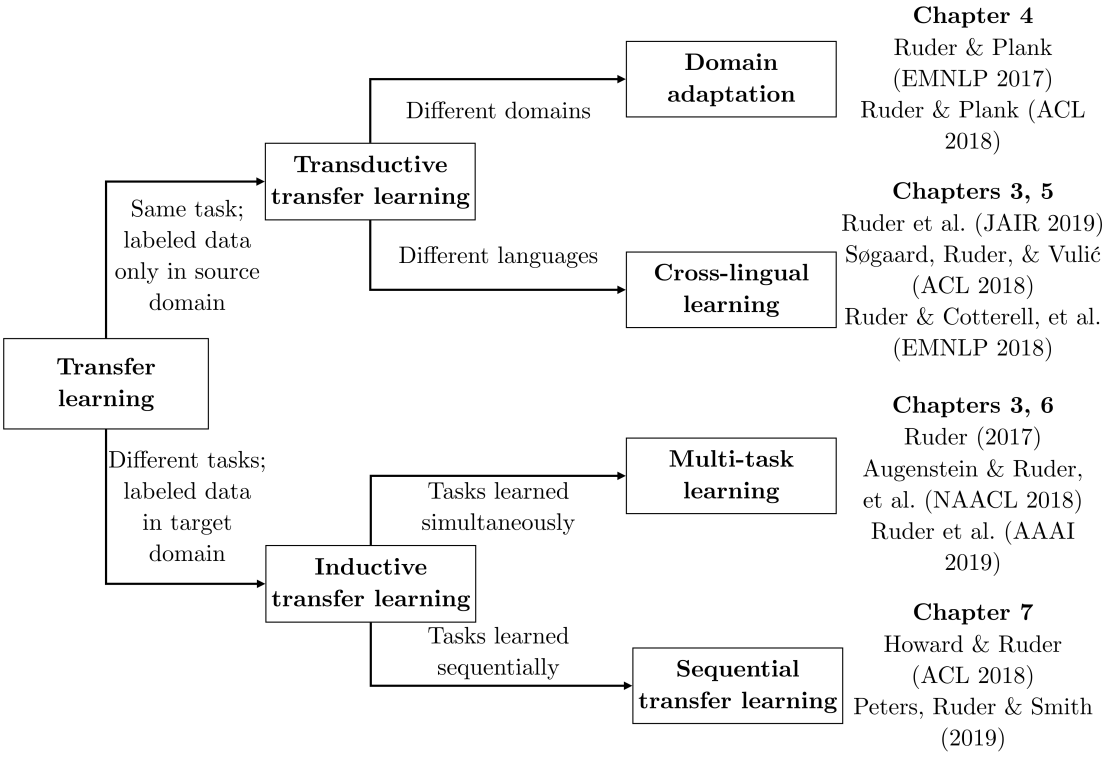


FIGURE 1.1: Contributions in this work.

- we present a taxonomy that reflects the most common transfer learning settings in natural language processing (§3.1.3);
- we show that cross-lingual word embedding models that learn on the word level optimize similar objectives (§3.5.4);
- we analyze the theoretical limitations of unsupervised cross-lingual embedding models (§5.1);
- we show how existing cross-lingual embedding approaches can be viewed as a latent-variable model (§5.2);
- and we propose a theoretical framework that generalizes over existing architectures to multi-task learning (§6.1).

In terms of practical contributions,

- we present extensive reviews of the four most common transfer learning settings in natural language processing: multi-task learning (§3.2), sequential transfer learning (§3.3), domain adaptation (§3.4), and cross-lingual learning (§3.5);
- we propose a novel eigenvector-based metric to gauge the potential of unsupervised bilingual dictionary induction between two languages (§5.1);
- we provide guidelines for adapting pretrained representations (§7.2);

- and we open-source our code[2].

We finally make the following empirical contributions:

- we present a model that automatically learns to select training examples that are relevant for a particular target domain (§4.1);
- we adapt semi-supervised learning methods to neural networks and compare them against state-of-the-art approaches (§4.2);
- we propose a more efficient semi-supervised learning method inspired by tri-training (§4.2);
- we empirically analyze the limitations of unsupervised cross-lingual word embedding models (§5.1);
- we present a novel latent-variable model for bilingual lexicon induction (§5.2);
- we present a novel multi-task learning model that automatically learns which layers to share between different tasks (§6.1);
- we present a novel multi-task learning model that integrates information from disparate label spaces (§6.2);
- we propose a novel framework for sequential transfer learning using pretrained language models and novel fine-tuning techniques (§7.1);
- and we compare the two prevalent adaptation methods with state-of-the-art pretrained representations on a diverse range of tasks (§7.2).

1.4 Thesis outline

In Chapter 2, we provide an overview of background information that is relevant in order to understand the contents of this thesis. We review fundamentals of probability and information theory and machine learning. We furthermore discuss neural network-based methods and tasks in natural language processing.

In Chapter 3, we define transfer learning and present a taxonomy for transfer learning for NLP. We then review the four transfer learning scenarios in detail: domain adaptation, cross-lingual learning, multi-task learning, and sequential transfer learning.

The following chapters focus on each of these scenarios. In each chapter, we will present novel methods for the respective setting that outperform the state-of-the-art on benchmark datasets.

Chapter 4 presents our work on data selection for domain adaptation. For supervised domain adaptation, we propose an approach that uses Bayesian Optimization to learn a

[2] `https://github.com/sebastianruder`

policy for selecting relevant training examples from multiple domains. For unsupervised domain adaptation, we adapt classic semi-supervised learning methods to neural networks and propose a novel method inspired by tri-training. Both approaches aim to select relevant examples that are either similar to the target domain or reliable and informative according to the semi-supervised learning model.

In Chapter 5, we first analyze the limitations of unsupervised cross-lingual word embedding models. We find that existing unsupervised approaches break down in the setting where languages are dissimilar and provide a weakly supervised method to ameliorate this. We furthermore propose a latent variable model with a regularizing matching prior that works well for low-resource languages. In addition, we provide a new perspective on existing cross-lingual word embedding models through the lens of latent variables.

In Chapter 6, we propose two novel architectures that improve sharing between tasks in multi-task learning. In multi-task learning, current approaches such as hard parameter sharing break down when tasks are dissimilar. Our first approach overcomes this by allowing the model to learn to what degree information between tasks should be shared. Our second approach incorporates information from the label spaces of other tasks.

In Chapter 7, we focus on the previously neglected adaptation phase in sequential transfer learning. We first propose a novel framework based on language modelling and new techniques for adaptation. We secondly analyze adaptation with state-of-the-art pretrained representations. We find that task similarity plays an important role and provide guidelines to the practitioner.

Chapter 8 finally contains our conclusion where we summarize our findings and provide an outlook into the future.

1.5 Publications

The work in this dissertation primarily relates to the following peer-reviewed articles (in order of publication):

1. **Ruder, S.** and Plank, B. (2017). Learning to select data for transfer learning with Bayesian Optimization. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing.*
2. Augenstein, I.[‡], **Ruder, S.**[‡], and Søgaard, A. (2018). Multi-task Learning of Pairwise Sequence Classification Tasks Over Disparate Label Spaces. In *Proceedings of NAACL-HLT 2018.*

[‡]Equal contribution.

3. Søgaard, A., **Ruder, S.** and Vulić, I. (2018). On the Limitations of Unsupervised Bilingual Dictionary Induction. In *Proceedings of ACL 2018*.
4. **Ruder, S.** and Plank, B. (2018). Strong Baselines for Neural Semi-supervised Learning under Domain Shift. In *Proceedings of ACL 2018*.
5. Howard, J.‡ and **Ruder, S.**‡ (2018). Universal Language Model Fine-tuning for Text Classification. In *Proceedings of ACL 2018*.
6. **Ruder, S.**‡, Cotterell, R.‡, Kementchedjhieva, Y., and Søgaard, A. (2018). A Discriminative Latent-Variable Model for Bilingual Lexicon Induction. In *Proceedings of EMNLP 2018*.
7. **Ruder, S.**, Bingel, J., Augenstein, I., and Søgaard, A. (2019). Latent Multi-task Architecture Learning. In *Proceedings of AAAI 2019*.
8. **Ruder, S.**, Vulić, I., and Søgaard, A. (2019). A Survey of Cross-lingual Word Embedding Models. To be published in *Journal of Artificial Intelligence Research*.

The following preprints are also discussed:

9. **Ruder, S.** (2017). An Overview of Multi-Task Learning in Deep Neural Networks. *arXiv preprint arXiv:1706.05098*.
10. Peters, M.‡, **Ruder, S.**‡, and Smith, N. A. (2019). To Tune or Not to Tune? Adapting Pretrained Representations to Diverse Tasks. *arXiv preprint arXiv:1903.05987*.

The following articles are related, but will not be extensively discussed in this thesis:

11. **Ruder, S.**, Ghaffari, P., and Breslin, J. G. (2016). Towards a continuous modeling of natural language domains. In *Uphill Battles in Language Processing Workshop, EMNLP 2016*.
12. **Ruder, S.**, Ghaffari, P., and Breslin, J. G. (2017). Knowledge Adaptation: Teaching to Adapt. *arXiv preprint arXiv:1702.02052*.
13. **Ruder, S.**, Ghaffari, P., and Breslin, J. G. (2017). Data Selection Strategies for Multi-Domain Sentiment Analysis. *arXiv preprint arXiv:1702.02426*.
14. Kementchedjhieva, Y., **Ruder, S.**, Cotterell, R., and Søgaard, A. (2018). Generalizing Procrustes Analysis for Better Bilingual Dictionary Induction. In *Proceedings of CoNLL 2018*.
15. Sanh, V., Wolf, T., and **Ruder, S.** (2019). A Hierarchical Multi-task Approach for Learning Embeddings from Semantic Tasks. In *Proceedings of AAAI 2019*.

Finally, while not directly related, the following articles have also been completed over the course of the PhD:

16. **Ruder, S.** (2016). An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*.

17. **Ruder, S.**, Ghaffari, P., and Breslin, J. G. (2016). A Hierarchical Model of Reviews for Aspect-based Sentiment Analysis. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP-16)*, pp. 999–1005.
18. **Ruder, S.**, Ghaffari, P., and Breslin, J. G. (2016). INSIGHT-1 at SemEval-2016 Task 4: Convolutional Neural Networks for Sentiment Classification and Quantification. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval 2016)*, pp. 178–182.
19. **Ruder, S.**, Ghaffari, P., and Breslin, J. G. (2016). INSIGHT-1 at SemEval-2016 Task 5: Deep Learning for Multilingual Aspect-based Sentiment Analysis. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval 2016)*.
20. **Ruder, S.**, Ghaffari, P., and Breslin, J. G. (2016). Character-level and Multi-channel Convolutional Neural Networks for Large-scale Authorship Attribution. *arXiv preprint arXiv:1609.06686*.
21. **Ruder, S.**, Glover, J., Mehrabani, A., and Ghaffari, P. (2018). 360° Stance Detection. In *Proceedings of NAACL-HLT 2018: System Demonstrations*, New Orleans, US.

Chapter 2

Background

This chapter provides background knowledge to set the stage for the subsequent chapters. It reviews fundamentals of probability and information theory (§2.1) that form the cornerstones of the techniques introduced throughout this thesis. It then introduces the reader to machine learning (§2.2) together with its most elementary methods.

We subsequently delve into the particular type of machine learning models that will be mostly used in this thesis, neural networks (§2.3). Finally, we give an overview of common tasks in natural language processing (§2.4).

2.1 Probability and information theory

Probability theory provides us with the lingua franca to discuss and analyze many of the methods presented throughout this thesis that are probabilistic in nature. Using probability theory, we can make statements about how likely it is that an event will occur given that other events already have occurred.[1] Throughout this thesis, we will seek to model such likelihoods computationally. Information theory similarly provides us with tools to describe the information encoded in events and measures to characterize a difference in information. The latter is a property that we often intend to minimize using machine learning methods.

2.1.1 Probability basics

Random variable At the heart of probability theory are *random variables*, which are functions that output specific values whose range depends on some underlying random

[1] Note that this is a frequentist interpretation of probability. From a Bayesian perspective, we would be concerned with the degree of belief that an event will occur.

process. For instance, a random variable X may take on the values x_1 and x_2. A random variable has a *probability distribution*, which specifies with what probability it takes on each of its values. Our goal in this thesis will generally be to define systems that allow us to model the probability distribution of such random variables.

PMF and PDF A random variable can be *discrete*, which means that it takes on a finite number of values. For discrete random variables, we define its probability distribution with a *probability mass function* (PMF), typically denoted as P. The probability mass function maps a value of a random variable to the probability of the random variable taking on that value.

$P(X = x_1)$ thus indicates the probability of X taking on the value x_1. If it is unambiguous to which random variable an event belongs, we will use $P(x_1)$ instead. Every event $x \in X$ has a probability $0 \leq P(x) \leq 1$. An impossible event has a probability of 0, while an event that is guaranteed to happen has a probability of 1.

If a random variable is able to take on any value in an interval, it is said to be *continuous* and we use a *probability density function* (PDF), usually designated as p to specify its probability distribution. In contrast to PMFs, PDFs do not provide the probability of a specific event. In fact, the probability of any specific point within the interval is 0. We rather measure the probability of landing inside an infinitesimally small region with volume δx with $p(x)\delta x$.

Both PMFs and PDFs are *normalized*: For a PMF, all probabilities must sum to 1, i.e. $\sum_{x \in X} P(x) = 1$, while for a PDF, the probabilities must integrate to 1, i.e. $\int_x p(x)dx$.

Joint, marginal, and conditional probability distribution A *joint probability distribution* is a probability distribution over multiple random variables at the same time. For events $x \in X$ and $y \in Y$, $P(X = x, Y = y)$ denotes the probability of both events happening simultaneously. We typically write $P(X = x, Y = y)$ as $P(x, y)$ for brevity.

Given a joint probability distribution, a *marginal probability distribution* is the probability distribution over a subset of the variables. Given $P(X, Y)$, we can calculate $P(x)$ for all $x \in X$ with the *sum rule* of probability:

$$P(x) = \sum_y P(x, y). \tag{2.1}$$

We essentially sum over the joint probabilities for all values of the variables that we are not interested in. For continuous random variables, we integrate instead:

$$p(x) = \int_y p(x, y)dy. \tag{2.2}$$

For making predictions about events, which is key for the models used in this thesis, we require computing the probability of an event given that some other event has occurred, also known as *conditional probability*. We denote the conditional probability of an event y given an event x as $P(y \mid x)$. We can define this probability as the joint probability of the events divided by the marginal probability of the event that already occurred:

$$P(y \mid x) = \frac{P(y, x)}{P(x)}. \tag{2.3}$$

By applying this definition repeatedly, we can decompose a joint probability distribution over multiple random variables into a product of conditional distributions over only one variable:

$$\begin{aligned} P(x, y, z) &= P(x \mid y, z)P(y, z) \\ P(y, z) &= P(y \mid z)P(z) \\ P(x, y, z) &= P(x \mid y, z)P(y \mid z)P(z). \end{aligned}$$

This formula is known as the *chain rule* or *product rule* of probability. For n events, it is defined as follows:

$$P(x_1, \ldots, x_n) = P(x_1) \prod_{i=2}^{n} P(x_2 \mid x_1, \ldots, x_{i-1}). \tag{2.4}$$

Independence Two random variables X and Y are *independent* if their joint probability distribution can be decomposed as a product of their individual probability distributions, i.e. for all $x \in X$ and $y \in Y$:

$$p(x, y) = p(x)p(y) \tag{2.5}$$

Two random variables X and Y are said to be *conditionally independent* given a random variable Z if we can decompose their joint probability distribution as conditional probability distributions, one involving X and the other involving Y, for all $x \in X$, $y \in Y$, and $z \in Z$:

$$p(x, y \mid z) = p(x \mid z)p(y \mid z). \tag{2.6}$$

Bayes' Rule In machine learning, we often would like to estimate the probability of $P(y \mid x)$ given knowledge of $P(x \mid y)$. If we also know $P(y)$, we can derive $P(y \mid x)$ using Equation 2.3:

$$\begin{aligned} P(y \mid x) &= \frac{P(x,y)}{P(x)} \\ P(x,y) &= P(x \mid y)P(y) \\ P(y \mid x) &= \frac{P(x \mid y)P(y)}{P(x)}. \end{aligned} \tag{2.7}$$

$P(y \mid x)$ is known as the *posterior probability* of y, $P(x \mid y)$ is the *likelihood* of x given y, $P(y)$ is the *prior probability* of y, and $P(x)$ is the *evidence*. The last equation is also known as *Bayes' rule* and is at the heart of *Bayesian methods* for machine learning. Bayesian Optimization (§4.1) is such a Bayesian approach. In the other chapters, we will focus on *frequentist methods* that do not explicitly model the prior $P(y)$. Our methods, however, will often have a bias (§2.2.4) that implicitly expresses a prior belief such as via regularization (§2.2.5).

Expectation In machine learning, we are often concerned with functions over random variables. One particular quantity of interest is the value a function $f(x)$ typically takes on with regard to a probability distribution $P(X)$ when x is drawn from P. This average value is also known as *expectation*, *expected value*, or the *mean* μ of $f(x)$ with respect to $P(x)$. For discrete variables, we can calculate the expectation via summation:

$$\mathbb{E}_{x \sim P}[f(x)] = \sum_x P(x)f(x) \tag{2.8}$$

where $x \sim P$ denotes that x is drawn from P. In essence, the expectation is the average over all values of a function weighted with their probability. For continuous variables, we again integrate:

$$\mathbb{E}_{x \sim p}[f(x)] = \int p(x)f(x)dx. \tag{2.9}$$

For brevity, we will often write $\mathbb{E}_x[f(x)]$ when it is clear from which distribution x is drawn and $\mathbb{E}[f(x)]$ if the random variable is also evident from context. If $f(x)$ is the identity, meaning $f(x) = x$, then the expectation is simply the mean of P:

$$\mu = \sum_x P(x)x$$

The *linearity of expectation* is a well-known property of the expectation, which states that the expectation of a sum of random variables is equal to the sum of their individual

expectations:

$$\mathbb{E}_x[af(x) + bg(x)] = a\mathbb{E}_x[f(x)] + b\mathbb{E}_x[g(x)]. \tag{2.10}$$

The *sample mean* s_n is the average value, if we sample x from $P(X)$ n times:

$$s_n = \frac{1}{n}\sum_{i=1}^{n} x_i \tag{2.11}$$

According to the *law of large numbers*, this sample average converges to the expectation of $P(X)$ as $n \to \infty$. Another useful property is given by the *central limit theorem*, which states that as n gets larger the sampling distribution of $\sqrt{n}(s_n - \mathbb{E}_x)$ approximates a normal distribution (§2.1.2).

Variance Beyond the average value, we are also often interested in measuring how much the values of a function $f(x)$ vary from this mean value as we sample different values of x from P. This quantity is known as *variance* of a function $f(x)$ and is defined as the expectation of the squared difference—or *deviation*—from its mean:

$$\mathrm{Var}(f(x)) = \mathbb{E}\Big[(f(x) - \mathbb{E}[f(x)])^2\Big]. \tag{2.12}$$

By expanding the binomial, we obtain the common mnemonic for the variance as the "mean of square minus square of mean":

$$\mathrm{Var}(f(x)) = \mathbb{E}[f(x)^2] - \mathbb{E}[f(x)]^2. \tag{2.13}$$

The *standard deviation* is simply the square root of the variance and is typically denoted as σ, while the variance is specified as σ^2. As the expectation of a square, the variance is always positive. Mean and variance are also known as *first* and *second moment* respectively of a probability distribution.

Going beyond just a single random variable, we can use the measure of *covariance* to gauge how much two random variables X and Y are linearly related:

$$\mathrm{Cov}(X, Y) = \mathbb{E}\Big[(X - \mathbb{E}[X])(Y - \mathbb{E}[Y])\Big]. \tag{2.14}$$

Two variables that are independent have zero covariance and two variables that have non-zero covariance are dependent.

The *Pearson correlation coefficient*, also known as Pearson's r and commonly denoted as ρ, measures how much two random variables X and Y are linearly correlated. It is defined as the covariance of the two variables divided by the product of their standard

deviations:

$$\rho(X, Y) = \frac{\text{Cov}(X, Y)}{\sigma_X \sigma_Y} \tag{2.15}$$

We will use the Pearson correlation coefficient as a way of measuring how well the predictions of our model correlate with the true values, such as for determining the similarity between words (§7.1.4).

In machine learning, many of our variables are not scalars, but vectors. Covariance extends naturally to vector random variables. The *covariance matrix* of a random vector $\mathbf{x} \in \mathbb{R}^n$ is an $n \times n$ matrix where

$$\text{Cov}(\mathbf{x})_{i,j} = \text{Cov}(x_i, x_j) \tag{2.16}$$

and the diagonal elements are simply the variance.

Canonical correlation analysis (CCA) is a method to find linear combinations of two vector random variables X and Y that have maximum correlation with each other. Specifically, we seek vectors $\mathbf{a}$ and $\mathbf{b}$ such that the random variables $\mathbf{a}^\top X$ and $\mathbf{b}^\top Y$ maximize the correlation $\rho(\mathbf{a}^\top X, \mathbf{b}^\top Y)$. The solution can be computed using singular value decomposition (SVD) on a correlation matrix. CCA has been used in domain adaptation and to learn representations of words in one and multiple languages (§3.4.2.2, §3.3.2.3, and §3.5.4.1 respectively).

2.1.2 Distributions

In machine learning, we will often make use of some common probability distributions. The simplest probability distribution is the *Bernoulli distribution*, which is a discrete distribution over a single binary random variable. It is defined by a parameter $\phi \in [0, 1]$, which controls the probability of the random variable X being equal to 1: $P(X = 1) = \phi$. Conversely, $P(X = 0) = 1 - \phi$. Together, they yield the PMF of the Bernoulli distribution:

$$P(X = x) = \phi^x (1 - \phi)^{1-x} \tag{2.17}$$

Throughout this thesis, binary classification, predicting one of two possible values, will be a common evaluation task. For this task, the output random variable follows a Bernoulli distribution. Logistic regression (see Equation 2.41) is a common example of a method that employs such an output distribution.

The *categorical distribution* or *multinoulli distribution* generalizes the Bernoulli distribution to a discrete random variable that can take on K different values. The output random variable for multi-class classification tasks follows a categorical distribution.

Both of the above are distributions over discrete random variables. The most commonly used distribution over a continuous random variable is the *Gaussian distribution* or *normal distribution*:

$$\mathcal{N}(x; \mu, \sigma^2) = \sqrt{\frac{1}{2\pi\sigma^2}} \exp(-\frac{1}{2\sigma^2}(x-\mu)^2). \tag{2.18}$$

In Section 5.2, we will use a Gaussian distribution to parameterize our model.

The final distribution that we will often employ is the *empirical distribution* $\hat{P}$ of our data. If our data is discrete, then we can simply use a multinoulli distribution where the probability of each data point is its frequency during training:

$$\hat{P}(\mathbf{x}) = \frac{1}{n} \sum_{i=1}^{n} \mathbf{1}_{\mathbf{x}_i = \mathbf{x}} \tag{2.19}$$

where $\mathbf{1}$ is an indicator function that is 1 if $\mathbf{x}_i = \mathbf{x}$ and otherwise 0. If our data is continuous, the empirical distribution is given by:

$$\hat{p}(\mathbf{x}) = \frac{1}{n} \sum_{i=1}^{n} \delta(\mathbf{x} - \mathbf{x}_i) \tag{2.20}$$

where $\delta(X)$ is the Dirac delta function, which is loosely defined so that $\int_{-\infty}^{\infty} \delta(x)dx = 1$, $\delta(X = 0) = \infty$ and $\delta(X) = 0$ otherwise. In the empirical distribution, as we shift $\delta(X)$ by $-\mathbf{x}_i$, we put probability mass of $1/n$ on each of the data points $\mathbf{x}_1, \ldots, \mathbf{x}_n$. We will generally use the empirical or *data generating distribution* to approximate the true underlying distribution that we would like to learn.

2.1.3 Information theory

Information theory [Shannon, 1948] was originally proposed to study the information contained in signals passed through a noisy channel. It is now at the heart of many disciplines, among them machine learning. Throughout this thesis, we will use measures from information theory to characterize the discrepancy between two probability distributions in terms of the information that they encode. This will provide us with a measure of "goodness" of the probability distribution that our model is learning compared to the empirical distribution of the data.

Self-information and entropy A fundamental measure of information theory is *self-information*, which allows us to determine the information content of an event $x \in X$:

$$I(x) = -\log P(x) \tag{2.21}$$

where log is the natural logarithm with *nat* (short for 'natural') as unit.

Another core quantity of information theory is *Shannon entropy*, which gauges the expected amount of information when an event x is drawn from a probability distribution P:

$$H(x) = \mathbb{E}_{x\sim P}[I(x)] = -\mathbb{E}_{x\sim P}[\log P(x)]. \tag{2.22}$$

The Shannon entropy effectively measures how much uncertainty is contained in a probability distribution.

We can extend the notion of entropy to two distributions and measure the *relative entropy* of $P(x)$ with respect to another probability distribution $Q(x)$, which is defined as:

$$D_{KL}(P||Q) = \mathbb{E}_{x\sim P}\Big[\log \frac{P(x)}{Q(x)}\Big] = \mathbb{E}_{x\sim P}[\log P(x) - \log Q(X)]. \tag{2.23}$$

This measure is also known as *Kullback-Leibler divergence* (KL divergence) or—less commonly—as *information gain*. The KL divergence between two distribution is always non-negative and zero if and only if the two distributions are equal.

Even though the Kullback-Leibler divergence is often used to measure the distance between probability distributions, it is asymmetric, i.e. generally $D_{KL}(P||Q) \neq D_{KL}(Q||P)$. A symmetric alternative is the *Jensen-Shannon (JS) divergence*:

$$JSD(P||Q) = \frac{1}{2}D_{KL}(P||M) + \frac{1}{2}D_{KL}(Q||M) \tag{2.24}$$

where $M = \frac{1}{2}(P + Q)$. We will describe the KL and JS divergences as measures of distance between natural language domains (§3.4.2.1) and use them in our experiments (§4.1).

Another quantity that is closely related to the KL divergence is the *cross-entropy*, which is defined as follows:

$$H(P, Q) = -\mathbb{E}_{x\sim P}[\log Q(X)]. \tag{2.25}$$

By adding and subtracting $\mathbb{E}_{x\sim P}[\log P(x)]$ and using the linearity of expectation, we obtain:

$$\begin{aligned} H(P, Q) &= -\mathbb{E}_{x\sim P}[\log P(x)] + \mathbb{E}_{x\sim P}[\log P(x)] - \mathbb{E}_{x\sim P}[\log Q(X)] \\ &= -\mathbb{E}_{x\sim P}[\log P(x)] + \mathbb{E}_{x\sim P}[\log P(x) - \log Q(X)] \end{aligned}$$

The term on the left is now just the definition of entropy, while the term on the right is the definition of the KL divergence between P and Q. Substituting the expressions from Equations 2.22 and 2.23 respectively for these, we obtain a decomposition of cross-entropy

as the sum of the entropy and KL divergence:

$$H(P, Q) = H(P) + D_{KL}(P||Q) \tag{2.26}$$

Pointwise mutual information (PMI) is another measure of the association between the outcomes of two discrete random variables X and Y. Specifically, it measures the discrepancy between their joint and the product of their individual probabilities:

$$PMI(x, y) = \log \frac{P(x, y)}{P(x)P(y)} \tag{2.27}$$

A classic approach in NLP is to compute the PMI between every pair of words in a corpus, which is stored in a matrix. Latent semantic analysis [Deerwester et al., 1990] (§3.3.2.3) can then be used to factorize such a matrix. *Mutual information* (MI), finally, is the expected value of the PMI:

$$I(X; Y) = \sum_{y \in Y} \sum_{x \in X} PMI(x, y) \tag{2.28}$$

We will use MI to characterize the association between learned representations and output labels (§7.2).

2.2 Machine learning

In this section, we introduce the reader to machine learning, which builds mathematical models from data. Many of the concepts introduced in this section will reappear throughout the thesis, either forming the building blocks used in more advanced neural network-based methods (§2.3) or supplying the theory that underpins many of the proposed models. In particular, we will frequently revisit the topic of generalization in machine learning (§2.2.4) as we will be seeking to create models that generalize to other domains, tasks, and languages.

In machine learning, each input is typically represented as a vector $\mathbf{x} \in \mathbb{R}^d$ of d *features*, where each feature contains the value for a particular attribute of the data and each example is assumed to be drawn independently from the data generating distribution $\hat{p}_{\text{data}}$.[2] An entire dataset can be seen as a matrix $\mathbf{X} \in \mathbb{R}^{n \times d}$ containing n examples, one example in each row.

[2]We use the subscript 'data' to make the difference to the model distribution p_{model} and the true distribution p clear. We will drop the subscript in later sections.

In *supervised learning*, for every input $\mathbf{x}_i$, the output is typically a separate label y_i, which can be arranged as a vector of labels $\mathbf{y}$ for the entire dataset. In *unsupervised learning*, no designated labels are available. Two common categories of machine learning tasks are *classification* and *regression*: In classification, the label y_i belongs to one of a predefined number of *classes* or *categories*. In regression, y_i is a continuous number.

Classification further subsumes *binary classification*, *multi-class classification*, and *multi-label classification*. Binary classification only deals with two classes, while multi-class classification deals with more than two classes. Typically, every example $\mathbf{x}_i$ only has one correct label y_i. In multi-label classification, every $\mathbf{x}_i$ may be associated with multiple labels.

In many scenarios throughout this thesis, the output may be more than a single number. Tasks with more complex outputs, known as *structured prediction*, are common in natural language processing and will be discussed in Section 2.4.

2.2.1 Maximum likelihood estimation

The most common way to design a machine learning algorithm is to use the principle of *maximum likelihood estimation* (MLE). An MLE model is defined as a function $p_{\text{model}}(\mathbf{x};\theta)$ that maps an input $\mathbf{x}$ to a probability using a set of *parameters* θ. As the true probability $p(\mathbf{x})$ of an example $\mathbf{x}$ is unknown, we approximate the true probability $p(\mathbf{x})$ with the probability $\hat{p}_{\text{data}}(\mathbf{x})$ under the empirical or data generating distribution.

The objective of MLE then is to bring the probability of our model $p_{\text{model}}(\mathbf{x};\theta)$ as close as possible to the empirical probability of the input $\hat{p}_{\text{data}}(\mathbf{x})$. In other words, MLE seeks to maximize the *likelihood* or probability of the data under the configuration of the model. The maximum likelihood estimator is defined as

$$\begin{aligned}\hat{\theta}_{\text{MLE}} &= \arg\max_{\theta} p_{\text{model}}(\mathbf{X};\theta) \\ &= \arg\max_{\theta} \prod_{i=1}^{n} p_{\text{model}}(\mathbf{x}_i;\theta)\end{aligned}$$

In practice, many of the probabilities in the product can be small, leading to underflow. Taking the logarithm does not change the arg max, but transforms the product into a sum, which results in a more convenient optimization problem [Goodfellow et al., 2016]:

$$\hat{\theta}_{\text{MLE}} = \arg\max_{\theta} \sum_{i=1}^{n} \log p_{\text{model}}(\mathbf{x}_i;\theta). \tag{2.29}$$

As the arg max also does not change under division by a constant value, we can divide by n to obtain an expectation with respect to the empirical distribution of the data $\hat{p}_{\text{data}}$ (see Equation 2.19):

$$\hat{\theta}_{\text{MLE}} = \arg\max_{\theta} \mathbb{E}_{\mathbf{x}\sim\hat{p}_{\text{data}}}[\log p_{\text{model}}(\mathbf{x}_i;\theta)]. \tag{2.30}$$

Rather than maximizing the likelihood of the data under the model, MLE can also be seen as minimizing the dissimilarity between the empirical distribution $\hat{p}_{\text{data}}$ and the model distribution p_{model} as measured by the KL divergence:

$$D_{\text{KL}}(\hat{p}_{\text{data}}||p_{\text{model}}) = \mathbb{E}_{\mathbf{x}\sim\hat{p}_{\text{data}}}[\log \hat{p}_{\text{data}}(\mathbf{x}) - \log p_{\text{model}}(\mathbf{x}_i;\theta)]. \tag{2.31}$$

As the term on the left, $\log \hat{p}_{\text{data}}(\mathbf{x})$ is only a function of the data generating distribution and not the model, we can train the model to minimize the KL divergence by only minimizing the term on the right-hand side, $-\log p_{\text{model}}(\mathbf{x}_i;\theta)$. Minimizing a negative term is the same as maximizing the term, so this objective is the same as the MLE objective in Equation 2.30:

$$\hat{\theta}_{\text{MLE}} = \arg\min_{\theta} -\mathbb{E}_{\mathbf{x}\sim\hat{p}_{\text{data}}}[\log p_{\text{model}}(\mathbf{x}_i;\theta)]. \tag{2.32}$$

Furthermore, this objective is also the same as minimizing the cross-entropy defined in Equation 2.25 between the empirical distribution $\hat{p}_{\text{data}}$ and the model distribution p_{model}:

$$\hat{\theta}_{\text{MLE}} = \arg\min_{\theta} H(\hat{p}_{\text{data}}, p_{\text{model}}). \tag{2.33}$$

Cross-entropy is a common loss term in machine learning and the objective function that is most commonly used in neural networks. Consequently, we will make frequent use of it throughout this thesis.

Conditional maximum likelihood The MLE estimator $p_{\text{model}}(\mathbf{x};\theta)$ discussed thus far essentially does unsupervised learning as it only seeks to estimate the likelihood of the data. For supervised learning, we instead need to estimate the conditional probability $P(y \mid \mathbf{x};\theta)$ in order to predict the label y given $\mathbf{x}$. The conditional maximum likelihood estimator is:

$$\hat{\theta}_{\text{MLE}} = \arg\max_{\theta} P(\mathbf{y} \mid \mathbf{X};\theta)$$

This can again be decomposed into:

$$\hat{\theta}_{\text{MLE}} = \arg\max_{\theta} \sum_{i=1}^{n} \log P(y \mid \mathbf{x}_i;\theta). \tag{2.34}$$

Point estimation The conditional maximum likelihood estimator is a *point estimator*: It provides the single 'best' prediction $\hat{y}$ for the true label y. A point estimator $\hat{\theta}$ is any function of the data that seeks to model the true underlying parameter θ^* of the data:

$$\hat{\theta} = g(\mathbf{X}) \tag{2.35}$$

As the data is assumed to be generated from a random process and $\hat{\theta}$ is a function of the data, $\hat{\theta}$ is itself a random variable.

2.2.2 Linear regression

The simplest example of a point estimator that maps from inputs to outputs is *linear regression*, which aims to solve a regression problem. Linear regression models a conditional probability distribution $p(y \mid \mathbf{x})$: it takes as input a vector $\mathbf{x} \in \mathbb{R}^d$ and aims to predict the value of a scalar $y \in \mathbb{R}$ using a vector $\theta \in \mathbb{R}^d$ of *weights* or *parameters* and an *intercept* or *bias* term $b \in \mathbb{R}$:

$$\hat{y}(\mathbf{x}; \theta) = \theta^\top \mathbf{x} + b \tag{2.36}$$

where $\hat{y}$ is the predicted value of y. The mapping from features to prediction is an *affine function*, i.e. a *linear function* plus a constant.

Mean squared error In order to learn the weights θ, we can minimize the model's *error*, a task-specific measure of how far the model's prediction $\hat{y}$ differs from the true value y. A common error measure is *mean squared error*, which is defined as:

$$\text{MSE} = \frac{1}{n} \sum_{i=1}^{n} (\hat{y}_i - y_i)^2 \tag{2.37}$$

Other commonly used terms for such an error measure are *objective function*, *cost function*, and *loss*. We can view mean squared error also as maximum likelihood estimation, specifically as the cross-entropy between the empirical distribution and a Gaussian model. Let the conditional distribution $p(y \mid \mathbf{x})$ modelled by linear regression be parameterized by a Gaussian. The conditional maximum likelihood estimator as defined in Equation 2.34 for linear regression is then:

$$\begin{aligned} \hat{\theta}_{\text{MLE}} &= \arg\max_{\theta} \sum_{i=1}^{n} \log p(y \mid \mathbf{x}_i; \theta) \\ &= \arg\max_{\theta} \sum_{i=1}^{n} \log \mathcal{N}(y; \hat{y}(\mathbf{x}; \theta), \sigma^2) \end{aligned}$$

where $\hat{y}(\mathbf{x};\theta)$ predicts the mean of the Gaussian and σ^2 is a constant. Substituting the definition of the Gaussian distribution from Equation 2.18, we obtain:

$$\hat{\theta}_{\text{MLE}} = \arg\max_{\theta} \sum_{i=1}^{n} \log \left[\sqrt{\frac{1}{2\pi\sigma^2}} \exp(-\frac{1}{2\sigma^2}(\hat{y}_i - y_i)^2) \right]$$

Taking the logarithm of a product and as $\log e^b = b$, we get:

$$\hat{\theta}_{\text{MLE}} = \arg\max_{\theta} \sum_{i=1}^{n} \frac{1}{2} \log \left(\frac{1}{2\pi\sigma^2} \right) - \frac{1}{2\sigma^2}(\hat{y}_i - y_i)^2$$

Applying the linearity of summation yields:

$$\hat{\theta}_{\text{MLE}} = \arg\max_{\theta} \frac{n}{2} \log \left(\frac{1}{2\pi\sigma^2} \right) - \frac{1}{2\sigma^2} \sum_{i=1}^{n} (\hat{y}_i - y_i)^2$$

The right-most term is just the mean squared error. Substituting the definition of MSE from Equation 2.37, we obtain:

$$\hat{\theta}_{\text{MLE}} = \arg\max_{\theta} \frac{n}{2} \log \left(\frac{1}{2\pi\sigma^2} \right) - \frac{n}{2\sigma^2} \text{MSE}$$

As n, π, and σ^2 are constants, MLE requires only to maximize the negative MSE, which is the same as minimizing the MSE.

Linear regression with mean squared error is also known as *linear least squares*. A common way to find a solution is to view the problem as a matrix equation (omitting the bias term):

$$\mathbf{X}\theta = \mathbf{y} \tag{2.38}$$

The *normal equation* then minimizes the sum of the squared differences between the left and the right side and yields the desired parameters θ:

$$\begin{aligned} \mathbf{X}^\top \mathbf{X} \hat{\theta} &= \mathbf{X}^\top \mathbf{y} \\ \hat{\theta} &= (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{y} \end{aligned} \tag{2.39}$$

$\mathbf{X}^\top \mathbf{X}$ is also known as *normal matrix*.

Logistic regression We can also apply linear regression to classification. In the case of binary classification, we have two classes, class 0 and class 1. We can transform the output of linear regression into a probability by 'squashing' it to be in the interval $(0, 1)$ using the *sigmoid* or *logistic function* σ, which is defined as:

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{2.40}$$

The probability produced by logistic regression is then calculated as follows:

$$\hat{p}(y = 1 \mid \mathbf{x}; \theta) = \hat{y} = \sigma(\theta^\top \mathbf{x}). \tag{2.41}$$

Specifying the probability of one of these classes determines the probability of the other class, as the output random variable follows a Bernoulli distribution.

For multi-class classification, we learn a separate set of weights $\theta_i \in \theta$ for the label y_i of the i-th class. We then use the *softmax* function to squash the values to obtain a categorical distribution:

$$\hat{p}(y_i|\mathbf{x}; \theta) = \frac{e^{\theta_i^\top \mathbf{x}}}{\sum_{j=1}^{C} e^{\theta_j^\top \mathbf{x}}} \tag{2.42}$$

where the denominator is the so-called *partition function* that normalizes the distribution by summing over the scores for all C classes.

We then calculate the cross-entropy between the empirical conditional probability $p(y \mid \mathbf{x})$ and the probability of our model $\hat{p}(y \mid \mathbf{x}; \theta)$ for each example $\mathbf{x}$:

$$H(p, \hat{p}; \mathbf{x}) = -\sum_{i=1}^{C} p(y_i \mid \mathbf{x}) \log \hat{p}(y_i \mid \mathbf{x}; \theta) \tag{2.43}$$

For binary classification, this simplifies to:

$$H(p, \hat{p}; \mathbf{x}) = -(1 - y) \log(1 - \hat{y}) - y \log \hat{y} \tag{2.44}$$

As our cost function $J(\theta)$, we minimize the average cross-entropy over all examples in our data:

$$J(\theta) = \frac{1}{n} \sum_{i=1}^{n} H(p, \hat{p}; \mathbf{x}_i) \tag{2.45}$$

Other loss functions There are many different loss functions, which are useful for particular applications. We will mention two loss functions here that will be used later. The *hinge loss* or *max-margin loss* is defined as follows for multi-class classification:

$$\mathcal{L}_h = \max(0, \delta - f(\mathbf{x}_p) + f(\mathbf{x}_n)) \tag{2.46}$$

where δ is a positive constant, usually 1, $f(\cdot)$ is a model, and $\mathbf{x}_p$ and $\mathbf{x}_n$ are positive and negative examples respectively. The hinge loss encourages the model to assign a score for positive examples that is higher than the score for negative examples by the margin δ. This loss is used in *support vector machines* (SVM) and also for learning monolingual and cross-lingual word embeddings (§3.3.2.3, §3.5.4).

The *Huber loss* is a piecewise loss function that is less sensitive to outliers than squared error:

$$\mathcal{L}_\delta = \begin{cases} \frac{1}{2}(y - \hat{y})^2 & \text{for } |y - \hat{y}| < \delta \\ \delta(|y - \hat{y}| - \frac{1}{2}\delta), & \text{otherwise.} \end{cases} \tag{2.47}$$

It is quadratic for small differences between the label y and the prediction $\hat{y}$ and linear for large values. It is used to compute the proxy $\mathcal{A}$ distance in domain adaptation (§3.4.2.1).

In contrast to linear regression with mean squared error, there is typically no closed-form solution to obtain the optimal weights for most loss functions. Instead, we iteratively minimize the error of our model using an algorithm known as *gradient descent*.

2.2.3 Gradient descent

Gradient descent is an efficient method to minimize an objective function $J(\theta)$. It updates the model's parameters $\theta \in \mathbb{R}^d$ in the opposite direction of the *gradient* $\nabla_\theta J(\theta)$ of the function. The gradient is the vector containing all the *partial derivatives* $\frac{\partial}{\partial \theta_i} J(\theta)$. The i-th element of the gradient is the partial derivative of $J(\theta)$ with respect to θ_i.

Gradient descent then updates the parameters:

$$\theta = \theta - \eta \cdot \nabla_\theta J(\theta) \tag{2.48}$$

where η is the *learning rate* that determines the magnitude of an update to our parameters. In practice, η is one of the most important settings when training a model. To guarantee convergence of the algorithm, the learning rate is often reduced or *annealed* over the course of training. In Section 7.1, we propose a novel *schedule* for learning rate annealing in transfer learning.

As we have seen previously, we typically minimize the expected value or average of an error function over the empirical distribution of our data:

$$J(\theta) = \mathbb{E}_{\mathbf{x},y\sim p_{\text{data}}} \mathcal{L}(\mathbf{x}, y, \hat{y}\theta) = \frac{1}{n} \sum_{i=1}^{n} \mathcal{L}(\mathbf{x}, y, \hat{y}, \theta). \tag{2.49}$$

The gradient $\nabla_\theta J(\theta)$is thus:

$$\nabla_\theta J(\theta) = \frac{1}{n} \sum_{i=1}^{n} \nabla_\theta \mathcal{L}(\mathbf{x}_i, y_i, \hat{y}_i, \theta) \tag{2.50}$$

This is known as *batch gradient descent* and is expensive as for each update the gradient needs to be computed for all examples in the data. Alternatively, *stochastic gradient descent* iterates trough the data, computes the gradient, and performs an update for each example i:

$$\nabla_\theta J(\theta) \approx \nabla_\theta \mathcal{L}(\mathbf{x}_i, y_i, \hat{y}_i, \theta) \tag{2.51}$$

While this is cheaper, the resulting gradient estimate is a lot more noisy. The most common approach is to choose a middle ground and compute the gradient over a *mini-batch* of m examples, which is commonly known as *mini-batch gradient descent* or stochastic gradient descent with mini-batches:

$$\nabla_\theta J(\theta) = \frac{1}{m} \sum_{i=1}^{m} \nabla_\theta L(\mathbf{x}_i, y_i, \hat{y}_i, \theta) \tag{2.52}$$

The mini-batch size m typically ranges from 2 to a few hundred and enables training of large models on datasets with hundreds of thousands or millions of examples. In practice, mini-batch gradient descent is the default setting and is often referred to as stochastic gradient descent as well.

While stochastic gradient descent works surprisingly well in practice and is the main way to train neural networks, it has a few weaknesses: It does not remember its previous steps and uses the same learning rate for all its parameters. We direct the reader to [Ruder et al., 2016b] for an overview of momentum-based and adaptive learning rate techniques that seek to ameliorate these deficiencies.

2.2.4 Generalization

The goal of machine learning is *generalization*, training a model that performs well on new and previously unseen inputs. To this end, the available data $\mathbf{X}$ is typically split into a part that is used for training, the *training set* and a second part reserved for evaluating the model, the *test set*. Performance on the test set is then used as a proxy for the model's ability to generalize to new inputs.

This measure is responsible for the main tension in machine learning: During training, we compute the *training error*, the error of the model on the training set, which we try to minimize. The actual measure of interest, however, is the *generalization error* or *test error*, the model's performance on the test set, which it has never seen before. This is also the main difference to optimization: While optimization seeks to find the minimum that *minimizes the training error*, machine learning aims to *minimize generalization error*.

Train and test sets are typically assumed to be *i.i.d.*: Examples in each dataset are *independent* from each other and train and tests sets are *identically distributed*, i.e. drawn from the same probability distribution.

Bias-variance trade-off The goal to minimize a model's generalization error gives rise to two desiderata:

1. to minimize the training error;
2. and to minimize the gap between training and test error.

This dichotomy is also known as *bias-variance trade-off*. If the model is not able to obtain a low error on the training set, it is said to have high *bias*. This is typically the result of erroneous assumptions in the learning algorithm that cause it to miss relevant relations in the data. On the other hand, if the gap between the training error and test error is too large, the model has high *variance*. It is sensitive to small fluctuations and models random *noise* in the training data rather than the true underlying distribution.

More formally, the bias of an estimator $\hat{\theta}$ is the expected difference between the value of the parameter $\hat{\theta}$ and the true underlying value of the parameter θ^* with regard to the data generating distribution:

$$\text{Bias}(\hat{\theta}) = \mathbb{E}[\hat{\theta} - \theta^*] \tag{2.53}$$

The estimator $\hat{\theta}$ is *unbiased* if $\text{bias}(\hat{\theta}) = \mathbf{0}$. For instance, the sample mean is an unbiased estimator of the mean of a distribution.

The *variance* of an estimator is simply its variance:

$$\text{Var}(\hat{\theta}) = \mathbb{E}[\hat{\theta}^2] - \mathbb{E}[\hat{\theta}]^2. \tag{2.54}$$

By rearranging, we get:

$$\mathbb{E}[\hat{\theta}^2] = \text{Var}(\hat{\theta}) + \mathbb{E}[\hat{\theta}]^2. \tag{2.55}$$

The square root of the variance of an estimator is called the standard error $\text{SE}(\hat{\theta})$.

To measure an estimator's performance, we can compare the mean squared error of the estimator $\hat{\theta}$ to the true parameter value θ^*:

$$\text{MSE} = \mathbb{E}[(\hat{\theta} - \theta^*)^2]$$

Expanding the binomial, we obtain:

$$\text{MSE} = \mathbb{E}[\hat{\theta}^2] - \mathbb{E}[2\hat{\theta}\theta^*] + \mathbb{E}[{\theta^*}^2]$$

We now replace $\mathbb{E}[\hat{\theta}^2]$ and $\mathbb{E}[\theta^{*2}]$ with the right-hand side of Equation 2.55 respectively:

$$\text{MSE} = \text{Var}(\hat{\theta}) + \mathbb{E}[\hat{\theta}]^2 - \mathbb{E}[2\hat{\theta}\theta^*] + \text{Var}(\theta^*) + \mathbb{E}[\theta^*]^2$$

We can now form another binomial expansion, reduce it to a binomial, and using the linearity of expectation, we get:

$$\begin{aligned}\text{MSE} &= \text{Var}(\hat{\theta}) + \text{Var}(\theta^*) + (\mathbb{E}[\hat{\theta}]^2 - \mathbb{E}[2\hat{\theta}\theta^*] + \mathbb{E}[\theta^*]^2) \\ &= \text{Var}(\hat{\theta}) + \text{Var}(\theta^*) + (\mathbb{E}[\hat{\theta}] - \mathbb{E}[\theta^*])^2 \\ &= \text{Var}(\hat{\theta}) + \text{Var}(\theta^*) + \mathbb{E}[\hat{\theta} - \theta^*]^2\end{aligned}$$

$\text{Var}(\theta)$ is the true variance σ^2 of the parameter θ and the right-most term under the square is the definition of the bias in Equation 2.53. Replacing both yields the *bias-variance decomposition* for squared error:

$$\text{MSE} = \text{Var}(\hat{\theta}) + \sigma^2 + \text{Bias}(\hat{\theta})^2 \tag{2.56}$$

This decomposition sheds more light on the trade-off between bias and variance in machine learning. The expected error of a model trained with mean squared error is thus lower bounded by the sum of three terms:

- The square of the bias of the method, i.e. the error caused by the simplifying assumptions inherent in the model.
- The variance of the method, i.e. how much its results vary across the mean.
- The variance of the true underlying distribution.

If a model has high bias it is also said to be *underfitting*. If a model has high variance, it is known to be *overfitting*. A key factor that determines whether a model underfits or overfits is its *capacity*, which is its ability to fit a variety of functions. One way to control a model's capacity is to choose an appropriate *hypothesis space*, the set of functions it can choose from to find the solution. The hypothesis space of linear regression is the set of all linear functions of its input. We can increase the capacity of linear regression by generalizing it to include polynomials of degree k:

$$\hat{y} = b + \sum_{i=1}^{k} \theta_i^\top \mathbf{x}^i \tag{2.57}$$

where $\theta_i \in \mathbb{R}^d$ are additional weight vectors for each polynomial. A machine learning model performs best when its capacity is appropriate for the task it is required to solve. A commonly used heuristic is expressed by *Occam's razor*, which states that among competing hypotheses that explain known observations equally well, one should choose the

"simplest", which in this context refers to the model with the lowest capacity. However, while simpler functions are more likely to generalize, we still require a hypothesis that is sufficiently complex to achieve low training error.

In machine learning, the *no free lunch theorem* [Wolpert and Macready, 1997] states that no algorithm is universally better than any other. Specifically, averaged over *all* possible data generating distributions, every classification algorithm achieves the same error when classifying previously unknown points. Our goal in practice is thus to bias the algorithm towards distributions or relations in the data that we are more likely to encounter in the real world and to design algorithms that perform well on particular tasks. Throughout this thesis, we will use bias and *inductive bias* interchangeably to describe assumptions that are encoded in a model about unseen data. We generally aim to develop models with an inductive bias that is useful to generalize to novel domains, tasks, and languages.

Statistical learning theory provides theoretical bounds on the generalization error: In particular, the difference between training error and generalisation error has been shown to grow with the capacity of the model but shrink as the number of training examples increases [Vapnik and Chervonenkis, 2015]. These bounds, however, are rarely used in practice as they are quite loose and it is difficult to determine the capacity of deep neural networks [Goodfellow et al., 2016]. Nevertheless, the generalisation behaviour of deep neural networks is an active area of research.

In practice, a validation set is often used in addition to tune different settings of the model, its *hyper-parameters*, such as the degree of the polynomial in logistic regression. If the test set is too small, another technique called *cross-validation* is typically used. Cross-validation repeats the training and test computations on different randomly chosen splits of the data and averages the test error over these splits. The most common variation is k-fold cross-validation, which splits the data into k subsets of equal size and repeats training and evaluation k times, using $k-1$ splits for training and the remaining one for testing.

2.2.5 Regularization

Another way to modify a model's capacity is to encourage the model to prefer certain functions in its hypothesis space over others. The most common way to achieve this is by adding a *regularization* term $\Sigma(\theta)$ to the cost function $J(\theta)$:

$$J(\theta) = \text{MSE} + \lambda\Sigma(\theta) \tag{2.58}$$

where λ controls the strength of the regularization. If $\lambda = 0$, we impose no restriction. As λ grows larger, the preference that we impose on the algorithm becomes more prominent.

The most popular forms of regularization leverage common *vector norms*. ℓ_1 regularization places a penalty on the ℓ_1 norm, i.e. the sum of the absolute values of the weights and is defined as follows:

$$\Sigma(\theta) = \|\theta\|_1 = \sum_i |\theta_i| \tag{2.59}$$

where $\theta_i \in \mathbb{R}$. ℓ_1 regularization is also known as *lasso* (**l**east **a**bsolute **s**hrinkage and **s**election **o**perator) and is the most common way to induce sparsity in a solution as the ℓ_1 norm will encourage most weights to become 0.

ℓ_2 regularization is defined as:

$$\Sigma(\theta) = \|\theta\|_2^2 \tag{2.60}$$

where $\|\theta\|_2 = \sqrt{\sum_i \theta_i^2}$ is the *Euclidean norm* or ℓ_2 norm. Somewhat counter-intuitively, ℓ_2 regularization thus seeks to minimize the *squared* ℓ_2 norm as in practice, the squared ℓ_2 norm is often more computationally convenient to work with than the ℓ_2 norm. For instance, derivatives of the squared ℓ_2 norm with respect to each element of θ depend only on the corresponding element, while derivatives of the ℓ_2 norm depend on the entire vector [Goodfellow et al., 2016].

ℓ_2 regularization is also known as *Tikhonov regularization*, *ridge regression*, and *weight decay*[3]. ℓ_2 regularization expresses a preference for smaller weights in a model.

Different forms of regularization may also be combined. The combination of ℓ_1 and ℓ_2 regularization is also known as *elastic net regularization*. It uses an α parameter to balance the contributions of both regularizers:

$$\Sigma(\theta) = \alpha\|\theta\|_1 + (1-\alpha)\|\theta\|_2^2. \tag{2.61}$$

Besides the ℓ_1 and ℓ_2 norms, the only other norm that is used occasionally for regularization is the ℓ_∞ norm or *max norm*, which penalizes only the maximum parameter value :

$$\|\theta\|_\infty = \max_i |\theta_i| \tag{2.62}$$

In some scenarios, we are interested in imposing a norm on a weight matrix $\mathbf{W} \in \mathbb{R}^{m \times n}$. For this case, we use the matrix counterpart of the ℓ_2 norm, the *Frobenius norm*:

$$\|\mathbf{W}\|_F = \sqrt{\sum_{i,j} \mathbf{W}_{i,j}{}^2} \tag{2.63}$$

[3]Note that ℓ_2 regularization and weight decay are not equivalent under all conditions [Loshchilov and Hutter, 2017a]

The Frobenius norm is useful for instance to express the preference that two weight matrices $\mathbf{W}_1$ and $\mathbf{W}_2$ should be orthogonal, i.e. $\mathbf{W}_1^\top \mathbf{W}_2 = I$. This is achieved by placing the squared Frobenius norm on the matrix product:

$$\Sigma(\mathbf{W}_1, \mathbf{W}_2) = \|\mathbf{W}_1^\top \mathbf{W}_2\|_F^2. \tag{2.64}$$

This orthogonality constraint is a common component of current approaches to domain adaptation (§3.4.2.2), which we use to encourage non-redundancy of the representations of different layers (§4.2, 6.1).

Another common matrix norm is the *nuclear norm* or *trace norm*, which applies the ℓ_1 norm to the vector of singular values σ_i of matrix $\mathbf{W}$:

$$\|\mathbf{W}\|_* = \sum_{i=1}^{\min\{m,n\}} \sigma_i(\mathbf{W}) \tag{2.65}$$

The trace norm is the tightest convex relaxation of the rank of a matrix [Recht et al., 2010], so can be useful to encourage a matrix to be low-rank. It has been frequently used in multi-task learning (§3.2.5.1).

While we have focused on vector and matrix norms in this section, any approach that implicitly or explicitly expresses a preference for particular solutions can be seen as regularization. Throughout this thesis, defining different ways to express such preferences via an appropriate inductive bias is one of the key themes (§1.2).

2.3 Neural networks

Neural networks have become the tool of choice in natural language processing in recent years. In this section, we will give an overview of the fundamental building blocks used in neural networks. Neural networks can be seen as compositions of functions. In fact, we can view the basic machine learning models described so far, linear regression and logistic regression, as simple instances of a neural network.

Recall that multi-class logistic regression consists of the following functions:

$$\begin{aligned} f(\mathbf{x}) &= \mathbf{W}\mathbf{x} + \mathbf{b} \\ g(\mathbf{y}) &= \mathsf{softmax}(\mathbf{y}) \end{aligned} \tag{2.66}$$

where $\mathbf{W} \in \mathbb{R}^{C \times d}$, $\mathbf{x} \in \mathbb{R}^d$, $\mathbf{b} \in \mathbb{R}^C$, $\mathbf{y} \in \mathbb{R}^C$, C is the number of classes and d is the dimensionality of the input. In the following, we will use $\mathbf{W}$ to designate a matrix of weights, while $\theta \ni \mathbf{W}, \mathbf{b}$ is the set of parameters of the model. Logistic regression can be

seen as a composition of the functions f and g: $g(f(\mathbf{x}))$ where $f(\cdot)$ is an affine function and $g(\cdot)$ is an *activation* function, in this case the softmax function.

A neural network is a composition of multiple such affine functions interleaved with non-linear activation functions. The softmax and sigmoid functions are common functions used at the final or *output layer* of a neural network to obtain a categorical and Bernoulli distribution respectively. Non-output layers are referred to as *hidden layers*. Linear regression can be seen as a neural network without a hidden layer and a linear activation function—the identity function—while logistic regression employs a non-linear activation function. Neural networks are typically named according to the number of hidden layers. A model with one hidden layer is known as a one-layer *feed-forward neural network*, which is also known as a *multilayer perceptron* (MLP):

$$\begin{aligned} \mathbf{h} &= \sigma_1(\mathbf{W}_1\mathbf{x} + \mathbf{b}_1) \\ \mathbf{y} &= \mathsf{softmax}(\mathbf{W}_2\mathbf{h} + \mathbf{b}_2) \end{aligned} \tag{2.67}$$

where σ_1 is the activation function of the first hidden layer. Note that each layer is parameterized with its own weight matrix $\mathbf{W}$ and bias vector $\mathbf{b}$. Layers typically have separate parameters, but different layers can also set their parameters to be the same, which is referred to as *tying* or *sharing* of such parameters. Such parameter sharing induces an inductive bias that can often help with generalization. We will see many instances of parameter sharing throughout this thesis, such as in multi-task learning (§3.2, §6).

Computing the output of one layer, e.g. $\mathbf{h}$ that is fed as input to subsequent layers, which eventually produce the output of the entire network $\mathbf{y}$ is known as *forward propagation*. As a composition of linear functions can be expressed as another linear function, the expressiveness of deep neural networks mainly comes from its non-linear activation functions.

Activation functions Besides the sigmoid (Equation 2.40) and softmax (Equation 2.42) functions that are mainly used at output layers, a common activation function for hidden layers is the *rectified linear unit* (ReLU), which is defined as:

$$\sigma(\mathbf{x}) = \max(0, \mathbf{x}). \tag{2.68}$$

Another activation function that is less often used in practice is the *hyperbolic tangent* or *tanh* function, which outputs values in the range $(-1, 1)$:

$$\sigma(\mathbf{x}) = \frac{e^{\mathbf{x}} - e^{-\mathbf{x}}}{e^{\mathbf{x}} + e^{-\mathbf{x}}}. \tag{2.69}$$

2.3.1 Layers and models

We will now detail the layers and models that are commonly applied to NLP tasks and that will be used throughout this thesis.

Recurrent neural network As text is sequential, we will be using models that can process a sequence of inputs. The most elementary neural network for sequential input is the *recurrent neural network* [RNN; Elman, 1990]. An RNN can also be seen as a feed-forward neural network with a dynamic number of hidden layers that are all set to have the same parameters. Rather than being "deep", the model is "wide" as it is unrolled through time. In contrast to a regular feed-forward neural network, however, it accepts a new input at every "layer" or time step. Specifically, the RNN maintains a hidden state $\mathbf{h}_t$, which represents its "memory" of the contents of the sequence at each time step t. At every time step, the RNN performs the following operation:

$$\begin{aligned} \mathbf{h}_t &= \sigma_h(\mathbf{W}_h\mathbf{x}_t + \mathbf{U}_h\mathbf{h}_{t-1} + \mathbf{b}_h) \\ \mathbf{y}_t &= \sigma_y(\mathbf{W}_y\mathbf{h}_t + \mathbf{b}_y) \end{aligned} \tag{2.70}$$

where σ_h and σ_y are activation functions. The RNN applies a transformation $\mathbf{U}_h$ to modify the previous hidden state $\mathbf{h}_{t-1}$ and a transformation $\mathbf{W}_h$ to the current input $\mathbf{x}_t$, which yields the new hidden state $\mathbf{h}_t$. At every time step, the RNN furthermore produces an output $\mathbf{y}_t$. In practice, the RNN has trouble learning over a large number of time steps (§2.3.2).

Long-short term memory *Long-short term memory* [LSTM; Hochreiter and Schmidhuber, 1997] networks are preferred compared to RNNs for dealing with sequential data as they can retain information for longer time spans, which is necessary for modelling long-term dependencies common in natural language. The LSTM can be seen as a more sophisticated RNN cell that introduces mechanisms to decide what should be "remembered" and "forgotten". The LSTM augments the RNN with a forget gate $\mathbf{f}_t$, an input gate $\mathbf{i}_t$, and an output gate $\mathbf{o}_t$, which are all functions of the current input $\mathbf{x}_t$ and the previous hidden state $\mathbf{h}_t$. These gates interact with the previous cell state $\mathbf{c}_{t-1}$, the current input, and the current cell state $\mathbf{c}_t$ and enable the model to selectively retain or

overwrite information. The entire model is defined as follows[4]:

$$\begin{aligned}
\mathbf{f}_t &= \sigma_g(\mathbf{W}_f\mathbf{x}_t + \mathbf{U}_f\mathbf{h}_{t-1} + \mathbf{b}_f) \\
\mathbf{i}_t &= \sigma_g(\mathbf{W}_i\mathbf{x}_t + \mathbf{U}_i\mathbf{h}_{t-1} + \mathbf{b}_i) \\
\mathbf{o}_t &= \sigma_g(\mathbf{W}_o\mathbf{x}_t + \mathbf{U}_o\mathbf{h}_{t-1} + \mathbf{b}_o) \\
\mathbf{c}_t &= \mathbf{f}_t \circ \mathbf{c}_{t-1} + \mathbf{i}_t \circ \sigma_c(\mathbf{W}_c\mathbf{x}_t + \mathbf{U}_c\mathbf{h}_{t-1} + \mathbf{b}_c) \\
\mathbf{h}_t &= \mathbf{o}_t \circ \sigma_h(\mathbf{c}_t)
\end{aligned} \tag{2.71}$$

where σ_g is the sigmoid activation function, σ_c and σ_h are the tanh activation function, and $\circ$ is element-wise multiplication, also known as *Hadamard product*.

LSTM cells can be stacked in multiple layers. In most cases, we will use a *bidirectional LSTM* [BiLSTM; Graves et al., 2013], which runs separate LSTMs forward and backward over the sequence. The hidden state $\mathbf{h}_t$ is the concatenation of the hidden states from the forward and backward LSTMs at time step t:

$$\mathbf{h}_t = [\mathbf{h}_{\text{fwd}}; \mathbf{h}_{\text{bwd}}] \tag{2.72}$$

Word embedding When applying neural networks to natural language tasks, each word w_i in the vocabulary V that occurs in the input text is typically mapped to a vector $\mathbf{x}_i$, which is known as the *word embedding* of w_i. The word embeddings are stored in a word embedding matrix $\mathbf{X} \in \mathbb{R}^{|V| \times d}$. An input sequence of words $w_1, \ldots, w_T$ is thus commonly represented as a sequence of the corresponding word embeddings $\mathbf{x}_1, \ldots, \mathbf{x}_T$, which is provided as input to a neural network.

Convolutional neural network Another commonly used neural network is the *convolutional neural network* [CNN; LeCun et al., 1998]. We will describe here its application to natural language tasks [Kalchbrenner et al., 2014, Kim, 2014].

The convolutional layer slides filters of different window sizes over the concatenation of input word embeddings $[\mathbf{x}_1, \ldots, \mathbf{x}_T] \in \mathbb{R}^{T \times d}$ where $[\cdot, \cdot]$ designates concatenation and d is the dimensionality of the word embeddings. Each filter with weights $\mathbf{W} \in \mathbb{R}^{kd}$ generates a new feature $c_i \in \mathbb{R}$ for each window of k words $\mathbf{x}_{i:i+k-1} \in \mathbb{R}^{kd}$ according to the following operation[5]:

$$c_i = \sigma(\mathbf{w} \cdot \mathbf{x}_{i:i+k-1} + b) \tag{2.73}$$

[4]The original LSTM did not include a forget gate, which was introduced by Gers et al. [1999].

[5]For brevity, we show both the filter and the windows as 'flattened' vectors as in [Kim, 2014]. If seen as matrices, each filter would involve an elementwise multiplication and a sum.

where $b \in \mathbb{R}$ is the bias term and σ is an activation function, most commonly ReLU. Sliding the filter over each window of k words yields a feature map $\mathbf{c} \in \mathbb{R}^{T-k+1}$:

$$\mathbf{c} = [c_1, \ldots, c_{T-k+1}] \tag{2.74}$$

Each entry in the feature map is thus the result of performing a calculation that considers only a small segment of the input sequence and that shares parameters with the calculations to its left and right (by applying the same filter). *Max-pooling* is typically applied to condense a feature map to its most important feature:

$$\tilde{c} = \max(\mathbf{c}) \tag{2.75}$$

The maximum values of the feature maps produced by all filters are concatenated to a vector $\hat{\mathbf{c}} \in \mathbb{R}^C$ where C is the number of filters. This vector is then fed to the next layer or an output layer. The CNN is an example of how parameter sharing can be used to incorporate an inductive bias into a model: As each filter is applied to multiple windows, the model learns to capture local features that are invariant to the location in the input.

Autoencoder An autoencoder is a neural network that is trained to reconstruct its input by compression it first into a low-dimensional representation. In the simplest case with one hidden layer, it uses an *encoder* to map the input $\mathbf{x}$ to a vector $\mathbf{z}$:

$$\mathbf{z} = \sigma(\mathbf{W}\mathbf{x} + \mathbf{b}) \tag{2.76}$$

where σ is an activation function. A *decoder* then maps the *latent representation* back to a reconstructed version of the original input $\mathbf{x}'$:

$$\mathbf{x}' = \sigma'(\mathbf{W}'\mathbf{z} + \mathbf{b}') \tag{2.77}$$

The model is trained to minimize a reconstruction loss such as the squared error between the original and the reconstructed input:

$$\mathcal{L} = \|\mathbf{x} - \mathbf{x}'\|^2 \tag{2.78}$$

Autoencoders are commonly used to learn representations and have been used in sequential transfer learning (§3.3.2.3), domain adaptation (§3.4.2.2), and cross-lingual learning (§3.5.5).

2.3.2 Back-propagation

Analogous to many machine learning models where closed-form solutions are not available, neural networks are typically trained with stochastic gradient descent. As each model consists of multiple layers, calculating the gradient of the loss function with regard to the parameters $\nabla_\theta J(\theta)$ is non-trivial. To compute the gradient, we use a dynamic programming algorithm known as *back-propagation* [Rumelhart et al., 1986].

Back-propagation relies on the *chain rule of calculus*, which given functions $y = g(x)$ and $z = f(g(x)) = f(y)$ defines the derivative $\frac{dz}{dx}$ of z with respect to x as the derivative of z with respect to y times the derivative of y with respect to x:

$$\frac{dz}{dx} = \frac{dz}{dy}\frac{dy}{dx} \tag{2.79}$$

For vectors $\mathbf{x} \in \mathbb{R}^m$ and $\mathbf{y} \in \mathbb{R}^n$ and scalar z, we analogously obtain the partial derivative of z with respect to x_i as follows:

$$\frac{\partial z}{\partial x_i} = \sum_j \frac{\partial z}{\partial y_j}\frac{\partial y_j}{\partial x_i} \tag{2.80}$$

In vector notation, the gradient of z with respect to $\mathbf{x}$ containing all partial derivatives of z with respect to each x_i can be determined using matrix-vector multiplication:

$$\nabla_{\mathbf{x}} z = \left(\frac{\partial \mathbf{y}}{\partial \mathbf{x}}\right)^\top \nabla_{\mathbf{y}} z \tag{2.81}$$

where $\frac{\partial \mathbf{y}}{\partial \mathbf{x}} \in \mathbb{R}^{n \times m}$ is the *Jacobian matrix* of g, the matrix containing all partial derivatives. The back-propagation algorithm now performs such a Jacobian-gradient product for each operation in our neural network graph [Goodfellow et al., 2016].

Let us first define a deep feed-forward neural network with L layers, with weight matrices $\mathbf{W}_l$ and bias parameters $\mathbf{b}_l$ where $l \in \{1, \ldots, L\}$. The model takes an input $\mathbf{x}$, produces an output $\hat{y}$, and minimizes a cost function $J = \mathcal{L}(\hat{y}, y)$:

$$\mathbf{a}_l = \mathbf{b}_l + \mathbf{W}_l \mathbf{h}_{l-1}$$
$$\mathbf{h}_l = \sigma_l(\mathbf{a}_l)$$

where $\mathbf{h}_0 = \mathbf{x}$ and $\hat{\mathbf{y}} = \mathbf{h}_L$. Forward-propagation proceeds from the first layer, computing the representation $\mathbf{a}_l$ that is then fed through an activation function σ_l, which yields a hidden state $\mathbf{a}_l$, which is provided to the next layer. This is repeated until the output is $\hat{\mathbf{y}}$ and the loss J is calculated.

Back-propagation proceeds in backwards order: It starts by computing the gradient $\nabla_{\hat{y}} J$ of the loss function J with respect to the output $\hat{y}$. It then computes the gradient of the

representation $\mathbf{h}^{(l)}$ and $\mathbf{a}^{(l)}$ for the last layer, from which it then obtains the gradients on the parameters of the layer. It then continues until it arrives at the gradients of the first layer. The gradient $\nabla_{\hat{y}} J$ of the loss function with respect to the output is:

$$\nabla_{\hat{y}} J = \nabla_{\hat{y}} \mathcal{L}(\hat{y}, y) \tag{2.82}$$

If we use a squared error loss, then the gradient is simply:

$$\nabla_{\hat{y}} J = \nabla_{\hat{y}} \frac{1}{2}(\hat{y} - y)^2 = y - \hat{y} \tag{2.83}$$

We can now obtain the gradient of the loss function with regard to the layer's pre-activation representation $\nabla_{\mathbf{a}_l} J$ with the chain rule:

$$\nabla_{\mathbf{a}^{(l)}} J = \left(\frac{\partial \mathbf{h}^{(k)}}{\partial \mathbf{a}^{(k)}}\right)^\top \nabla_{\hat{y}} J = \sigma'(\mathbf{a}^{(l)}) \tag{2.84}$$

As our activation function is elementwise, we obtain:

$$\nabla_{\mathbf{a}_l} J = \sigma'(\mathbf{a}_l) \circ \nabla_{\hat{y}} J \tag{2.85}$$

This demonstrates why it is desirable for activation functions to be differentiable[6]. The sigmoid activation function, for instance, has a convenient derivative:

$$\sigma'(\mathbf{x}) = \sigma(\mathbf{x})(1 - \sigma(\mathbf{x})) \tag{2.86}$$

From this, we can now obtain the gradients of the model parameters based on each parameter's contribution:

$$\begin{aligned} \nabla_{\mathbf{b}_l} J &= \nabla_{\mathbf{a}_l} J \\ \nabla_{\mathbf{W}_l} J &= \nabla_{\mathbf{a}_l} J \; \mathbf{h}_{l-1}{}^\top \end{aligned} \tag{2.87}$$

We subsequently propagate the gradient to the next lower-level layer and repeat the calculations:

$$\nabla_{\mathbf{h}_{l-1}} J = \mathbf{W}_l{}^\top \nabla_{\mathbf{W}_l} J \tag{2.88}$$

As we can see, in order to compute the gradient on a parameter, the gradients on all computations that are the result of this parameter need to be known. For each layer, the computed gradients are then used to update the corresponding parameters with gradient descent.

[6]ReLU is mathematically differentiable everywhere except at 0. In practice, this does not cause problems as the exact value of 0 occurs rarely and a *right derivative* and a *left derivative* are still available, which can be used instead.

When optimizing RNNs the gradient needs to be propagated through the past time steps rather than through the depth of the model. Consequently, this is known as *back-propagation through time* [BPTT; Werbos, 1988]. As the error cannot be back-propagated indefinitely, it is generally propagated until a fixed length that is defined in advance. Longer lengths should enable the model to learn longer-range dependencies.

A common issue encountered when optimizing RNNs in practice is known as *exploding* or *vanishing gradients*. During forward propagation, the hidden state is multiplied many times with the weight matrix, once per time step. During back-propagation, this leads to the gradients being multiplied with the same values over and over. This causes the values to either explode, i.e. become very large or vanish, i.e. become very small, rendering the model unable to learn. Mitigating the exploding and vanishing gradient problem is one of the motivations for the development of the LSTM.

2.4 Natural language processing

Natural language processing (NLP) aims to teach computers to understand natural language. As the facility for language is abstract, we will take a more concrete view by defining NLP by way of its tasks, which generally map a text to linguistic structures that encode its meaning [Smith, 2011]. Examples of such structures can be seen in Table 2.1.

We will use the machine learning tools presented in the previous section to learn a mathematical model of this mapping. Specifically, we will aim to train a model that can map from an input x consisting of a sequence of words to an output y generally using the principle of maximum likelihood estimation (§2.2.1) to find a set of parameters θ that maximizes the conditional probability $P_\theta(y \mid x)$ of our model. We focus on *discriminative models* that model the conditional probability directly from raw data rather than using a *generative model* that learns the joint probability distribution $P(x, y)$.

Bag-of-words In some contexts (§4), we will represent a text as a *bag-of-words* (BOW) $\mathbf{x} \in \mathbb{R}^{|V|}$ where V is the vocabulary. Each entry x_i corresponds to the number of occurrences of the i-th word in the vocabulary in the text. This frequency may be weighted with *term frequency-inverse document frequency* (tf-idf), which additionally reflects how important a term is in a *corpus*, a collection of texts. An *n-gram* is a contiguous sequence of n words in a text. In the standard BOW model, we consider only *unigrams*, sequences of one word. We may additionally consider *bigrams*, sequences of two words. The bag-of-words ignores grammar and word order. To capture compositionality and dependencies in the input, throughout this thesis, our preferred way of representing

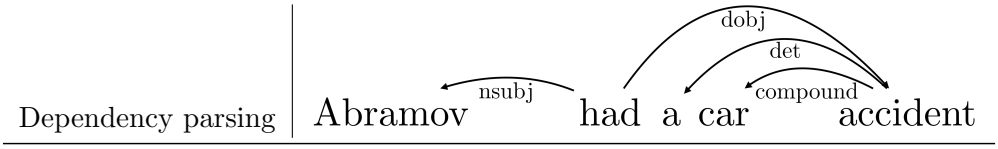
Bilingual dictionary induction	Abramov	hatte	ein	Auto	Unfall
Language modelling	had	a	car	accident	-
Sentiment analysis Topic classification	`Negative` `News`				
Dependency parsing					
NER	B-PERSON	O	O	O	O
SRL	B-ARG0	B-V	B-ARG1	I-ARG1	I-ARG1
Chunking	O	B-VP	B-NP	I-NP	I-NP
POS tagging	NNP	VBD	DT	NN	NN
WORDS	Abramov	had	a	car	accident

TABLE 2.1: Annotations for all tasks discussed in this thesis for an example sentence (bottom). Tasks are ordered roughly from low-level syntactic tasks (bottom) to high-level semantic tasks (top).

a sequence of words $w_1, \ldots, w_T$ will be to encode it as a sequence of the corresponding word embeddings $\mathbf{x}_1, \ldots, \mathbf{x}_T$.

Evaluation metrics NLP systems are typically evaluated with regard to their performance on the test set of the specific task. For binary classification, accuracy is the common evaluation measure, which is defined as follows:

$$Acc = \frac{TP + TN}{TP + TN + FP + FN} \tag{2.89}$$

where TP, TN, FP, and FN are the number of true positives, true negatives, false positives, and false negatives respectively. Intuitively, the number of true predictions is divided by the number of all predictions. For multi-class classification, the F_1 score is used, which is the harmonic mean of precision and recall:

$$F_1 = 2\,\frac{P \cdot R}{P + R}, \quad P = \frac{TP}{TP + FP}, \quad R = \frac{TP}{TP + FN}. \tag{2.90}$$

The F_1 metric balances precision, the fraction of correctly predicted instances and recall, the fraction of correctly predicted instances out of all instances of the category. It thus provides a good estimate of the overall quality of a model. Certain tasks may use specialized evaluation metrics, which will be introduced with the corresponding task.

We will now review the main NLP tasks that will be tackled in this thesis. We show

annotations for all tasks for an example sentence in Table 2.1.[7] POS tagging, chunking, named entity recognition (NER), and semantic role labelling (SRL) are *sequence labelling* tasks, which seek to assign each word w_i a corresponding label y_i.

Part-of-speech (POS) tagging POS tagging is the task of tagging a word in a text with its corresponding part-of-speech. A part-of-speech is a category of words with similar grammatical properties. Common English parts of speech are noun, verb, adjective, adverb, pronoun, preposition, conjunction, etc. Part-of-speech tagging is difficult as many words can have multiple parts of speech. Parts of speech can be arbitrarily fine-grained and are typically based on the chosen tag set. The most common tag set used by the Penn Treebank [Marcus et al., 1993] comprises 36 tags[8]. However, POS tags vary greatly between languages due to cross-lingual differences. The creation of a "universal" tagset has been an ongoing development: Petrov et al. [2012] proposed a tag set of 12 coarse-grained categories, while the current tag set of the Universal Dependencies 2.0 [Nivre et al., 2016] contains 17 tags[9]. When applying a POS tagger to a new domain, current models particularly struggle with word-tag combinations that have not been seen before (§4.2.3.4).

Chunking Chunking, also known as shallow parsing, aims to identify continuous spans of tokens that form syntactic units. Chunks are different from parts of speech as they typically represent higher order structures such as noun phrases or verb phrases. Approaches typically use BIO notation, which differentiates the beginning (B) and the inside (I) of chunks. O is used for tokens that are not part of a chunk. Both POS tagging and chunking act mostly on the grammatical and syntactic level, compared to the following tasks, which capture more of the semantic and meaning-related aspects of the text.

Named entity recognition (NER) NER is the task of detecting and tagging entities in text with their corresponding type, such as PERSON or LOCATION. BIO notation is also used for this task, with O representing non-entity tokens. Entity categories are pre-defined and differ based on the application. Common categories are person names, organizations, locations, time expressions, monetary values, etc. NER is a common component of information extraction systems in many domains. Despite high numbers ($\approx$ 92–93 F_1) on the canonical CoNLL-2003 newswire dataset [Sang and Meulder, 2003], current NER systems do not generalize well to new domains. NER corpora for specialized

[7] Note that it is rare that a dataset provides multiple annotations per example.

[8] `https://www.ling.upenn.edu/courses/Fall_2003/ling001/penn_treebank_pos.html`

[9] `http://universaldependencies.org/u/pos/`

domains such as medical and biochemical articles [Kim et al., 2004, Krallinger et al., 2013] have consequently been developed.

Semantic role labelling (SRL) SRL aims to model the predicate-argument structure of a sentence. It assigns each word or phrase to its corresponding semantic role such as an agent, goal, or result. The FrameNet project [Baker et al., 1998] defined the first large lexicon consisting of *frames* such as `Apply_heat` and associated roles, known as *frame elements* such as `Cook`, `Food`, and `Heating_instrument`. Predicates that evoke this frame, e.g. "fry", "bake", and "boil" are known as *lexical units*. PropBank [Kingsbury and Palmer, 2002] added a layer of semantic roles to the Penn Treebank. These semantic roles are specific to each individual verb, but employ the same tags, which—inspired by Dowty [1991]—start from Arg0 that roughly indicates a proto-Agent, to Arg1, which designates a proto-Patient, etc., and are usually used in SRL applications.

Dependency parsing Dependency parsing is the task of extracting a dependency parse of a sentence that represents its grammatical structure and defines the relationships between "head" words and words, which modify those heads. Dependency parsing differs from constituency parsing, which focuses on identifying phrases and the recursive structure of a text. A *typed dependency structure* labels each relationship between words, while an *untyped dependency structure* only indicates which words depend on each other. Dependency parsing is used in many downstream applications such as coreference resolution, question answering, and information extraction as the relations between the head and its dependents serve as an approximation to the semantic relationships between a predicate and its arguments.

Topic classification Topic classification and sentiment analysis are *text classification* tasks. They assign a category not to every word but to contiguous sequences of words, such as a sentence or document. Topic classification aims to assign topics that depend on the chosen dataset, typically focusing on the news domain. As certain keywords are often highly predictive of specific topics, word order is less important for this task. Consequently, traditional BOW models with tf-idf weighted unigram and bigram features are often employed as strong baselines.

Sentiment analysis Sentiment analysis is the task of classifying the polarity of a given text. Usually this polarity is binary (`positive` or `negative`) or ternary (`positive`, `negative`, or `neutral`). Most datasets belong to domains that contain a large number of emotive texts such as movie and product reviews or tweets. In review domains, star

ratings are generally used as a proxy for sentiment. Sentiment analysis has become one of the most popular tasks in NLP. Variants of the task employ different notions of sentiment and require determining the sentiment with respect to a particular target (§6.2).

Language modelling Language modelling aims to predict for each word in a text the next word. It is generally known as an unsupervised learning problem, as it only requires access to the raw text.[10] Despite its simplicity, language modelling is fundamental to many advances in NLP (§3.3.2.3) and has many concrete practical applications such as intelligent keyboards, email response suggestion, spelling autocorrection, etc.

Bilingual dictionary induction Bilingual dictionary induction, which is also known as word-level translation, aims to assign to each word in a source language, e.g. English, its translation in a target language, such as German. The data that needs to be translated is usually an unordered list of words such as the words in a dictionary rather than the words in a sentence depicted here. Bilingual dictionary induction generally does not account for *polysemous* words that have multiple translations. In addition, word lists used for evaluation may be noisy as they are often automatically compiled based on word alignment in parallel corpora.

The tasks in Table 2.1 are ordered roughly from low-level syntactic tasks that aim to assign categories to particular grammatical or syntactic units to higher-level tasks that require some knowledge about semantics, the meaning of the words and the sentence. Throughout this thesis, we will develop methods that are applicable to multiple tasks in this hierarchy and propose approaches that allow sharing of information among different tasks (§1.2).

2.5 Conclusions

In this chapter, we have laid out background knowledge in probability and information theory and machine learning that is necessary for the subsequent chapters. We have also introduced the neural network methods and natural language processing tasks that we will work with throughout this thesis. The last section on NLP in particular sets the scene for the next chapter on transfer learning, where we will deal with transferring knowledge across different NLP tasks. The next chapter will also discuss neural network methods for particular transfer learning scenarios in-depth and provide more intuitions around generalization in machine learning (§2.2.4).

[10] Depending on the definition, it can also be seen as *self-supervised learning* (§3.3.2).

Chapter 3

Transfer Learning

This chapter provides an overview of the literature of transfer learning in general and for NLP in particular. It studies how machine learning models can be transferred to data outside of their training distribution and specifically focuses on transfer across different tasks, domains, and languages in NLP (§2.4). As transfer learning has been used to refer to different areas in different contexts, we will first provide a definition of transfer learning. Based on this definition, we will define a taxonomy and review the four prevalent settings of transfer learning in NLP in the corresponding sections:

1. multi-task learning (§3.2);
2. sequential transfer learning (§3.3);
3. domain adaptation (§3.4);
4. and cross-lingual learning (§3.5).

3.1 Introduction

In the classic supervised learning scenario of machine learning depicted in Figure 3.1, if we intend to train a model for some task and domain A, we assume that we are provided with labelled data for the same task and domain. We can now train a model on this dataset and expect it to perform well on unseen data of the same task and domain. In other words, we expect our data to be i.i.d. (§2.2.4). When given data for some other task or domain B, we require again labelled data of the same task or domain, which we can use to train a new model that is expected to perform well on this type of data.

The traditional supervised learning paradigm breaks down when we do not have sufficient labelled data for the desired task or domain to train a reliable model. Transfer learning allows us to deal with this scenario by leveraging the data of some related task or domain,

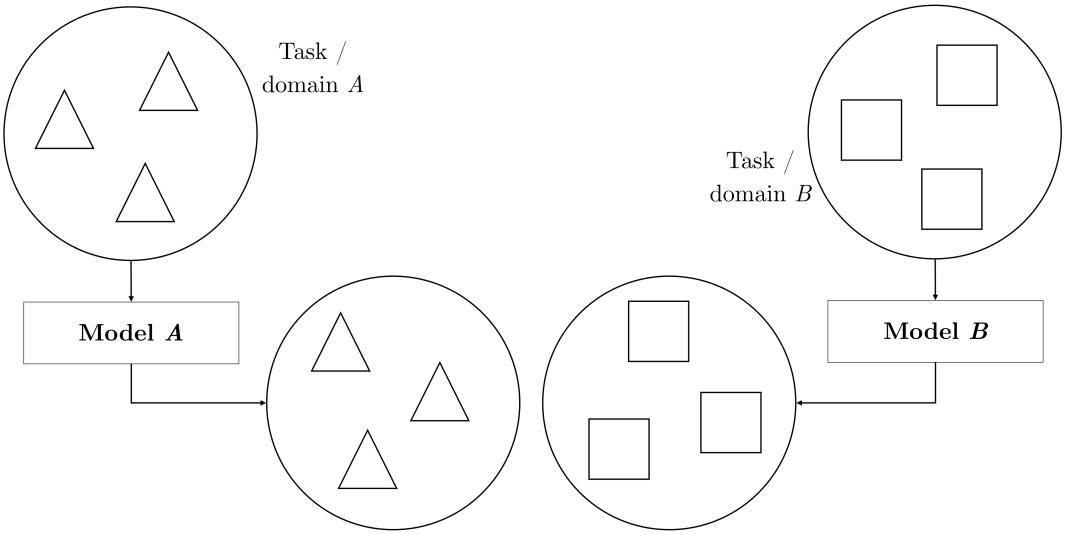


FIGURE 3.1: The traditional supervised learning setup in machine learning.

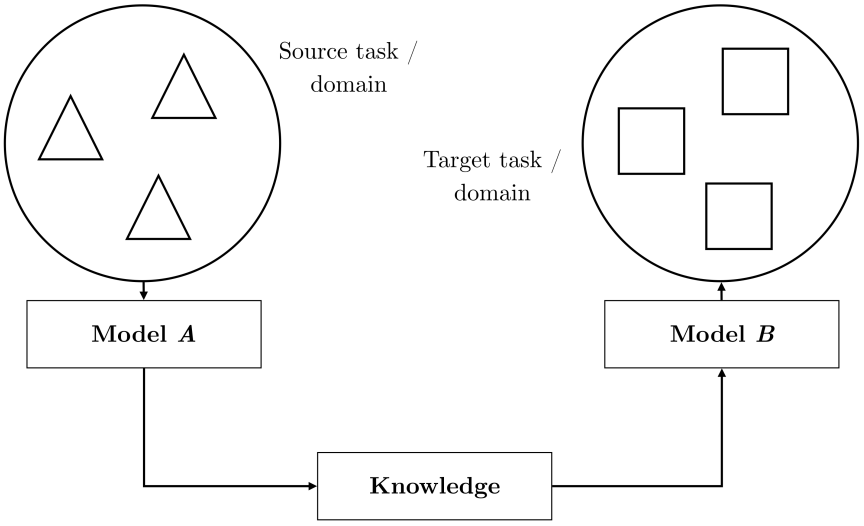


FIGURE 3.2: The transfer learning setup.

known as the *source task* and *source domain*. We store the knowledge gained in solving the source task in the source domain and apply it to the *target task* and *target domain* as can be seen in Figure 3.2.

In practice, we seek to transfer as much knowledge as we can from the source setting to our target task or domain. This knowledge can take on various forms depending on the task and data. In most cases throughout this thesis, it relates to the representations learned by neural network models (§2.3).

3.1.1 Definition

We now provide a definition of transfer learning following the notation of Pan and Yang [2010] with binary document classification as a running example. Transfer learning involves the concepts of a domain and a task. A domain $\mathcal{D}$ consists of a feature space $\mathcal{X}$ and a marginal probability distribution $P(X)$ over the feature space, where $X = \{x_1, \cdots, x_n\} \in \mathcal{X}$. For document classification with a bag-of-words representation, $\mathcal{X}$ is the space of all document representations, x_i is the i-th term vector corresponding to some document and X is the random variable associated with the sample of documents used for training.

Given a domain $\mathcal{D} = \{\mathcal{X}, P(X)\}$, a task $\mathcal{T}$ consists of a label space $\mathcal{Y}$, a prior distribution $P(Y)$, and a conditional probability distribution $P(Y \mid X)$ that is typically learned from the training data consisting of pairs $x_i \in X$ and $y_i \in \mathcal{Y}$.[1] In our document classification example, $\mathcal{Y}$ is the set of all labels, i.e. {`True`, `False`} and y_i is either `True` or `False`.

Given a source domain $\mathcal{D}_S$, a corresponding source task $\mathcal{T}_S$, as well as a target domain $\mathcal{D}_T$ and a target task $\mathcal{T}_T$, the objective of transfer learning now is to learn the target conditional probability distribution $P_T(Y_T \mid X_T)$ in $\mathcal{D}_T$ with the information gained from $\mathcal{D}_S$ and $\mathcal{T}_S$ where $\mathcal{D}_S \neq \mathcal{D}_T$ or $\mathcal{T}_S \neq \mathcal{T}_T$. Typically, either a limited number of labeled target examples or a large number of unlabeled target examples are assumed to be available.

As both the domain $\mathcal{D}$ and the task $\mathcal{T}$ are defined as tuples, the inequalities between the different members of the tuple give rise to five transfer learning scenarios, which we will discuss below.

3.1.2 Scenarios

Given source and target domains $\mathcal{D}_S$ and $\mathcal{D}_T$ where $\mathcal{D} = \{\mathcal{X}, P(X)\}$ and source and target tasks $\mathcal{T}_S$ and $\mathcal{T}_T$ where $\mathcal{T} = \{\mathcal{Y}, P(Y), P(Y \mid X)\}$ source and target conditions can vary in five ways, which we will illustrate in the following using our document classification example. The first three scenarios deal with a mismatch between the source and target tasks, i.e. $\mathcal{T}_S \neq \mathcal{T}_T$, while the last two scenarios occur when there is a discrepancy between source and target domains, i.e. $\mathcal{D}_S \neq \mathcal{D}_T$.

1. $P_S(Y_S) \neq P_T(Y_T)$. The prior distributions of the source and target tasks are different, i.e. the documents have different label distributions. This is important in

[1] Note that the definition of Pan and Yang [2010] does not include the prior distribution $P(Y)$ for the task.

generative models, which model the prior explicitly [Chan and Ng, 2006, Finkel and Manning, 2009], but has also been applied to discriminative models [Chelba and Acero, 2006].

2. $P_S(Y_S \mid X_S) \neq P_T(Y_T \mid X_T)$. The conditional probability distributions of the source and target tasks are different, i.e. source and target documents are unbalanced with regard to their classes. This scenario is quite common in practice and approaches such as over-sampling, under-sampling, or SMOTE [Chawla et al., 2002] are widely used.
3. $\mathcal{Y}_S \neq \mathcal{Y}_T$. The label spaces between the two tasks are different, i.e. documents need to be assigned different labels in the target task. In this case, the most important other distinction is whether the tasks are learned sequentially or simultaneously. Learning multiple tasks at the same time is known as *multi-task learning* (§3.2), while we will use *sequential transfer learning* (§3.3) to denote the sequential case. In practice, this scenario usually occurs with scenario 1 and 2, as it is extremely rare for two different tasks to have different label spaces, but the same prior and conditional probability distributions.[2]
4. $P_S(X_S) \neq P_T(X_T)$. The marginal probability distributions of source and target domain are different, e.g. the documents discuss different topics. This scenario is generally known as *domain adaptation* (§3.4). Domain adaptation typically also requires tackling scenarios 1 and 2, as domains may differ in the prior and conditional distributions of their labels.
5. $\mathcal{X}_S \neq \mathcal{X}_T$. The feature spaces of the source and target domain are different, i.e. the documents are written in two different languages. In the context of natural language processing, we will refer to this scenario as *cross-lingual learning* or *cross-lingual adaptation* (§3.5).[3]

The setting when the source task is different from the target task is typically known as *inductive transfer learning*, while in the *transductive transfer learning* setting source and target tasks are the same.

3.1.3 Taxonomy

We now define a taxonomy for transfer learning for NLP based on these scenarios. Specifically, we adapt the taxonomy of Pan and Yang [2010] to the transfer learning

[2]The exception is if there exists a 1:1 mapping between source and target labels. This, however, renders the problem trivial and transfer unnecessary.

[3]The boundaries between scenarios 4 and 5 may sometimes appear blurred, as domains such as legal documents and social media posts have very different vocabularies. However, in scenario 5, we generally assume corresponding pairs of source and target features. Once we know the mapping $f : \mathcal{X}_S \rightarrow \mathcal{X}_T$, we have $P_S(f(X_S)) = P_T(X_T)$; in scenario 4, there is no such mapping. We discuss methods to learn such a mapping in Section 3.5.

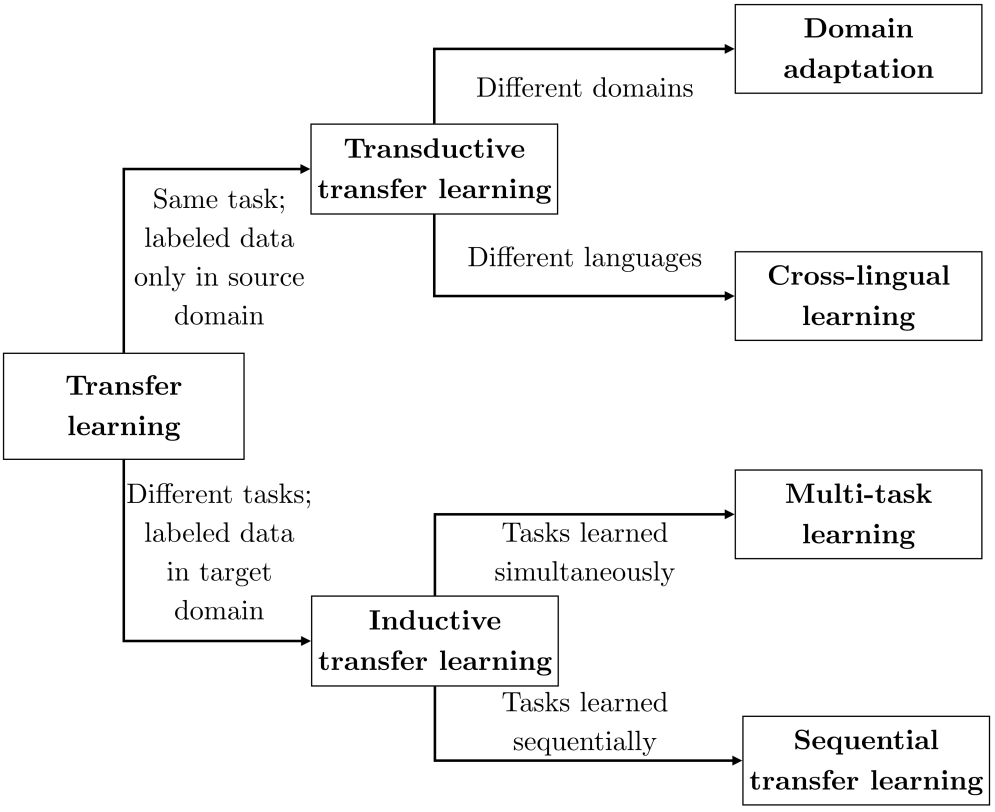


FIGURE 3.3: A taxonomy for transfer learning for NLP.

scenarios that are most commonly encountered in NLP. In particular, we make the following changes to their taxonomy:

- We add the NLP-specific setting of *cross-lingual learning*.
- We omit the *sample selection bias / covariance shift* setting that corresponds to scenario 2 as this typically treated as a special case of domain adaptation (§3.4.3.1).
- We introduce the term *sequential transfer learning* to highlight the difference to multi-task learning. We treat *self-taught learning* [Raina et al., 2007] and *unsupervised transfer learning* [Dai et al., 2008] as instances of this category.

The complete taxonomy for transfer learning for NLP can be seen in Figure 3.3. We will review approaches of these four transfer learning settings in the following. *Meta-learning* and *lifelong learning* can both be seen as instances of sequential transfer learning and will be discussed in the corresponding section.

3.2 Multi-task learning

* Multi-task learning (MTL) has led to successes in many applications of machine learning, from natural language processing [Collobert and Weston, 2008] and speech recognition [Deng et al., 2013] to computer vision [Girshick, 2015] and drug discovery [Ramsundar et al., 2015]. This section aims to give a general overview of MTL, particularly in deep neural networks. We will initially motivate MTL from different perspectives (§3.2.2). We will then introduce the two most frequently employed methods for MTL in Deep Learning (§3.2.3). Subsequently, we will describe mechanisms that illustrate why MTL works in practice (§3.2.4). We will provide context for more recent approaches by discussing the literature in MTL (§3.2.5). Finally, we will talk about commonly used types of auxiliary tasks and discuss what makes a good auxiliary task for MTL (§3.2.6).

3.2.1 Introduction

Machine learning generally involves training a model to perform a single task. By focusing on one task, however, we risk omitting information that might help us do better on the metric of interest. Specifically, we ignore knowledge that comes from the training signals of related tasks. Alternatively, by sharing representations between related tasks, we can enable our model to generalize better on our original task. This approach is called multi-task learning. Specifically, "MTL improves generalization by leveraging the domain-specific information contained in the training signals of related tasks" [Caruana, 1998].

MTL is also known as joint learning, learning to learn, and learning with auxiliary tasks. Generally, as soon an optimization problem involves more than one loss function, we are effectively doing multi-task learning (in contrast to *single-task learning*). In such scenarios, it helps to think about what we are trying to do explicitly in terms of MTL and to draw insights from it. Even if we are only optimizing one loss as is the typical case, chances are there is an auxiliary task that could help improve performance on our main task.

3.2.2 Motivation

We can motivate multi-task learning in different ways: Biologically, we can see multi-task learning as being inspired by human learning. For learning new tasks, we often apply the knowledge we have acquired by learning related tasks. For instance, a baby first learns

*This section is adapted from: **Ruder, S.** (2017). An Overview of Multi-Task Learning in Deep Neural Networks. *arXiv preprint arXiv:1706.05098.*

to recognize faces and can then apply this knowledge to recognize other objects [Wallis and Bülthoff, 1999].

From a pedagogical perspective, we often learn tasks first that provide us with the necessary skills to master more complex techniques. This is true for learning the proper way of falling in martial arts as much as learning a language [Brown and Lee, 1994].

Finally, we can motivate multi-task learning from a machine learning point of view: Multi-task learning introduces an inductive bias provided by the auxiliary tasks, which cause the model to prefer hypotheses that explain more than one task [Caruana, 1993]. As we will see shortly, this generally leads to solutions that generalize better.

3.2.3 Two methods for MTL in neural networks

So far, we have focused on theoretical motivations for MTL. To make the ideas of MTL more concrete, we will now look at the two most commonly used ways to perform multi-task learning in deep neural networks. In the context of deep learning, multi-task learning is typically done with either *hard* or *soft parameter sharing* of hidden layers.

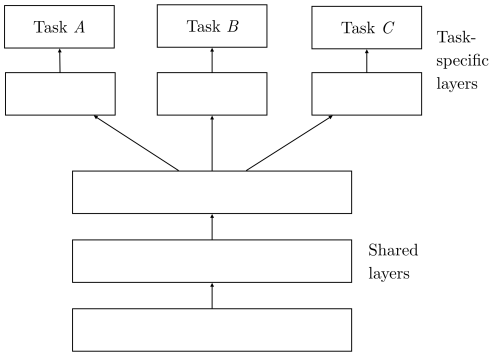


FIGURE 3.4: Hard parameter sharing

Hard parameter sharing Hard parameter sharing is the most commonly used approach to MTL in neural networks and goes back to Caruana [1993]. It is generally applied by sharing the hidden layers between all tasks, while keeping several task-specific output layers as can be seen in Figure 3.4.

Hard parameter sharing greatly reduces the risk of overfitting. In fact, Baxter [1997] showed that the risk of overfitting the shared parameters is an order T smaller than overfitting the task-specific output layers where T is the number of tasks. This makes sense intuitively: The more tasks we are learning simultaneously, the more our model has to find a representation that captures all of the tasks and the smaller is our chance of overfitting on the original task.

Soft parameter sharing In soft parameter sharing on the other hand, each task has its own model with its own parameters. The distance between the parameters of the model is then regularized in order to encourage the parameters to be similar, as shown in Figure 3.5. Duong et al. [2015b] for instance use ℓ_2 distance for regularization, while Yang and Hospedales [2017] use the trace norm.

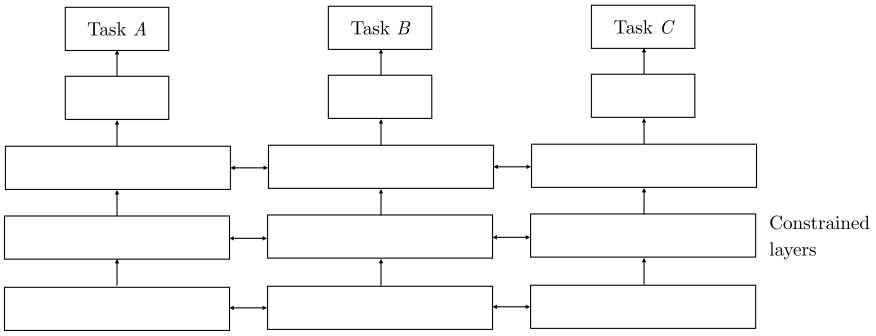


FIGURE 3.5: Soft parameter sharing

The constraints used for soft parameter sharing in deep neural networks have been greatly inspired by regularization techniques for MTL that have been developed for other models, which we will soon discuss.

3.2.4 Why does MTL work?

Even though an inductive bias obtained through multi-task learning seems intuitively plausible, in order to understand MTL better, we need to look at the mechanisms that underlie it. Most of these have first been proposed by Caruana [1998]. For all examples, we will assume that we have two related tasks A and B, which rely on a common hidden layer representation F.

Implicit data augmentation MTL effectively increases the sample size that we are using for training our model. As all tasks are at least somewhat noisy, when training a model on some task A, our aim is to learn a good representation for task A that ideally ignores the data-dependent noise and generalizes well. As different tasks have different noise patterns, a model that learns two tasks simultaneously is able to learn a more general representation. Learning just task A bears the risk of overfitting to task A, while learning A and B jointly enables the model to obtain a better representation F through averaging the noise patterns.

Attention focusing If a task is very noisy or data is limited and high-dimensional, it can be difficult for a model to differentiate between relevant and irrelevant features. MTL can help the model focus its attention on those features that actually matter as other tasks will provide additional evidence for the relevance or irrelevance of those features.

Eavesdropping Some features G are easy to learn for some task B, while being difficult to learn for another task A. This might either be because A interacts with the features in a more complex way or because other features are impeding the model's ability to learn G. Through MTL, we can allow the model to *eavesdrop*, i.e. learn G through task B. The easiest way to do this is through *hints* [Abu-Mostafa, 1990], which directly train the model to predict the most important features.

Representation bias MTL biases the model to prefer representations that other tasks also prefer. This will also help the model to generalize to new tasks in the future as a hypothesis space that performs well for a sufficiently large number of training tasks will also perform well for learning novel tasks, as long as they are from the same environment [Baxter, 2000].

Regularization Finally, MTL acts as a regularizer by introducing an inductive bias. As such, it reduces the risk of overfitting as well as the Rademacher complexity of the model, which is its ability to fit random noise [Søgaard and Goldberg, 2016].

3.2.5 MTL in non-neural models

In order to better understand MTL in deep neural networks, we will now look to the existing literature on MTL for linear models, kernel methods, and Bayesian algorithms. In particular, we will discuss two main ideas that have been pervasive throughout the history of multi-task learning:

1. enforcing sparsity across tasks through norm regularization;
2. and modelling the relationships between tasks.

Most of these approaches, as they are based on regularization, can be seen as forms of soft parameter sharing. Many classic methods are based on the idea that tasks should share a small set of active features that are common across tasks.

Many MTL approaches in the literature deal with *homogeneous multi-task learning*: They assume that all tasks are associated with a single output, e.g. the multi-class MNIST dataset is typically cast as 10 binary classification tasks. More recent approaches

deal with a more realistic, *heterogeneous multi-task learning* setting where each task corresponds to a unique set of outputs such as distinct NLP tasks (§2.4) that are learned jointly.

3.2.5.1 Block-sparse regularization

Notation In order to better connect the following approaches, let us first introduce some notation. We have T tasks. For each task t, we have a model m_t with parameters a_t of dimensionality d. We can write the parameters as a column vector a_t:

$$a_t = \begin{bmatrix} a_{1,t} \\ \vdots \\ a_{d,t} \end{bmatrix}^\top$$

We now stack these column vectors $a_1, \ldots, a_T$ column by column to form a matrix $A \in \mathbb{R}^{d \times T}$. The i-th row of A then contains the parameter $a_{i,\cdot}$ corresponding to the i-th feature of the model for every task, while the j-th column of A contains the parameters $a_{\cdot,j}$ corresponding to the j-th model.

Many existing methods make the assumption that a sparse set of features are shared across tasks [Argyriou and Pontil, 2007]. In terms of our task parameter matrix A, this means that all but a few rows are 0, which corresponds to only a few features being used across *all* tasks. In order to enforce this, they generalize the ℓ_1 norm (§2.2.5) to the MTL setting. Recall that the ℓ_1 norm, also known as lasso, is a constraint on the sum of the parameters, which forces all but a few parameters to be exactly 0.

While in the single-task setting, the ℓ_1 norm is computed based on the parameter vector a_t of the respective task t, for MTL we compute it over our task parameter matrix A. In order to do this, we first compute an ℓ_q norm across each row a_i containing the parameter corresponding to the i-th feature across all tasks, which yields a vector $b = \begin{bmatrix} \|a_1\|_q \ldots \|a_d\|_q \end{bmatrix} \in \mathbb{R}^d$. We then compute the ℓ_1 norm of this vector, which forces all but a few entries of b, i.e. rows in A to be 0.

Block-sparse regularization As we can see, depending on what constraint we would like to place on each row, we can use a different ℓ_q. In general, we refer to these mixed-norm constraints as ℓ_1/ℓ_q norms. They are also known as *block-sparse regularization*, as they lead to entire rows of A being set to 0. Zhang and Huang [2008] use ℓ_1/ℓ_∞ regularization, while Argyriou and Pontil [2007] use a mixed ℓ_1/ℓ_2 norm. The latter is also known as *group lasso* and was first proposed by Yuan and Lin [2006].

Argyriou and Pontil [2007] also show that the problem of optimizing the non-convex group lasso can be made convex by penalizing the trace norm of A, which forces A to be low-rank and thereby constrains the column parameter vectors $a_{\cdot,1}, \ldots, a_{\cdot,t}$ to live in a low-dimensional subspace. Lounici et al. [2009] furthermore establish upper bounds for using the group lasso in multi-task learning.

As much as this block-sparse regularization is intuitively plausible, it is very dependent on the extent to which the features are shared across tasks. Negahban and Wainwright [2008] show that if features do not overlap by much, ℓ_1/ℓ_q regularization might actually be worse than element-wise ℓ_1 regularization.

For this reason, Jalali et al. [2010] improve upon block-sparse models by proposing a method that combines block-sparse and element-wise sparse regularization. They decompose the task parameter matrix A into two matrices B and S where $A = B + S$. B is then enforced to be block-sparse using ℓ_1/ℓ_∞ regularization, while S is made element-wise sparse using lasso. Recently, Liu et al. [2016b] propose a distributed version of group-sparse regularization.

A related approach is the structural learning method by Ando and Zhang [2005a] who learn a common structure shared by multiple related tasks from a large number of automatically generated auxiliary classification problems. In the case of chunking, one choice is to treat the word at each position as an auxiliary label [Ando and Zhang, 2005b]. More generally, certain features may be masked and auxiliary classifiers are then learned to predict these based on the observed input. Structural correspondence learning is a successful application of this method to domain adaptation (§3.4.2.2). The idea is also related to self-supervised learning (§3.3.2) and—more recently—masked language models (§3.3.2.3).

3.2.5.2 Learning task relationships

While the group-sparsity constraint forces our model to only consider a few features, these features are largely used across all tasks. All of the previous approaches thus assume that the tasks used in multi-task learning are closely related. However, each task might not be closely related to all of the available tasks. In those cases, sharing information with an unrelated task might actually hurt performance.

Clustering of tasks Rather than sparsity, we would thus like to leverage prior knowledge indicating that some tasks are related while others are not. In this scenario, a constraint that enforces a clustering of tasks might be more appropriate. Evgeniou et al.

[2005] assume that the task parameters are put together into K different clusters where C_k indicates the tasks in each cluster. The task parameters in the k-th cluster are then encouraged to be close to the cluster mean $\bar{a}_k = (\sum_{t \in C_k} a_{\cdot,t})/|C_k|$:

$$\Omega = \sum_{k=1}^{K} \sum_{t \in C_k} \rho_t \|a_{\cdot,t} - \bar{a}_k\|^2 + \rho \|\bar{a}_k\|^2 \tag{3.1}$$

where ρ_t trades of the proximity of each task parameter vector $a_{\cdot,t}$ towards its cluster mean $\bar{a}_k$ and ρ controls the magnitude of the cluster means. They apply this constraint to kernel methods, but it is equally applicable to linear models. This constraint, however, requires that the number of clusters is known in advance.

A similar constraint for SVMs was also proposed by Evgeniou and Pontil [2004]. Their constraint is inspired by Bayesian methods and seeks to make all models close to some mean model. In SVMs, the loss thus trades off having a large margin for each SVM with being close to the mean model.

Jacob et al. [2009] make the assumptions underlying cluster regularization more explicit by decomposing the penalty into three separate norms:

1. A global penalty which measures how large our column parameter vectors are on average: $\Omega_{mean}(A) = \|\bar{a}\|^2$ where $\bar{a}$ is the average weight vector of all tasks.
2. A measure of between-cluster variance that measures how close to each other the clusters are: $\Omega_{between}(A) = \sum_{k=1}^{K} |C_k| \|\bar{a}_k - \bar{a}\|^2$.
3. A measure of within-cluster variance that gauges how compact each cluster is: $\Omega_{within} = \sum_{k=1}^{K} \sum_{t \in C_k} \|a_{\cdot,t} - \bar{a}_k\|$.

The final constraint then is the weighted sum of the three norms:

$$\Omega(A) = \lambda_1 \Omega_{mean}(A) + \lambda_2 \Omega_{between}(A) + \lambda_3 \Omega_{within}(A)$$

As this constraint assumes clusters are known in advance, they introduce a convex relaxation of the above penalty that allows to learn the clusters at the same time.

In another scenario, tasks might not occur in clusters but have an inherent structure. Kim and Xing [2010] extend the group lasso to deal with tasks that occur in a tree structure, while Chen et al. [2010] apply it to tasks with graph structures. While the previous approaches to modelling the relationship between tasks employ norm regularization, Thrun [1996] were the first ones who presented a task clustering algorithm using k-nearest neighbours.

Bayesian methods Much other work on learning task relationships for multi-task learning uses Bayesian methods: Heskes [2000] propose a Bayesian neural network for multi-task learning by placing a prior on the model parameters to encourage similar parameters across tasks. Lawrence and Platt [2004] extend Gaussian processes (GP) to MTL by inferring parameters for a shared covariance matrix. As this is computationally very expensive, they adopt a sparse approximation scheme that greedily selects the most informative examples. Yu et al. [2005] also use GP for MTL by assuming that all models are sampled from a common prior.

Bakker and Heskes [2003] place a Gaussian as a prior distribution on each task-specific layer. In order to encourage similarity between different tasks, they propose to make the mean task-dependent and introduce a clustering of the tasks using a mixture distribution. Importantly, they require task characteristics that define the clusters and the number of mixtures to be specified in advance.

Building on this, Xue et al. [2007] draw the distribution from a Dirichlet process and enable the model to learn the similarity between tasks as well as the number of clusters. They then share the same model among all tasks in the same cluster. Daumé III [2009] proposes a hierarchical Bayesian model, which learns a latent task hierarchy, while Zhang and Yeung [2010] use a GP-based regularization for MTL and extend a previous GP-based approach to be more computationally feasible in larger settings.

Online multi-task learning Other approaches focus on the online multi-task learning setting: Cavallanti et al. [2010] adapt some existing methods such as the approach by Evgeniou et al. [2005] to the online setting. They also propose a MTL extension of the regularized Perceptron, which encodes task relatedness in a matrix. They use different forms of regularization to bias this task relatedness matrix, e.g. the closeness of the task characteristic vectors or the dimension of the spanned subspace. Importantly, similar to some earlier approaches, they require the task characteristics that make up this matrix to be provided in advance. Saha et al. [2011] then extend the previous approach by learning the task relationship matrix.

Grouping of tasks Kang et al. [2011] assume that tasks form disjoint groups and that the tasks within each group lie in a low-dimensional subspace. Within each group, tasks share the same feature representation whose parameters are learned jointly together with the group assignment matrix using an alternating minimization scheme. However, a total disjointness between groups might not be the ideal way, as the tasks might still share some features that are helpful for prediction.

Kumar and Daumé III [2012] in turn allow two tasks from different groups to overlap by assuming that there exist a small number of latent basis tasks. They then model the parameter vector a_t of every actual task t as a linear combination of these: $a_t = Ls_t$ where $L \in \mathbb{R}^{k \times d}$ is a matrix containing the parameter vectors of k latent tasks, while $s_t \in \mathbb{R}^k$ is a vector containing the coefficients of the linear combination. In addition, they constrain the linear combination to be sparse in the latent tasks; the overlap in the sparsity patterns between two tasks then controls the amount of sharing between these. Finally, Crammer and Mansour [2012] learn a small pool of shared hypotheses and then map each task to a single hypothesis.

3.2.6 Auxiliary tasks

MTL is a natural fit in situations where we are interested in obtaining predictions for multiple tasks at once. Such scenarios are common for instance in finance or economics forecasting where we might want to predict the value of many possibly related indicators; marketing where multiple consumer preferences are modelled at once [Allenby and Rossi, 1998]; or in bioinformatics where we might want to predict symptoms for multiple diseases simultaneously. In scenarios such as drug discovery, where tens or hundreds of active compounds should be predicted, MTL accuracy increases continuously with the number of tasks [Ramsundar et al., 2015].

In many situations, however, we only care about performance on one task. In this section, we will thus look at how main and auxiliary tasks should interact to maximize performance and how we can find a suitable auxiliary task in order to reap the benefits of multi-task learning.

What layers should tasks share? One of the main considerations in using multi-task learning with deep neural networks is to determine which layers should be shared. In NLP, recent work focused on finding better task hierarchies for multi-task learning: Søgaard and Goldberg [2016] show that low-level tasks such as POS tagging and NER should be supervised at lower layers when used as auxiliary tasks. Building on this finding, Hashimoto et al. [2017] pre-define a hierarchical architecture consisting of several NLP tasks as a joint model for multi-task learning. In [Sanh et al., 2019], we propose a hierarchical architecture for semantic tasks. Yang et al. [2017b] enumerate different ways layers can be shared across two sequence tagging tasks. Rather than constraining a model to a certain sharing structure, Misra et al. [2016] use what they refer to as cross-stitch units to learn a linear combination of the output of the previous layers. In Chapter 6, we will build on such approaches and propose models that automatically learn a hierarchy as well as custom sharing structures.

How should the tasks interact during training? In multi-task learning, mini-batches from different tasks are typically sampled uniformly at random. As multi-task learning minimizes a weighted sum of task losses $\frac{1}{T}\sum_{i=1}^{T}\lambda_i\mathcal{L}_i$ where T is the number of tasks, choosing appropriate weights λ for each task is crucial. These weights can be tuned on a validation set just as any other hyper-parameter. The default choice, which works reasonably well in practice is to weigh all tasks equally by setting $\lambda_1 = \ldots = \lambda_T = c$ where c is an arbitrary constant. More principled approaches are possible such as using uncertainty to weigh the task losses [Kendall et al., 2018].

Alternatively, we can modify the sampling of tasks. Given two tasks $\mathcal{T}_1$ and $\mathcal{T}_2$ that are sampled with probabilities p_1 and p_2 respectively, if $p_1 = 2p_2$, we are effectively weighing $\mathcal{T}_1$ with $\lambda_1 = 2\lambda_2$ as examples are seen twice as often. Adjusting the sampling ratio of different tasks thus has the same effect as assigning different weights. Kiperwasser and Ballesteros [2018] propose different predefined sampling schedules. In Sanh et al. [2019], we propose a proportional sampling strategy, which samples mini-batches proportional to the number of training examples of a task.

As multi-task learning becomes a common tool, sampling strategies will become more elaborate. In particular, as schedules are used that adjust the sampling strategy over the course of training, multi-task learning and sequential transfer learning will move closer together (§3.3.1).

What auxiliary tasks are helpful? Finding an auxiliary task is largely based on the assumption that the auxiliary task should be related to the main task in some way and that it should be helpful for predicting the main task.

However, we still do not have a good notion of when two tasks should be considered similar or related. Caruana [1998] defines two tasks to be similar if they use the same features to make a decision. Baxter [2000] argues only theoretically that related tasks share a common optimal hypothesis class, i.e. have the same inductive bias. Ben-David and Schuller [2003] propose that two tasks are $\mathcal{F}$-related if the data for both tasks can be generated from a fixed probability distribution using a set of transformations $\mathcal{F}$. While this allows to reason over tasks where different sensors collect data for the same classification problem, e.g. object recognition with data from cameras with different angles and lighting conditions, it is not applicable to tasks that do not deal with the same problem. Xue et al. [2007] finally argue that two tasks are similar if their classification boundaries, i.e. parameter vectors are close.

Recent work [Alonso and Plank, 2017] has found auxiliary tasks with compact and uniform label distributions to be preferable for sequence tagging problems in NLP, which

we have confirmed in experiments (§6.1). [Bingel and Søgaard, 2017] find that such tasks with uniform label distributions are less likely to get stuck in suboptimal minima and observe that gains are more likely for main tasks that quickly plateau with non-plateauing auxiliary tasks. The most comprehensive study on task relatedness to this date has been done in computer vision (CV), albeit for sequential transfer learning: Zamir et al. [2018] propose a task taxonomy that organizes 26 CV tasks based on how well a model pretrained on one task transfers to another task. In the following, we will provide guidelines for choosing a suitable auxiliary task and review the most common types of auxiliary tasks used in NLP.

3.2.6.1 Common types of auxiliary tasks

The most common types of auxiliary tasks can be broadly categorized into four types based on the source of supervision and the amount of information that the model needs to predict, from a few bits to millions of bits per sample:

1. **Statistical**: This category of auxiliary tasks requires predicting low-level information related to the statistics of the input data, which are typically a few bits per sample.
2. **Selective unsupervised**: Selective unsupervised auxiliary tasks selectively predict certain parts of the input. They typically require predicting 10s to 100s of bits for some samples.
3. **Supervised**: Supervised auxiliary tasks leverage data from a related or adversarial task or additional supervision that is closely linked to the data, such as the inverse task, a coarse-grained variant, and features that are either unhelpful or only available after prediction. Supervised auxiliary tasks require predicting 10s to 10,000s bits per sample.
4. **Unsupervised**: These auxiliary tasks generally require predicting all parts of the input, which can amount to millions of bits per sample.

Statistical auxiliary tasks

Statistical auxiliary tasks involve predicting certain underlying statistics of the data. Caruana [1998] uses tasks that predict different statistics of the road, such as its intensity as auxiliary tasks for predicting the steering direction in a self-driving car. Plank et al. [2016] predict the log frequency of a word as an auxiliary task for sequence tagging. Intuitively, this makes the representation predictive of frequency, which encourages the model to not share representations between common and rare words, which benefits the handling of rare tokens. For text-to-speech, Arik et al. [2017] jointly predict the phoneme

duration and frequency profile. Another facet of this auxiliary task is to predict attributes of the user, such as their gender, which has been shown to be beneficial for predicting mental health conditions [Benton et al., 2017] or other demographic information [Roy, 2017]. We can think of other statistics that might be beneficial for a model to encode, such as the frequency of POS tags, parsing structures, or entities, the preferences of users, a sentence's coverage for summarization, or a user's website usage patterns.

Selective unsupervised auxiliary tasks

As mentioned before, MTL can be used to learn features that might not be easy to learn just using the original task. An effective way to achieve this is to use selective unsupervised auxiliary tasks. A prominent example of such auxiliary tasks are hints [Abu-Mostafa, 1990], i.e. predicting the features as an auxiliary task. Such hints can focus the model's attention on these characteristics by encouraging it explicitly to predict them. For sentiment analysis, Yu and Jiang [2016] predict whether the sentence contains a positive or negative domain-independent sentiment word, which sensitizes the model towards the sentiment of the words in the sentence. For name error detection, Cheng et al. [2015] predict if a sentence contains a name. Hints are also useful for focusing a model's attention. For learning to steer [Caruana, 1998], a single-task model might typically ignore lane markings as these make up only a small part of the image and are not always present. Predicting lane markings as auxiliary task, however, forces the model to learn to represent them; this knowledge can then also be used for the main task. Hints are useful whenever a task includes certain highly predictive terms or features and are a common source of distant supervision (§3.3.2.1).

Supervised auxiliary tasks

Related task Using a related supervised task as an auxiliary task for MTL is the classical choice. Prominent examples include Zhang et al. [2014] using head pose estimation and facial attribute inference as auxiliary tasks for facial landmark detection; Liu et al. [2015] jointly learning query classification and web search; and Girshick [2015] jointly performing object detection and region proposal, i.e. predicting the coordinates of an object in an image. As using an existing related task is by far the most common choice, we will discuss common related tasks used in NLP in Section 3.2.6.2.

Adversarial In some circumstances, we have access to a task that is *opposite* of what we want to achieve. This data can be leveraged using an adversarial loss, which does not seek to minimize but maximize the training error. This loss is commonly used in

domain adaptation and will be further discussed in Section 3.4.2.2. An adversarial loss can be added to many tasks in order to learn task-independent representations [Liu et al., 2017]. It can also be used with a statistical or selective unsupervised auxiliary task to ignore certain features of the input that have been found to be detrimental to generalization, such as word frequency information [Gong et al., 2018]. Finally, an adversarial auxiliary task might also help to combat bias and ensure more privacy by encouraging the model to learn representations, which do not contain information that would allow the reconstruction of sensitive user attributes [Li et al., 2018b, Elazar and Goldberg, 2018].

Learning the inverse Another auxiliary task that might be useful in many circumstances is to learn the inverse of the task together with the main task. A popular example of this framework is CycleGAN [Zhu et al., 2017], which can generate photos from paintings. An inverse auxiliary loss, however, is applicable to many other tasks: MT might be the most intuitive, as every translation direction such as English→French directly provides data for the inverse direction, as Xia et al. [2016] demonstrate. Enforcing cycle-consistency enables unsupervised MT [Lample et al., 2018b]. Xia et al. [2017] show that this has applications not only to MT, but also to image classification (with image generation as its inverse) and sentiment classification (paired with sentence generation). For multimodal translation, Elliott and Kádár [2017] jointly learn an inverse task by predicting image representations. It is not difficult to think of inverse complements for many other tasks: Entailment↔hypothesis generation; video captioning↔video generation; speech recognition↔speech synthesis, etc.

Quantization smoothing For many tasks, the training objective is quantized, i.e. while a continuous scale might be more plausible, labels are available as a discrete set. This is the case in many scenarios that require human assessment for data gathering, such as predicting the risk of a disease (e.g. `low`, `medium`, `high`) or sentiment analysis (`positive`, `neutral`, `negative`). Using less quantized auxiliary tasks might help in these cases, as they might be learned more easily due to their objective being smoother. For instance, Balikas and Moura [2017] jointly learn fine-grained and coarse-grained sentiment analysis.

Predicting inputs In some scenarios, it is impractical to use some features as inputs as they are unhelpful for predicting the desired objective. However, they might still be able to guide the learning of the task. In those cases, the features can be used as outputs rather than inputs. Caruana and de Sa [1997] present several problems where this is applicable.

Using the future to predict the present In many situations, some features only become available *after* the predictions are supposed to be made. For instance, for self-driving cars, more accurate measurements of obstacles and lane markings can be made only once the car is passing them. Caruana [1998] also gives the example of pneumonia prediction, after which the results of additional medical trials will be available. For these examples, the additional data cannot be used as features as it will not be available as input at runtime. However, it can be used as an auxiliary task to impart additional knowledge to the model during training.

Unsupervised auxiliary tasks

Representation learning The goal of an auxiliary task in MTL is to enable the model to learn representations that are shared or helpful for the main task. All auxiliary tasks discussed so far do this implicitly: They are closely related to the main task, so that learning them likely allows the model to learn beneficial representations. On the other hand, a model can also be trained explicitly with an unsupervised task that is known to induce general-purpose representations such as language modelling [Rei, 2017]. Such general-purpose tasks will be further discussed in Section 3.3.2.

Conditioning the initial state Finally, one representation that is particularly useful to learn is the representation of the initial state of a sequence model. The initial state of a recurrent neural network is typically initialized to a $\mathbf{0}$ vector, but may also be learned. In both cases, however, it is independent of the sequence and thus unable to adapt. Weng et al. [2017] propose to add a suitable bias to the initial encoder and decoder states for NMT by training it with a variant of language modelling, predicting the bag-of-words of the sentence.

3.2.6.2 Related tasks used in NLP

We will now look in more detail at existing NLP tasks, which have been used as auxiliary tasks to improve the performance of a main task.

Speech recognition Recent multi-task learning approaches for automatic speech recognition (ASR) typically use additional supervision signals that are available in the speech recognition pipeline as auxiliary tasks to train an ASR model end-to-end. Phonetic recognition and frame-level state classification can be used as auxiliary tasks to induce helpful intermediate representations. Toshniwal et al. [2017] find that positioning the

auxiliary loss at an intermediate layer improves performance. Similarly, Arik et al. [2017] predict the phoneme duration and frequency profile as auxiliary tasks for speech synthesis.

Machine translation The main benefit MTL has brought to machine translation (MT) are models that can simultaneously encode and translate to multiple languages, which can not only improve performance, but also enable zero-shot translation: Dong et al. [2015] jointly train the decoders; Zoph and Knight [2016] jointly train the encoders, while Johnson et al. [2016] jointly train both encoders and decoders; finally, Malaviya et al. [2017] train one model to translate from 1017 languages into English.

Other tasks have also shown to be useful for MT: Luong et al. [2016] show gains using parsing and image captioning as auxiliary tasks; Niehues and Cho [2017] combine NMT with POS tagging and NER; Wu et al. [2017] jointly model the target word sequence and its dependency tree structure; finally, Kiperwasser and Ballesteros [2018] use POS tagging and dependency parsing.

Multilingual tasks Similarly to MT, it can often be beneficial to jointly train models for different languages: Gains have been shown for dependency parsing [Duong et al., 2015b, Ammar et al., 2016a], named entity recognition [Gillick et al., 2016], part-of-speech tagging [Fang and Cohn, 2017], document classification [Pappas and Popescu-belis, 2017], discourse segmentation [Braud et al., 2017], sequence tagging [Yang et al., 2016], and semantic role labelling [Mulcaire et al., 2018]. Many of these approaches rely on cross-lingual word embeddings (§3.5) for initialization.

Language grounding For grounding language in images or videos, it is often useful to enable the model to learn causal relationships in the data. For video captioning, Pasunuru and Bansal [2017] jointly perform next-frame and entailment prediction, while Hermann et al. [2017] predict the next frame and the words that represent the visual state for language learning in a simulated environment.

Semantic parsing For a task where multiple label sets or formalisms are available such as for semantic parsing, an interesting MTL strategy is to learn these formalisms together: To this end, Guo et al. [2016] jointly train on multi-typed treebanks; Peng et al. [2017] learn three semantic dependency graph formalisms simultaneously; Fan et al. [2017a] jointly learn different Alexa-based semantic parsing formalisms; Zhao and Huang [2017] jointly train a syntactic and a discourse parser; and Hershcovich et al. [2018] train on four semantic parsing representations. For more shallow semantic parsing such as

frame-semantic argument identification, Swayamdipta et al. [2017, 2018] predict whether an n-gram is syntactically meaningful, i.e. a syntactic constituent.

Representation learning Rather than learning representations based on a single loss, intuitively, representations should become more general as more tasks are used to learn them. As an example of this strategy, Hashimoto et al. [2017] jointly train a model on multiple NLP tasks, while Jernite et al. [2017] propose several discourse-based artificial auxiliary tasks for sentence representation learning. Other multi-task representation learning approaches will be discussed in Section 3.3.2.4.

Question answering For question answering (QA) and reading comprehension, it is beneficial to learn the different parts of a more complex end-to-end model together: Choi et al. [2017] jointly learn a sentence selection and answer generation model, while Wang et al. [2017] jointly train a ranking and reader model for open-domain QA.

Information retrieval For relation extraction, information related to different relations or roles can often be shared. To this end, Jiang [2009] jointly learn linear models between different relation types; Liu et al. [2015] jointly train domain classification and web search ranking; Yang and Mitchell [2017] jointly predict semantic role labels and relations; Katiyar and Cardie [2017] jointly extract entities and relations; and in [Sanh et al., 2019], we jointly learn coreference resolution, relation extraction, entity mention detection, and NER.

Chunking Chunking has been shown to benefit from being jointly trained with low-level tasks such as POS tagging [Collobert and Weston, 2008, Søgaard and Goldberg, 2016, Ruder et al., 2019a].

3.2.7 Summary

In this section, we have reviewed both the historic literature in multi-task learning as well as more recent work on MTL applied to NLP. We have provided insights on why MTL works and shared practical tips on choosing an auxiliary task. The discussed multi-task learning methods and particularly sharing of parameters are core to many approaches in other transfer learning scenarios discussed in this chapter. In the next section, we will present the closely related sequential transfer learning paradigm, which trains models sequentially rather than simultaneously.

3.3 Sequential transfer learning

This section gives an overview of sequential transfer learning, arguably the most frequently used transfer learning scenario in natural language processing and machine learning. We will discuss the most common scenario involving a source and target task (§3.3.2). In this context, we will present common distantly supervised (§3.3.2.1), supervised (§3.3.2.2), and unsupervised (§3.3.2.3) source tasks including fundamental methods in natural language processing. We will then discuss common architectures (§3.3.2.5) and transfer methods (§3.3.3). Subsequently, we will talk about the setting using multiple tasks (§3.3.4) and finally review evaluation scenarios (§3.3.5).

3.3.1 Introduction

We define *sequential transfer learning* as the setting where source and target tasks are different and training is performed in sequence. That means, models are not optimized jointly as in multi-task learning but each task is learned separately. The goal of sequential transfer learning is to transfer information from the model trained on the source task to improve performance of the target model. For this reason, sequential transfer learning is also sometimes known as *model transfer* [Wang and Zheng, 2015]. Compared to multi-task learning, sequential transfer learning is useful mainly in three scenarios:

(a) data for the tasks is not available at the same time;
(b) the source task contains much more data than the target task;
(c) adaptation to many target tasks is necessary.

Generally, sequential transfer learning is expensive when training the source model, but enables fast adaptation to a target task, while multi-task learning may be expensive when training the target model. Despite these differences, both inductive transfer learning methods are in fact closely related.

Multi-task learning vs. sequential transfer learning Multi-task learning and sequential transfer learning can be seen as being on opposite sides of a spectrum. Assume we have two tasks $\mathcal{T}_1$ and $\mathcal{T}_2$ that are trained during the intervals $[t_1, t_2]$ and $[t_3, t_4]$ respectively where $t_1 < t_2$, $t_3 < t_4$, and $t_1 \leq t_3$. In the typical multi-task learning setting, $t_1 = t_3$ and $t_2 = t_4$. In other words, both tasks commence and conclude training at the same time. In sequential transfer learning on the other hand, $t_2 < t_3$, the second task is only trained once training of the first task has terminated. Between these extremes, different *curricula* are possible where $t_3 < t_2$, which can be seen in the spectrum in Figure 3.6. *Curriculum learning* [Bengio et al., 2009] is a related technique that imposes

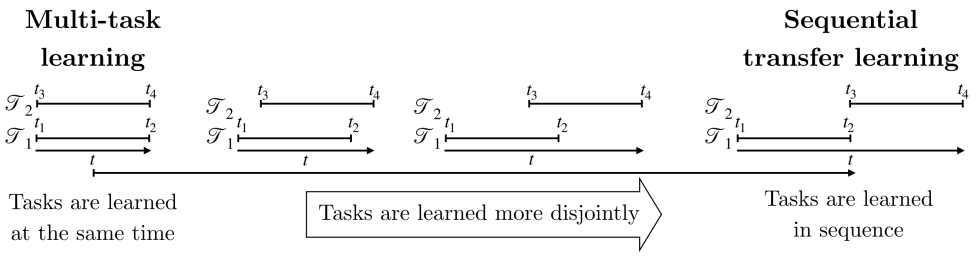


FIGURE 3.6: The inductive transfer learning spectrum with multi-task learning and sequential transfer learning at opposite ends. By progressively overlapping the training intervals of two tasks, multi-task learning gradually transforms into sequential transfer learning.

an order (usually based on a notion of difficulty) on the training examples of a *single task*.

This view is still overly simplistic as particularly in sequential transfer learning with multiple tasks (§3.3.4), it may be useful to occasionally revisit an already learned task so that it is not forgotten. As it becomes more commonplace to train an algorithm on multiple tasks, we expect the areas of multi-task learning and sequential transfer learning to converge.

Sequential transfer learning stages Sequential transfer learning typically consists of two stages: A *pretraining* phase and an *adaptation* phase. In the pretraining phase, the model is trained on the source task. In the adaptation phase, the knowledge of the trained model is transferred to the target task. While the pretraining phase may be expensive, it only needs to be performed once. In contrast, the adaptation phase is typically very efficient, which makes sequential transfer learning very useful in practice.

To maximize the usefulness of sequential transfer learning, it is common to choose a source task that will enable learning a representation that will not only help for a specific target task, but that will be useful for a wide range of target tasks. The pursuit of such *universal* representations is a long-standing challenge in representation learning and has recently received increasing attention. Due to the *no free lunch theorem* (§2.2.4), obtaining a representation that will help for *all* possible tasks is impossible. However, staying within the space of NLP tasks and assuming that leveraging the underlying structure of language will help us generalize, we can expect that we can do significantly better than training from scratch. To illustrate the impact of such universal representations, we draw a comparison to computer vision.

Sequential transfer learning in computer vision Sequential transfer learning is ubiquitous in computer vision where it is commonly practised in the form of pretraining on the ImageNet dataset [Deng et al., 2009]. As ImageNet became a common benchmark in computer vision [Krizhevsky et al., 2012b], researchers realized that the weights learned in state-of-the-art models could serve as universal representations that enable learning with as few as one example of the target task [Donahue et al., 2014]. Pretrained ImageNet models have been used to achieve state-of-the-art results in tasks such as object detection [He et al., 2017], semantic segmentation [Zhao et al., 2017], human pose estimation [Papandreou et al., 2017], and video recognition [Carreira and Zisserman, 2017]. At the same time, they have enabled the application of models to domains where the number of training examples is small and annotation is expensive. Transfer learning via pretraining on ImageNet is in fact so effective in CV that not using it is now considered foolhardy [Mahajan et al., 2018].[1]

NLP has so far not had an *ImageNet moment* [Ruder, 2018], signifying the availability of a comparably powerful source of universal representations. However, as we will see in the following, such representations may just be in reach.

3.3.2 Pretraining

Similar to the choice of an auxiliary task in multi-task learning (§3.2.6), a pretraining task should capture properties that might be useful for a number of target tasks. We consider a pretraining task to be *universal* if it helps on *most* NLP tasks.[2]

A good pretraining task provides access to a large amount of data for training the model. As unlabelled data is relatively easily accessible in most cases, the main differentiator between pretraining tasks is the *source of supervision*. We will distinguish three sources, in order of ease of obtaining the training data:

1. **No supervision**: No supervision is the easiest to obtain as it only requires access to the raw form of the data such as the unlabelled text.
2. **Distant supervision**: Distant supervision [Mintz et al., 2009] uses heuristics and domain expertise to automatically obtain large amounts of noisily supervised data.
3. **Traditional supervision**: Traditional supervision finally requires manually labelling each training example.

[1] Note that recent results suggest that ImageNet pretraining may not be useful for some specialized classification tasks [Kornblith et al., 2018] and mainly speeds up training for larger datasets [He et al., 2018a]. These studies, however, still emphasize the general utility of ImageNet pretraining.

[2] We say *most* instead of *all* as we can generally construct adversarial tasks that can be thought to be in the space of NLP tasks and that test for the non-existence of a property. For instance, we might define the goal of a task as being invariant to demographic information. Even if trained to ignore it, this property is nevertheless encoded in pretrained representations [Elazar and Goldberg, 2018].

Self-supervised learning is a technique closely related to distant supervision. Both require domain expertise to define the way the labels are generated. The main distinction is that in distant supervision, only certain instances are assigned labels that are often noisy and based on predefined patterns, while self-supervision automatically generates labels for all training examples. Self-supervision is mainly used in computer vision where pretraining tasks include predicting the position of patches in an image [Doersch et al., 2015] or predicting colour from gray scale values [Zhang et al., 2016a]. Language modelling is sometimes referred to as self-supervised learning rather than unsupervised learning [Xu et al., 2009].

While we focus on the above three sources of supervision, other forms of supervision are possible: Different heuristics can be combined to obtain more accurate labels [Ratner et al., 2016] and models pretrained on different tasks can be used to label more data [Vinyals et al., 2015]. In addition, multiple sources of supervision can be combined (§3.3.2.4).

3.3.2.1 Distantly supervised pretraining

Distant supervision was first proposed for training models for relation extraction where any sentence that contains a pair of entities that participate in a known Freebase relation was assumed to be an example of that relation [Mintz et al., 2009]. Another successful application of distant supervision is sentiment analysis on Twitter data [Go et al., 2009]. Typically, models are trained to predict emoticons in a large corpus of Twitter messages.

More recently, emoticon prediction is used as a pretraining task for deep neural networks [Severyn and Moschitti, 2015], which enables training models on millions of tweets. Felbo et al. [2017] scale up this strategy to predict a large number of emojis on more than a billion tweets. They apply their pretrained model not only to sentiment analysis, but also to emotion and sarcasm detection tasks, demonstrating that a specialized pretraining task can be useful for an array of related target tasks. We can similarly imagine distantly supervised tasks that endow the model with capabilities that are useful for certain categories of target tasks.

Distantly supervised pretraining tasks bear a striking resemblance to useful auxiliary tasks for multi-task learning. In particular, many of them can be seen as providing hints (§3.2.6.1). This has the effect of encouraging the model to concentrate on aspects that only make up a small part of its input, which would likely be ignored by a purely unsupervised model. In this vein, Yang et al. [2017a] pretrain a word segmentation model by predicting punctuation, while Ziser and Reichart [2018] pretrain a model to predict pivot unigrams or bigrams.

Another source of distant supervision that might be missed by unsupervised methods are discourse markers that indicate specific inter-sentence relations. Such a pretraining task makes the model more sensitive to how different sentences relate to each other, which can in turn improve its representation of the meaning of a sentence. Jernite et al. [2017] propose to predict conjunctions, while Nie et al. [2017] predict discourse markers such as "but" and "because".

3.3.2.2 Supervised pretraining

Compared to distantly supervised approaches, which use heuristics that are particularly suited to a certain category of tasks, supervised pretraining leverages existing tasks and datasets. In some cases existing tasks are chosen that are suitable for a particular downstream task: Zoph et al. [2016] train a machine translation model on a high-resource language pair and then transfer this model to a low-resource language pair. Yang et al. [2017a] pretrain a POS tagging model and apply it to word segmentation. Other approaches [Min et al., 2017, Golub et al., 2017, Wiese et al., 2017] pretrain a model on the large open-domain Stanford Question Answering Dataset (SQuAD) and transfer it to a more specialized QA domain.

Recent approaches employ different supervised tasks with large datasets to learn general-purpose representations. These methods select a task that is thought to require general knowledge about the structure of language. Examples of such universal supervised pretraining tasks are predicting definitions in a dictionary [Hill et al., 2016b], paraphrasing [Wieting et al., 2016], natural language inference [Conneau et al., 2017], translation [McCann et al., 2017], constituency parsing [Subramanian et al., 2018], and image captioning [Kiela et al., 2018a].

While the ImageNet task is supervised, it is questionable whether a supervised pretraining task that is inherently limited by the cost of annotation is the right choice for natural language processing given the multiple levels of meaning (§2.4) and the cornucopia of domains, tasks, and languages that NLP requires dealing with.

3.3.2.3 Unsupervised pretraining

Unsupervised pretraining, in contrast, is a much more scalable approach and closer to the way humans learn, without requiring millions of labelled examples [Carey and Bartlett, 1978, Pinker, 2013]. In theory, it is more general than supervised learning. Supervised learning just requires the model to capture in its representations whatever is necessary to the given task, discarding everything else. Unsupervised pretraining, conversely, should

allow the model to capture more general aspects of the structure and meaning of language that should then also be more transferable.

Unsupervised pretraining has also been referred to as *self-taught learning* [Raina et al., 2007] and *unsupervised transfer learning* [Dai et al., 2008]. Both leverage unlabelled data to learn representations, but differ in whether the target task is supervised or unsupervised. Raina et al. [2007] use sparse coding to learn a set of basis vectors from unlabelled data. They then solve a least squares problem to express an example as a sparse linear combination of these bases. Dai et al. [2008] use auxiliary unlabelled data to cluster target data. As their method clusters both data simultaneously, their approach is a multi-task rather than a sequential transfer learning method.

Unsupervised pretraining has a long history in natural language processing. Many of the most fundamental approaches in NLP can be considered as forms of unsupervised pretraining. Most methods focus on learning representations of words from unlabelled data. These methods broadly fall into two categories:

(a) Methods that use a variant of the task of language modelling (§2.4).
(b) Methods that use matrix factorization to factorize a word-word co-occurrence matrix.[3]

In the following, we will first review seminal approaches for pretraining word representations, then discuss more recent word embedding methods, and finally look at the latest generation of deep pretrained models.

Traditional word representations

Latent Semantic Analysis (LSA) Latent Semantic Analysis [Deerwester et al., 1990] has been one of the most widely used methods for learning dense representations of words. Given a sparse word-word co-occurrence matrix $\mathbf{C}$ obtained from a corpus, a common approach is to first replace every entry in $\mathbf{C}$ with its pointwise mutual information (PMI) [Church and Hanks, 1990] score (§2.1.3), thus yielding a PMI matrix $\mathbf{P}$.[4] We factorize $\mathbf{P}$ using singular value decomposition (SVD), which decomposes $\mathbf{P}$ into the product of three matrices:

$$\mathbf{P} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^\top \tag{3.2}$$

where $\mathbf{U}$ and $\mathbf{V}$ are in column orthonormal form and $\mathbf{\Sigma}$ is a diagonal matrix of singular values. We subsequently obtain the word embedding matrix $\mathbf{X}$ by reducing the word

[3] As Levy and Goldberg [2014] observe, some language modelling variants implicitly perform matrix factorization of a co-occurrence matrix. This distinction may thus not be as clear-cut.

[4] Positive PMI is used by Levy et al. [2017].

representations to dimensionality k the following way:

$$\mathbf{X} = \mathbf{U}_k \mathbf{\Sigma}_k \tag{3.3}$$

where $\mathbf{\Sigma}_k$ is the diagonal matrix containing the top k singular values and $\mathbf{U}_k$ is obtained by selecting the corresponding columns from $\mathbf{U}$. LSA is a form of dimensionality reduction; while using co-occurrence counts of words as features directly is impractical, reducing them to a lower dimension enables their use in many applications. Closely related to LSA is canonical correlation analysis (CCA), which has also been used to learn representations of words [Dhillon et al., 2011].

Brown clustering One problem with classic language models is that features are extremely sparse as many word pairs are never observed in a corpus. Brown clustering [Brown et al., 1992] is a seminal unsupervised pretraining technique in NLP that seeks to combat this sparsity by assigning every word to one out of C classes. Using a class-based bigram model, the conditional probability that a word follows another word can be decomposed as the product of the class *transition* probabilities $P(c_{i+1} \mid c_i)$ and the word *emission* probability $P(w_{i+1}|c_i)$:

$$P(w_{i+1}|w_i) = P(c_{i+1} \mid c_i)P(w_{i+1}|c_i) \tag{3.4}$$

where c_i is the class of w_i, the i-th word in the data.[5] The authors propose a greedy hierarchical agglomerative clustering algorithm for finding a hard class assignment for each word: The algorithm initially assigns each word to its own class. It then iteratively merges classes so that the average mutual information is maximized until C classes remain. In a post-processing step, words are re-arranged if moving them to a new class increases the average mutual information of the partition. The result of the method is a binary tree that includes the words as its leafs. The path to each word, which consists of the concatenated binary codes is then used as feature in a downstream task. Despite only relying on bigram counts, Brown clusters reveal similar relations as later neural approaches, such as grouping words together with the same morphological stem, e.g. *performed*, *perform*, *performs*, and *performing* and clustering related words with different stems, such as *attorney*, *counsel*, *trial*, *court*, and *judge*.

Brown clusters have been applied to a wide range of tasks such as NER [Miller et al., 2004, Ratinov and Roth, 2009] and dependency parsing [Koo et al., 2008, Suzuki et al., 2009]. More recently, they have also been found to be an important signal for part-of-speech tagging on Twitter [Owoputi et al., 2013] and cross-lingual NER [Mayhew et al., 2017].

[5] Note that this is very similar to a hidden Markov model.

Latent dirichlet allocation (LDA) LDA [Blei et al., 2003] is a generative probabilistic model of a document that represents each topic as a mixture over latent topics, with each topic being represented as a distribution over words. The parameters of LDA are usually estimated using Gibbs sampling [Porteous et al., 2008]. The most common use case for LDA is to explore the topics in a corpus. However, the posterior Dirichlet parameters, i.e. the distribution over topics can also be used as the representation of a document for text classification [Blei et al., 2003]. Conversely, we can also use the distribution over topics for each word as the word representation.

LDA is the only seminal unsupervised pretraining technique that does not clearly belong to either the language models nor to the matrix factorization methods. On closer inspection, LDA can be seen as a form of dimensionality reduction and is thus related to matrix factorization approaches. As a latent variable model, LDA is also similar to HMMs, which can be used as language models.

Pretrained HMM language models Despite not having been widely adopted, the use of latent variable HMM language models to learn representations for sequence labelling [Huang and Yates, 2009, 2010a,b] is the closest non-neural analogue to the current prevailing approach of pretraining language models. Huang and Yates [2009] train a latent variable model HMM on an unlabelled corpus to learn the probability of a word appearing before and after another word. They then use these probability distribution as features for the left and right contexts in a sequence labelling model. LSA is applied to these context vectors to obtain a dense representation. Huang and Yates [2010a] adapt these learned representations to semantic role labelling. Huang and Yates [2010b] use a Factorial HMM with multiple hidden layers in order to capture more aspects of the word representations. This approach is conceptually strikingly similar to the multi-layered neural network language models used these days.[6]

Word embeddings

All previous approaches learn representations of words that are then used as features in downstream models. The following approaches learn dense word representations with neural networks. In this context, such representations are also known as word embeddings. A common argument for word embeddings is that they are low-dimensional, dense, and more expressive than traditional approaches [Baroni et al., 2014]. However, classic approaches such as LSA also induce dense representations and—provided the

[6]Huang and Yates [2010b] describe their approach in a way that would not look out of place in a contemporary paper using neural language models. In their own words: "By adding in multiple hidden layers to our sequence model, we aim to learn a multi-dimensional representation that may help us to capture word features from multiple perspectives."

settings are the same—are competitive with more recent neural approaches [Levy et al., 2015].

Collobert and Weston (C&W) In a seminal paper, Collobert and Weston [2008] learn word embeddings by training a neural network on a corpus $\mathcal{C}$ to output a higher score for a correct word sequence than for an incorrect one. For this purpose, they use a max-margin loss (see Equation 2.46):

$$\mathcal{L}_{\text{MMHL}} = \sum_{i=C}^{|\mathcal{C}|-C} \sum_{w' \in V} \max(0, 1 - f([w_{i-C}, \ldots, w_i, \ldots, w_{i+C}]) + f([w_{i-C}, \ldots, w', \ldots, w_{i+C}])) \tag{3.5}$$

The outer sum iterates over all words in the corpus $\mathcal{C}$, while the inner sum iterates over all words in the vocabulary. Each word sequence consists of a center word w_i and a window of C words to its left and right. $f(\cdot)$ is a neural network that outputs a score given a word sequence and is trained to output a higher score for a word sequence occurring in the corpus (the left term) than a word sequence where the center word is replaced by an arbitrary word w' from the vocabulary (the right term).

Skip-gram with negative sampling (SGNS) Skip-gram with negative sampling [Mikolov et al., 2013a] is arguably the most popular method to learn word embeddings due to its training efficiency and robustness [Levy et al., 2015]. SGNS approximates a language model but focuses on learning efficient word representations rather than accurately modelling word probabilities. It induces representations that are good at predicting surrounding context words given a target word w_t. The objective is shown in Figure 3.7. To this end, it minimizes the negative log-likelihood of the training data under the following *skip-gram* objective:

$$\mathcal{L}_{\text{SGNS}} = -\frac{1}{|\mathcal{C}|} \sum_{t=1}^{|\mathcal{C}|} \sum_{-C \leq j \leq C, j \neq 0} \log P(w_{t+j} \mid w_t) \tag{3.6}$$

$P(w_{t+j} \mid w_t)$ is computed using the softmax function:

$$P(w_{t+j} \mid w_t) = \frac{\exp({\tilde{\mathbf{x}}_{t+j}}^\top \mathbf{x}_t)}{\sum_{i=1}^{|V|} \exp({\tilde{\mathbf{x}}_i}^\top \mathbf{x}_t)} \tag{3.7}$$

where $\mathbf{x}_i$ and $\tilde{\mathbf{x}}_i$ are the word and context word embeddings of word w_i respectively. The skip-gram architecture can be seen as a neural network without a hidden layer. The word embedding $\mathbf{x}_i$ of the input word w_i is then the same as the hidden state of the model.

This word embedding $\mathbf{x}_i$ is fed into a softmax layer, where each word has a *separate* representation $\tilde{\mathbf{x}}_i$, which represents how it behaves in the context of the input word.[7] Generally, $\mathbf{x}_i$ is used as the final word representation, although combining both $\mathbf{x}_i$ and $\tilde{\mathbf{x}}_i$ can be beneficial [Levy et al., 2015].

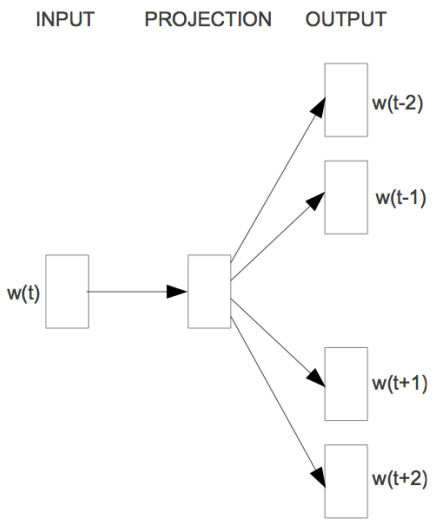


FIGURE 3.7: The skip-gram model

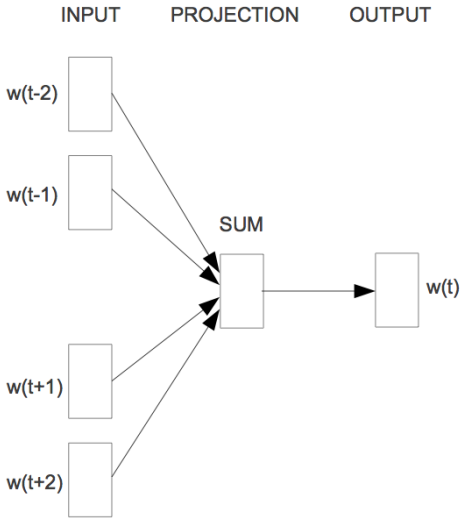


FIGURE 3.8: The continuous bag-of-words model

As the partition function in the denominator of the softmax is expensive to compute, SGNS uses *negative sampling*, which approximates the softmax to make it computationally more efficient. Negative sampling is a simplification of *noise contrastive estimation* [Gutmann and Hyvärinen, 2012], which was applied to language modeling by Mnih and Teh [2012]. Similar to noise contrastive estimation, negative sampling trains the model to distinguish a target word w_t from negative samples drawn from a noise distribution P_n. In this regard, it is similar to the C&W model as defined above, which ranks true sentences above noisy sentences. Negative sampling is defined as follows:

$$P(w_{t+j} \mid w_t) = \log \sigma({\tilde{\mathbf{x}}_{t+j}}^\top \mathbf{x}_t) + \sum_{i=1}^{k} \mathbb{E}_{w_i \sim P_n} \log \sigma(-{\tilde{\mathbf{x}}_i}^\top \mathbf{x}_t) \tag{3.8}$$

where σ is the sigmoid function and k is the number of negative samples. The distribution P_n is empirically set to the unigram distribution raised to the $3/4^{th}$ power. Levy and Goldberg [2014] observe that negative sampling does not in fact minimize the negative log-likelihood of the training data as in Equation 3.6, but rather implicitly factorizes a shifted PMI matrix, very similar to LSA.

[7] Word and context word need to have distinct representations as words hardly appear in the context of themselves. With no hidden layer, the only way for ${\tilde{\mathbf{x}}_i}^\top \mathbf{x}_i$ to have a low score is if $\tilde{\mathbf{x}}_i$ and $\mathbf{x}_i$ are different vectors [Goldberg and Levy, 2014]. Conversely, in deep language models, $\mathbf{x}_i$ and $\tilde{\mathbf{x}}_i$ are often constrained to be the same [Inan et al., 2016].

Continuous bag-of-words (CBOW) Continuous bag-of-words can be seen as the inverse of the skip-gram architecture: The model receives as input a window of C context words and seeks to predict the target word w_t by minimizing the CBOW objective:

$$\begin{aligned} \mathcal{L}_{\text{CBOW}} &= -\frac{1}{|\mathcal{C}|}\sum_{t=1}^{|\mathcal{C}|}\log P(w_t \mid w_{t-C},\ldots,w_{t-1},w_{t+1},\ldots,w_{t+C}) \\ P(w_t \mid w_{t-C},\ldots,w_{t+C}) &= \frac{\exp(\tilde{\mathbf{x}}_t{}^\top \mathbf{x}_s)}{\sum_{i=1}^{|V|}\exp(\tilde{\mathbf{x}}_i{}^\top \mathbf{x}_s)} \end{aligned} \tag{3.9}$$

where $\mathbf{x}_s$ is the sum of the word embeddings of the words $w_{t-C},\ldots,w_{t+C}$, i.e. $\mathbf{x}_s = \sum_{-C\leq j\leq C, j\neq 0}\mathbf{x}_{t+j}$. This is depicted in Figure 3.8. The CBOW architecture is typically also trained with negative sampling for the same reason as the skip-gram model.

Many extensions to the SGNS and CBOW models have been proposed over the years. Le and Mikolov [2014] extend SGNS and CBOW to model paragraphs and documents by adding a vector that represents the paragraph. Depending on the model, this vector is used to either predict the centre word or the context words. Bojanowski et al. [2017] extend SGNS by representing each word as a bag of character n-grams, which is particularly useful for modelling words in morphologically rich languages. Pagliardini et al. [2018] extend CBOW with compositional n-gram features.

Global vectors (GloVe) Global vectors [Pennington et al., 2014] learn word representations via matrix factorization. GloVe minimizes the difference between the dot product of the embeddings of a word w_i and its context word c_t and the logarithm of their number of co-occurrences within a certain window size[8]:

$$\mathcal{L}_{\text{GloVe}} = \sum_{i,j=1}^{|V|} f(\mathbf{C}_{ij})(\mathbf{x}_i{}^\top\tilde{\mathbf{x}}_j + b_i + \tilde{b}_j - \log\mathbf{C}_{ij})^2 \tag{3.10}$$

where b_i and $\tilde{b}_j$ are the biases corresponding to word w_i and its context word w_j, $\mathbf{C}_{ij}$ captures the number of times word w_i occurs with context word w_j, and $f(\cdot)$ is a weighting function that assigns relatively lower weight to rare and frequent co-occurrences.

Deep pretrained representations

All the presented neural network-based methods so far are shallow models that trade expressiveness for efficiency. In order to be scalable, they use a neural network without a hidden layer. This, however, limits the relations they can capture.

[8] GloVe favors slightly larger window sizes (up to 10 words to the right and to the left of the target word) compared to SGNS [Levy et al., 2015].

In analogy to computer vision, using word embeddings is akin to initializing an image model with pretrained representations that only encode edges: they will be helpful for many tasks, but they fail to capture higher-level information that might be even more useful [Ruder, 2018]. A model initialized with word embeddings needs to learn from scratch not only to disambiguate words, but also to derive meaning from a sequence of words. This is the core aspect of language understanding, and it requires modelling complex language phenomena such as compositionality, polysemy, anaphora, long-term dependencies, agreement, negation, and many more. It should thus come as no surprise that NLP models initialized with these shallow representations still require a huge number of examples to achieve good performance.

At the core of recent advances is one key paradigm shift: going from just initializing the first layer of our models to pretraining the entire model with hierarchical representations. If learning word vectors is like only learning edges, these approaches are like learning the full hierarchy of features, from edges to shapes to high-level semantic concepts. The following models consequently use deep neural networks to learn representations. Another advantage of such models is that document representations can be easily obtained as the hidden state of the model.

Autoencoding Dai and Le [2015] propose to pretrain an LSTM autoencoder that reconstructs the words in a sentence. Compared to language modelling, which makes a prediction for every word, the autoencoder first processes all words in a sentence and then predicts all words of the input sequence without receiving more information. The autoencoder objective is thus conceptually easier than the language modelling objective as the model does not need to "look into the future" but merely needs to "remember the past". Hill et al. [2016a] propose to use a denoising autoencoder instead. This model makes the task more challenging by adding noise to the input, which should make the learned representations more robust. The noise consists of randomly deleting words and randomly scrambling non-overlapping bigrams.

Skip-thoughts Instead of reconstructing the words in a sentence, skip-thoughts [Kiros et al., 2015] trains an RNN to reconstruct the words in the *preceding* and *succeeding* sentences. Skip-thoughts can thus be seen as an extension of SGNS to the sentence level. This notably requires training on longer documents consisting of ordered sentences. While being conceptually simple, the original skip-thoughts takes weeks to train.

Hill et al. [2016a] propose a cheaper variant that sums the word embeddings of the input sentence, while Jernite et al. [2017] present discourse-based objectives that can be used as cheaper alternatives. Their first objective trains a model to predict whether two

sentences have been switched. They frame next-sentence prediction as a classification task where given the first three sentences of a paragraph the model must choose the sentence immediately following among five candidates from later in the paragraph. Logeswaran and Lee [2018] similarly propose a more efficient formulation of skip-thoughts, which replaces the decoder with a classifier that chooses the target sentence from a set of candidate sentences. Devlin et al. [2018] further reduce next-sentence prediction to a binary classification task of predicting whether a sentence follows another sentence.

Pretrained language models While many of the earlier approaches such as Brown clustering, SGNS, and CBOW are variants of a language model, they were all proposed as approximations to make language modelling more computationally feasible with limited computational resources. With the increase in compute in recent years, it has now become feasible to pretrain deep neural language models. Dai and Le [2015] first proposed to pretrain language models, but only considered training on in-domain documents, which limited their approach. Peters et al. [2017] pretrain a language model for sequence labelling tasks, while Ramachandran et al. [2017] apply a language model to summarization and machine translation.

Peters et al. [2018a] first show that a pretrained language model is useful for a wide variety of NLP tasks. Radford et al. [2018] train a deeper model on more data. Devlin et al. [2018] introduce a masked language model objective, which randomly masks some words in the input and then predicts only the masked tokens. In contrast to a denoising autoencoder, only the masked words rather than the entire input are reconstructed. Compared to other pretraining tasks such as translation, skip-thoughts, and autoencoding, language modelling is more sample-efficient and performs best on syntactic tasks [Zhang and Bowman, 2018]. In Section 7.1, we present a pretrained language model that has achieved state-of-the-art performance on a range of text classification tasks.

The rise of pretrained language models has major implications for transfer learning for NLP. Pretrained language models have achieved improvements between 10-20% on a wide variety of tasks, some of which have been studied for decades. Pretrained language models can help with domain adaptation [Joshi et al., 2018] and enable few-shot and zero-shot learning [Howard and Ruder, 2018, Radford et al., 2018].

In addition, representations from pretrained language models capture hierarchical features that are analogous to ones captured by ImageNet models in computer vision [Krizhevsky et al., 2012b]: the word embedding layer captures morphological information; lower layers capture local syntax, while the upper layers capture longer range semantics such as coreference [Peters et al., 2018b]. Pretrained language model representations have also been shown to implicitly learn logic rules [Krishna et al., 2018].

3.3.2.4 Multi-task pretraining

Distantly supervised, supervised, and unsupervised pretraining tasks are not at odds but can complement each other. Using multi-task learning (§3.2), a model can be trained on multiple tasks at once. Pretraining on multiple tasks might help make representations more general, as a hypothesis space that performs well on a sufficiently large number of tasks is also expected to perform well on learning novel tasks from the same environment [Baxter, 2000]. Subramanian et al. [2018] perform multi-task pretraining on skip-thoughts, machine translation, constituency parsing, and natural language inference. Similarly, Cer et al. [2018] combine pretraining on skip-thoughts, response prediction, and natural language inference, while Devlin et al. [2018] combine masked language modelling with next-sentence prediction.

3.3.2.5 Architectures

While reliable word representations can be learned with shallow approaches such as SGNS and GloVe, most NLP tasks require representations of sentences and documents. Nevertheless, simple architectures such as an average of pretrained word embeddings are surprisingly strong baselines for sentence and document-level representations. Wieting et al. [2016] find that training a model based on an average of word vectors outperforms more complex architectures on their paraphrase pretraining task. Arora et al. [2017] propose to additionally subtract the principal component from a weighted average of pretrained word embeddings, while Chen [2017] propose to add noise to the average of pretrained embeddings. Another conceptually simple architecture is the deep averaging network [DAN; Iyyer et al., 2015], which averages word vectors, followed by two hidden layers. Cer et al. [2018] use DAN for pretraining and achieve similar performance compared to a more complex model.

While such simple models are fast to train, recent developments show that deeper models outperform shallow models and scale with the amount of training data and depth of the network. Wieting and Gimpel [2017] show that LSTMs outperform an average of word vectors on paraphrase pretraining and propose a hybrid model that combines the two. Conneau et al. [2017] experiment with different models and find that a BiLSTM that condenses hidden representations with max-pooling performs best. Kiros and Chan [2018] propose a more lightweight version of this model by using convolutions and a dynamic weighting of embeddings instead of recurrent connections. Recently, the Transformer [Vaswani et al., 2017], an architecture that is based on self-attention has shown convincing results [Radford et al., 2018, Cer et al., 2018, Devlin et al., 2018].

3.3.3 Adaptation

The previous section focused on different methods for the first stage of sequential transfer learning. Work on the second stage, the adaptation phase, has been a lot scarcer. In general, there are two main ways to adapt a pretrained model to a target task: *feature extraction* and *fine-tuning*. Feature extraction uses pretrained representations as features that are provided as input to a separate model, while fine-tuning trains a pretrained model directly on data of the target task.

Feature extraction In feature extraction, a model's weights are 'frozen' and the pretrained representations are used in a downstream model similar to classic feature-based approaches [Koehn et al., 2003]. For instance, pre-trained word representations can be provided as additional input to a model [Turian et al., 2010]. Another common approach is to train a linear classifier on top of a pretrained document representation [Klementiev et al., 2012b].

Fine-tuning In contrast, fine-tuning involves updating the pretrained representations. In this case, the pretrained representations are effectively used as initialization for the parameters of the model on the downstream task. In contrast, no gradients are propagated through the frozen pretrained parameters in the feature-based approach.

Feature extraction has the benefit that existing task-specific models can be reused. In addition, when the same data needs to be used repeatedly, either for different tasks or many iterations of training, extracting features once is cheaper than fine-tuning. On the other hand, fine-tuning is convenient as it requires minimal task-specific modifications and allows us to adapt a single general-purpose model to many tasks.

Fine-tuned word embeddings have generally been found to perform better than their static counterparts [Kim, 2014]. A downside of fine-tuning word embeddings is that only parameters of words that are seen during training are updated, while the embeddings of other words become *stale*. This can be harmful if the training set is very small or the test set contains many OOV words such as in reading comprehension [Dhingra et al., 2017]. When using frozen embeddings, unseen words can still be initialized with their pretrained representation if one is available. Freezing word embeddings may also be useful when training data is noisy, such as in annotation projection [Plank et al., 2018].

In [Peters et al., 2019], we compare feature extraction and fine-tuning with state-of-the-art pretrained representations. We find that both generally achieve similar performance, but that fine-tuning performs better when source and target tasks are similar, while feature extraction performs better when source and target tasks are distant.

Residual adapters Rather than adding parameters to the top of a pretrained model, a recent strategy inserts parameters *between* the layers of a pretrained model. These *residual adapters* are layers of different designs [Rebuffi et al., 2017, 2018] that have a small number of parameters compared to the entire model and are connected via residual connections with the rest of the network. During adaptation, only the adapters are fine-tuned. The residual connection enable the model to ignore the adapters in case they are not helpful for the downstream task. Similar to feature extraction, adapters enable reuse of the same pretrained parameters for many tasks, while—similar to fine-tuning—they only add a small number of task-specific parameters.

3.3.3.1 Fine-tuning settings

However, work has been scarce on *how* fine-tuning should be done. In computer vision, it is common to freeze most of the network and fine-tune only the top layers of the model [Long et al., 2015b]. As NLP models are typically more shallow than their computer vision counterparts, Felbo et al. [2017] propose *chain-thaw*, which trains one layer at a time, analogous to layer-wise pretraining in deep belief nets [Hinton et al., 2006]. In Section 7.1, we propose *gradual unfreezing*, a method that unfreezes one layer at a time until the entire network is trained jointly.

Using a learning rate that is lower than the learning rate used for training the base model is typically recommended for fine-tuning so as not to update the parameters too much. However, not much work has looked into how this learning rate should be selected. To this end, we propose a new learning rate schedule for fine-tuning in Section 7.1.

Another way to encourage the new parameters not to deviate too much from existing ones is to use an appropriate regularizer. A simple way to enforce this is to place an ℓ_2 regularizer on the difference between the pretrained and the new model parameters [Wiese et al., 2017]. Such a constraint can be seen as a sequential version of soft parameter sharing in multi-task learning (§3.2.3). A more advanced version applies the regularizer selectively to parameters that were important to the previous task, for instance using the Fisher information matrix [Kirkpatrick et al., 2017]. Rather than regularizing the parameters of the new model, the predictions of the fine-tuned model can also be encouraged to stay close to the predictions of the pretrained model, using e.g. cross-entropy [Riemer et al., 2017]. This is similar to distillation [Hinton et al., 2015] but has the downside that source and target tasks must be of the same type.

3.3.3.2 A framework for adaptation

While feature extraction and fine-tuning each have their benefits, we can also view them see as two instances of the same framework. Let us assume we are provided with a pretrained source model with parameters θ_S and L_S layers. In order to adapt this pretrained model to a different target task, we need to add a number of task-specific parameters θ_T consisting of L_T layers. These can be a linear classification layer [Howard and Ruder, 2018], special tokens [Devlin et al., 2018], or an entirely separate model [Peters et al., 2018a]. The parameters of the adapted model are then $\theta_A = \theta_S \cup \theta_T$ with $L_A = L_S + L_T$ layers where L_S and L_T contain layers in the intervals $[1, L_S]$ and $(L_S, L_A]$ respectively.

The key hyper-parameter of the adaptation process is the learning rate η, which is set to a lower value than the one used during pretraining so as not to distort the learned parameters too much. In addition, η may change during adaptation if a learning rate schedule is used. η may also vary based on the layer as different layers encode different information [Howard and Ruder, 2018] and a different η may be necessary for source and target model parameters θ_S and θ_T respectively. Let $\eta_t^{(l)}$ thus be the learning rate of the adapted model's l-th layer at iteration t. In this framework, feature extraction and fine-tuning can be defined as follows.

Feature extraction Feature extraction corresponds to the case where

$$\eta_t^{(l)} = 0, \quad \forall l \in [1, L_S] \, \forall t. \tag{3.11}$$

In other words, feature extraction sets the learning rate of layers of the source model θ_S to 0, thereby preventing parameters updates, and only trains the task-specific parameters θ_S.

Fine-tuning Fine-tuning on the other hand requires updating at least one of the source layers during adaptation:

$$\eta_t^{(l)} > 0, \quad \exists l \in [1, L_S] \, \exists t. \tag{3.12}$$

Fine-tuning only the last layers [Long et al., 2015a] and unfreezing schedules such as chain-thaw [Felbo et al., 2017] and gradual unfreezing [Howard and Ruder, 2018] that keep lower layers frozen during early epochs are special cases of this.

The framework demonstrates that feature extraction and fine-tuning are not two distinct choices, but two extremes of a spectrum. One part that has not been analyzed previously is the role of the task-specific parameters θ_T. In Section 7.2, we find that adding many

task-specific parameters with fine-tuning achieves worse performance, while adding many parameters with feature extraction is beneficial.

3.3.4 Lifelong learning

The previous sections focused on sequential transfer learning with one source and one target task. Sequential transfer learning, however, is not limited to two tasks. When applied to many tasks, it is commonly known as *lifelong learning* [Thrun, 1996, 1998]. Lifelong learning has received occasional interest in the NLP community. *Learning to learn* and *meta-learning* are sometimes used synonymously. Here, we will treat lifelong learning as the high-level setting, which refers to learning multiple tasks in sequence and discuss meta-learning as a method to achieve this.

In sequential transfer learning with two tasks, there is a clear distinction between source and target task. We want to leverage information from the source task to perform well on the target task. In lifelong learning, there is no such distinction. We are provided with T tasks in sequential order and we learn each one after the other. Crucially, we assume access to some form of knowledge base (KB), where we store the information acquired for each task and which we can use to solve the next task. According to Thrun [1998], "the acquisition, representation and transfer of domain knowledge are the key scientific concerns that arise in lifelong learning". Silver et al. [2013] present a general review of lifelong learning and outline essential ingredients, challenges, and benefits. In contrast, we focus on approaches with application to NLP.

A lifelong learning model consists of two core components: a knowledge base and a learner. The knowledge base stores information of the past tasks and is used by the learner to solve the next task. Approaches typically store this information in one of three forms, from low-level to high-level:

1. **Functional**: Individual training examples for each task are stored. This allows us to treat the lifelong learning setting as multi-task learning, but does not enable abstraction over past tasks. In addition, this setting becomes more expensive as new tasks are added. Nevertheless, due to its simplicity, it is the most commonly used setting.
2. **Relational**: Information is stored in the form of concepts, patterns, and relations extracted from previous tasks. This enables much higher-level abstraction, but requires models that can reason over relations.
3. **Representational**: Past information is stored in the form of parameters learned by a model. These can also be past hidden states of the model on data of previous

tasks, although the number of hidden states increases linearly with the number of examples.

Memory-based approaches In a classic paper, Thrun [1996] presents several algorithms for a simple lifelong learning problem where each task is a binary classification task that involves identifying a particular concept in an image, such as a dog. Each task aims to learn a function f_T from a training set X_T, which contains positive and negative examples of the concept. The training sets of the previous tasks $X_1, \ldots, X_{T-1}$ are called *support sets* as they can aid with learning f_T. In a first proposed approach, the supports sets are used to learn a representation that minimizes the distance between examples of the same concept and maximizes the distance between a concept and a negative example using a max-margin loss.

In a second approach, they learn a distance function that given two examples returns whether they belong to the same concept. In both cases, inference is done with a memory-based approach, such as k-nearest neighbours, with the KB storing all support sets. In these formulations, for a new task, a new representation or function needs to be learned from scratch, which make these methods similar to multi-task learning.

Thrun and O'Sullivan [1996] propose to cluster tasks. To this end, they determine how well task-specific distance metrics transfer across tasks via cross-validation. They then maximize the average performance of distance metrics across task clusters, with each task cluster having one distance metric. For a new task, the most related task cluster is identified and the distance metric of that cluster is used.

The early approaches proposed by Thrun [1996] learn a metric space and then leverage non-parametric memory-based models such as k-nearest neighbours for inference. Such approaches are very similar to models relying on attention [Bahdanau et al., 2015] and memory-based neural networks [Weston et al., 2015, Kumar et al., 2016a].

In this vein, Vinyals et al. [2016] propose matching networks, a neural network-based model that learns to embed concepts into a joint space for few-shot learning. It then uses attention over the examples in the support set of each class to identify a new image. Snell et al. [2017] propose prototypical networks, an extension of the former model, which uses prototypes rather than the entire support sets for inference. The prototypes are defined simply as the mean of the embeddings of the examples in the support set of each class. Ren et al. [2018] extend prototypical networks to the semi-supervised setting. Matching networks and prototypical networks are also often cited as examples of meta-learning, which will be discussed in the end of this section.

Kaiser et al. [2017] propose a memory module consisting of key-value pairs that can be added to different models for lifelong learning. The memory module thus effectively functions as the KB, which stores relevant information of past tasks. Lopez-Paz and Ranzato [2017] similarly propose to store examples of past tasks in an episodic memory. When training the model on a new task, they place inequality constraints on the model to ensure that the adapted model does not perform worse on data of the past tasks. Sprechmann et al. [2018] also store examples in a memory. At test time, the memory is used to dynamically update the parameters of the model.

Rusu et al. [2016] propose progressive networks, which consists of multiple "columns" of neural networks. Every column is a neural network that is trained on a task and then frozen. A new column is added for each new task, with lateral connections connecting each layer in the new column with the preceding layer of all previously learned columns. The model can be seen as a multi-task learning model that allows each task to converge before moving on to a new task (§3.3.1); alternatively, it can be seen as an extension of feature extraction to the lifelong setting. A deficiency of progressive networks is that the number of parameters scales linearly with the number of tasks. Schwarz et al. [2018] mitigate this weakness with Progress & Compress, which learns in a cycle of active learning (progression) followed by consolidation (compression). Rather than having a column for each previous task, the approach uses a single network as a KB and an active column that learns the current task. After learning, the active column is distilled into the KB with a modified version of elastic weight consolidation [Kirkpatrick et al., 2017].

Lifelong learning as online multi-task learning Lifelong learning can also be seen as an online form of multi-task learning. In this sense, many lifelong learning methods extend or build upon classic multi-task learning approaches (§3.2.5.2).

Ruvolo and Eaton [2013] propose an Efficient Lifelong Learning Algorithm (ELLA), an online variant of the multi-task learning model by Kumar and Daumé III [2012], which assumes a small number of latent basis tasks. The parameter vector of each task is then modelled as a linear combination of these. The optimization in this model, however, depends on the training examples of all tasks. Ruvolo and Eaton [2013] remove this dependency by approximating the optimization with a second-order Taylor expansion. In addition, task vectors are only updated when training data of the corresponding task is encountered. Ammar et al. [2014] extend this approach to policy gradient methods in reinforcement learning.

Lifelong information extraction A particular application area where lifelong learning has been applied in NLP is the continuous extraction of information from web corpora.

Banko and Etzioni [2007] propose an agent that leverages OpenIE and WordNet and learns in several steps: Given a concept, it uses extraction patterns to obtain new concept classes. In a second abstraction step, it seeks to derive more general relations for the facts extracted with OpenIE and the available concepts. In addition, they propose several search strategies to guide the agent towards which concepts and relations it should explore next.

In another case study of lifelong learning, Carlson et al. [2010] propose a "never-ending language learner" (NELL) that continuously learns to extract more information from the web. The extracted information is stored in a KB, which is initially seeded with an ontology and examples for each predicate. Facts are extracted from the web using different subcomponents and only the most supported facts are integrated into the KB. At the next iteration, the subcomponents are retrained using the updated KB and used to extract more facts in a semi-supervised way. As in semi-supervised learning (§3.4.4), it is important that models *make uncorrelated errors* and that they rely on *different views*. This method is also similar to multi-task learning, as many functions are learned jointly. Mitchell et al. [2018] extend NELL with word and relation embeddings, which are used in a separate relation classification module.

Lifelong sentiment analysis Another common application area for lifelong learning in NLP is sentiment analysis. Chen et al. [2015] propose a Naive Bayes approach for lifelong sentiment analysis. In their KB, they store how often each word occurred in positive and negative documents in the past domains. The KB also counts for each word the number of tasks where it is more likely to have positive or negative sentiment. This information is then used to regularize the Naive Bayes model when training on a new task.

For aspect-based sentiment analysis, Wang et al. [2016] apply a lifelong topic modelling method [Chen and Liu, 2014]. The method consists of three steps: LDA is first run to produce a set of prior topics. In a second step, prior knowledge sets are mined for each topic. Finally, the prior knowledge sets are used to guide the generation of better topics. Wang et al. [2016] use this method to identify similar aspects in past domains and mine terms that frequently co-occur across multiple domains, which are then incorporated into the inference process. For aspect extraction, Liu et al. [2016a] propose to first extract an initial set of aspects using high-precision rules. In a second step, they recommend additional aspects extracted with high-recall rules based on similarity and association. Shu et al. [2017] adapt a CRF for aspect extraction. They use patterns of common dependency relations as features that occur with aspects seen in past domains. Once the

model learns about new aspects, new patterns can be generated, which can help with extracting new aspects.

Meta-learning Meta-learning usually trains a model that is a level higher than the model that is typically optimized. A common strategy for meta-learning is to have a separate *controller* that modulates the learning of the base model.

Andrychowicz et al. [2016] apply this paradigm to optimization and train an LSTM controller to perform gradient descent. Ha et al. [2017] propose a "hypernetwork" that is trained to generate the weights of a neural network that is applied to another task. Chen et al. [2018] use a shared meta-LSTM to control the parameters of private layers in multi-task learning, while Wolf et al. [2018] use a higher-level LSTM to update the weights of a language model. These approaches can also be thought of as having a hierarchical model that reasons on different time-scales or abstraction levels. Such weights are sometimes referred to as *fast weights* and *slow weights* [Ba et al., 2016] in analogy to synapses operating at many different time-scales.

Another recently proposed strategy is to perform a form of meta-optimization, which learns model parameters that allow for more effective learning of a new task. Essentially, this strategy arrives at a parameter initialization that can easily be fine-tuned on a new task. The first approach in this line is model-agnostic meta-learning [MAML; Finn et al., 2017]. MAML, however, requires differentiating through the optimization process, which makes it costly for tasks with many update steps. Nichol and Schulman [2018] propose Reptile, a more scalable variant that repeatedly samples a task, trains on it, and then moves the initialization closer to the trained weights of that task. MAML has been applied to semantic parsing [Huang et al., 2018] and low-resource machine translation [Gu et al., 2018] in NLP.

Sequential transfer learning with fine-tuning can similarly be seen as aiming to find a good initialization that facilitates learning the target task. While meta-learning aims to learn such an initialization explicitly by repeatedly simulating the adaptation scenario, pretraining implicitly finds such an initialization by training with a general-purpose pretraining task. Meta-learning approaches are mainly applied to few-shot learning, as repeatedly fine-tuning on large datasets is expensive. In addition, in order to find an initialization that is applicable to an arbitrary target task, a diverse number of tasks from the environment need to be sampled during training, which is costly as it requires first to create relevant tasks and then to gather annotations.

3.3.5 Evaluation scenarios

In sequential transfer learning, in most cases the most important metric is the performance on the target task or *forward transfer* [Lopez-Paz and Ranzato, 2017]. In the lifelong setting, forward transfer applies to all future tasks. *Positive forward transfer* takes place when the performance on a previous task improves the performance on a future task. We observe *negative forward transfer* also known as *negative transfer* if the past experience impedes the learning of a new task.

Negative transfer is a key problem in domain adaptation [Rosenstein et al., 2005] where transfer between dissimilar domains e.g. according to proxy $\mathcal{A}$ distance (§3.4.2.1) leads to worse performance [Blitzer et al., 2007]. As such, it is a key obstacle that we seek to overcome by selecting more relevant and informative training examples (§4).

Negative transfer is also common in multi-task learning. Alonso and Plank [2017] for instance found MTL with hard parameter sharing only useful in one out of five tasks. To this end, we design our multi-task learning models to enable flexible sharing of parameters so that even if tasks are not similar, performance will not deteriorate (§6). As a result, our models outperform single-task learning in almost all cases.

In some settings, we also care about how well we perform on the previous tasks, which is known as *backward transfer*. We get *positive backward transfer* if we do better on a previous task after having trained on a new task. In contrast, *negative backward transfer* takes place when we do worse on previous tasks after having trained on new tasks. This phenomenon is a common occurrence when training on multiple tasks and is also known as *catastrophic forgetting*.

3.3.6 Summary

In this section, we have described sequential transfer learning, which is—due to its ease of use—the prevalent form of transfer learning with neural networks. We have discussed the pretraining and adaptation phases as well as sequential transfer learning with multiple tasks. The reviewed evaluation metrics are relevant for many other transfer learning scenarios. Mitigating negative transfer plays an important role in domain adaptation, which aims to bridge dissimilar domains and will be discussed in the next section.

3.4 Domain adaptation

Domain adaptation is useful when applying machine learning models to the real world. In the following, we will study the three main types of domain adaptation methods that have been applied in NLP:

1. representation approaches (§3.4.2);
2. data weighting and selection methods (§3.4.3);
3. and self-labelling approaches (§3.4.4).

Domain adaptation is usually studied in the setting with a single source domain. However, approaches for dealing with multiple source domains have also been proposed (§3.4.5).

3.4.1 Introduction

In machine learning training and test data are typically assumed to be i.i.d. (§2.2.4). This assumption, however, breaks down when models are applied to the real world where the distribution of the data differs from the data seen during training. This gives rise to the problem of domain adaptation, which deals with adapting from a training distribution to a different distribution at test time.

Most existing datasets do not accurately evaluate the robustness of our models as they only challenge a model's ability to *interpolate*; with a sufficient number of examples, a model can be expected to do well on examples that are similar to the ones it has seen before. In order to test how well a model actually generalizes, it needs to be able to *extrapolate* to examples that are outside of its training distribution.

For an overview of classic domain adaptation approaches, we refer the reader to the surveys of Jiang [2008] and Margolis [2011]. For a more recent overview of domain adaptation for computer vision, refer to [Patel and Gopalan, 2015] and [Csurka, 2017]. In this section, we will focus mainly on domain adaptation for NLP.

Recall that in domain adaptation, the marginal probability distributions P_S and P_T of the source domain $\mathcal{D}_S$ and target domain $\mathcal{D}_T$ are different. Compared to sequential transfer learning, domain adaptation seeks to learn representations that are beneficial for a particular target domain rather than being useful in general. Whereas sequential transfer learning requires access to labelled instances of the target task, domain adaptation is generally studied in the unsupervised setting where a sufficient number of labelled examples in the source domain and only unlabelled examples in the target domain are assumed to be available. A subset of methods study the supervised case where a small number of labelled target instances are available. Domain adaptation techniques have

also been applied to the cross-lingual setting where the feature spaces between the two domains are different, which is also known as *heterogeneous domain adaptation* [Zhou et al., 2014, 2015].

3.4.2 Representation approaches

Representation approaches try to change the underlying representation of the data. Approaches either try to identify features that are common in both domains based on a notion of domain similarity (§3.4.2.1) or try to represent both data in a shared low-dimensional space (§3.4.2.2). Furthermore, recent work has seen a combination of both strategies.

3.4.2.1 Distribution similarity approaches

The key tenant of distribution similarity approaches is to make the feature distributions of source and target domains as similar as possible. A naive way to achieve this in NLP is to simply ignore features that do not occur in the target data. Aue and Gamon [2005] and Margolis et al. [2010] apply this strategy to sentiment analysis and speech act classification respectively. Dredze et al. [2007] did not find the approach useful for parsing.

Many of the distribution similarity approaches rely on a measure of distance between the source and target domain. Here, we briefly review the most commonly used measures. For clarity, we redefine the KL and JS divergences (see Equations 2.23 and 2.24) in terms of source and target domain distributions. Recall that in practice, we can approximate the expectation $\mathbb{E}_{\mathbf{x}\sim P}[\mathbf{x}]$ with the sample mean $\frac{1}{|X|}\sum_{\mathbf{x}\in X}\mathbf{x}$ (see Equation 2.11).

Kullback-Leibler (KL) divergence The Kullback-Leibler divergence between a source domain distribution and a target domain distribution P_T is defined as follows:

$$D_{KL}(P_S||P_T) = \mathbb{E}_{\mathbf{x}\sim P_S}[\log P_S(\mathbf{x}) - \log P_T(\mathbf{x})]. \tag{3.13}$$

The KL divergence is a special case of an f-divergence, a general divergence measure between probability distributions, also known as Csiszár's f-divergence [Csiszár, 1967] or Ali-Silvey distance [Ali and Silvey, 1966].

Jensen-Shannon (JS) divergence The Jensen-Shannon divergence is a symmetric version of the KL divergence:

$$JSD(P_S||P_T) = \frac{1}{2}D_{KL}(P_S||M) + \frac{1}{2}D_{KL}(P_T||M) \tag{3.14}$$

where $M = \frac{1}{2}(P_S + P_T)$.

Rényi divergence The Rényi divergence [Rényi, 1961] is defined as:

$$D_R = \frac{1}{\alpha - 1}\mathbb{E}_{\mathbf{x}\sim P_S}[\log P_S(\mathbf{x})^{\alpha} - \log P_T(\mathbf{x})^{\alpha-1}]. \tag{3.15}$$

where $0 < \alpha < \infty$ and $\alpha \neq 1$. In the limit $\alpha \to 1$ the Rényi divergence is equivalent to the KL divergence. The above divergence measures are *parametric*, i.e. they require assumptions about the functional form of the underlying distribution in the form of parameters.[9] In comparison, the following measures require no such assumptions.

Maximum Mean Discrepancy (MMD) MMD [Borgwardt et al., 2006, Gretton et al., 2012] is a distance function between the sample means of two distributions P_S and P_T based on a mapping to a Reproducing Kernel Hilbert Space (RKHS) $\mathcal{H}$. Approaches generally employ an empirical estimate of the MMD, which is defined as:

$$MMD(P_S, P_T) = \Big\|\mathbb{E}_{\mathbf{x}\sim P_S}[f(\mathbf{x})] - \mathbb{E}_{\mathbf{x}\sim P_T}[f(\mathbf{x})]\Big\|_{\mathcal{H}} \tag{3.16}$$

where $f(\cdot) : \mathcal{X} \to \mathcal{H}$, i.e. $f(\cdot)$ maps an example $\mathbf{x}$ to the RKHS. In practice, the mapping is obtained using an appropriately chosen kernel such as an RBF kernel [Bousmalis et al., 2016] or a neural network [Tzeng et al., 2014]. In this case, MMD can be seen as measuring the distance between the means of embeddings of X_S and X_T.

Wasserstein distance The Wasserstein distance [Arjovsky et al., 2017] or Earth Mover's distance measures the minimum "cost" of transforming one distribution into the other. It can be approximated as follows:

$$W(P_S, P_T) = \max \mathbb{E}_{\mathbf{x}\sim P_S}[f(\mathbf{x})] - \mathbb{E}_{\mathbf{x}\sim P_T}[f(\mathbf{x})] \tag{3.17}$$

where the function $f(\cdot)$ can again be parameterized by a neural network. The theory requires f to be a 1-Lipschitz continuous function, i.e. a function with absolute value bounded by 1. In the original paper, this is achieved by clipping the weights. Gulrajani

[9] They can be approximated in non-parametric ways, e.g. using k-nearest neighbours [Poczos et al., 2011].

et al. [2017] improve on this by directly constraining the gradient to be close to 1. MMD and Wasserstein distance look similar and are—in fact—both members of the same family of integral probability metrics [Müller, 1997][10].

Proxy $\mathcal{A}$ distance The $\mathcal{A}$ distance [Kifer et al., 2004] is defined as follows:

$$d_{\mathcal{A}}(P_S, P_T) = 2 \sup_{A \in \mathcal{A}} |P_S(A) - P_T(A)| \tag{3.18}$$

where $\mathcal{X}$ is a feature space and $\mathcal{A}$ is a collection of subsets of $\mathcal{X}$. Intuitively, the $\mathcal{A}$ distance is the largest change in probability of a subset from source to target. Ben-David et al. [2007] show that computing the $\mathcal{A}$ distance for a finite sample is the same as minimizing the empirical risk of a classifier that is trained to differentiate between examples drawn from P_S and P_T. In practice, they compute a proxy $\mathcal{A}$ distance, which they define as:

$$pd_{\mathcal{A}}(P_S, P_T) = \frac{100}{|X_S| + |X_T|} \sum_{\mathbf{x}, y \in X_S + X_T} 1 - \mathcal{L}_\delta(\mathbf{x}, y) \tag{3.19}$$

where $\mathcal{L}_\delta$ is the Huber loss (see Equation 2.47) and $y = 1$ if $\mathbf{x} \in X_S$ and $y = 0$ if $\mathbf{x} \in X_T$. If the loss is large, the proxy $\mathcal{A}$ distance is close to 0 or negative, which indicates that the domains are indistinguishable using a linear classifier. If the domains are maximally different, the loss is 0 and the proxy $\mathcal{A}$ distance is 100; a classifier is thus easily able to tell them apart.

Blitzer et al. [2007] use the proxy $\mathcal{A}$ distance to characterise domains for sentiment analysis and find a correlation with the adaptation loss. Intuitively, the more distant two domains are in terms of a classifier's ability to differentiate them, the harder it is to adapt from one to the other. Ben-David et al. [2007] use the proxy $\mathcal{A}$ distance to analyse the representations learned by different methods. They conclude that a good domain adaptation method induces a representation that *maximally confuses* a domain classifier. This desideratum of maximal confusion has become more influential with recent deep learning-based methods, as we will see in the next section. Ravi et al. [2008] use different domain characteristics to predict the performance of a parser on a target domain, while Van Asch and Daelemans [2010] compare different distance metrics for predicting cross-domain performance. They find that Rényi divergence with $\alpha = 0.99$ performs best.

For distributional similarity approaches, the most common strategy is to use a representation that minimizes the distance between the representations of the two domains,

[10] Integral probability metrics use a class of *witness functions* to distinguish between P_S and P_T [Binkowski et al., 2018]. For Wasserstein distance, this is the class of 1-Lipschitz functions, while for MMD this is the unit ball in $\mathcal{H}$.

while at the same time maximizing the performance on the source domain data. Satpal and Sarawagi [2007] use this strategy to select features for conditional random fields, with the distance measure being the sum of distances between sample means for each feature. Arnold et al. [2007] use a similar distance measure, but scale rather than penalize features in maximum entropy classifiers. Pan et al. [2008], Chen et al. [2009], and Pan et al. [2011] minimize the MMD between the source and target feature distributions.

3.4.2.2 Latent feature learning

Latent feature learning methods aim to project the data into a low-dimensional space with the intention that this space makes the domains more similar as it is calculated based on the features of both the source and the target domain. Many approaches [Ando, 2004, Huang and Yates, 2009, Ciaramita and Chapelle, 2010, Pan et al., 2010] have used SVD for this projection, with mixed results. Guo et al. [2009] use LDA, while Blitzer et al. [2009] use CCA.

Blitzer et al. [2006, 2007] introduce structural correspondence learning (SCL). SCL applies block-sparse multi-task learning models (§3.2.5.1), specifically the idea of [Ando and Zhang, 2005a,b] to use multiple auxiliary classifiers to domain adaptation. In SCL, each task involves the prediction of a *pivot feature* with a linear classifier whose weights are stored in the column of a matrix. These pivot features can either be manually defined or automatically selected based on high mutual information with the label. SVD is then performed on the resulting matrix and the top singular vectors are used as the representation. SCL has been applied to cross-lingual sentiment classification [Prettenhofer and Stein, 2010, Wei and Pal, 2010], conversation summarization [Sandu et al., 2010], and entity recognition [Ciaramita and Chapelle, 2010], among other tasks. Tan and Cheng [2009] and Ji et al. [2011] proposed improved versions of SCL based on feature and example weighting and learning separate predictors for each domain respectively.

In another classic method, Daumé III [2007] proposes EasyAdapt, which uses source and target domain kernel functions $\phi^S(\mathbf{x})$ and $\phi^T(\mathbf{x})$ to map the representation of an example $\mathbf{x}$ to an augmented embedding space:

$$\phi^S(\mathbf{x}) = \langle \mathbf{x}, \mathbf{x}, \mathbf{0} \rangle, \quad \phi^T(\mathbf{x}) = \langle \mathbf{x}, \mathbf{0}, \mathbf{x} \rangle \tag{3.20}$$

Both domains share the first part of the embedding space, while the second and third part are specific to source and target domain respectively. This should enable the model to identify both shared and domain-specific features. While being appealing due to its simplicity, the approach is restricted to the supervised domain adaptation setting. In an

earlier method, Daumé III and Marcu [2006] augment the feature space of maximum entropy models by adding automatically learned indicator variables that indicate whether the example was drawn from the in-domain or the general-domain distribution. Daumé III et al. [2010] later extend EasyAdapt to the unsupervised setting by requiring that source and target hypothesis agree on unlabelled data, which is achieved by the feature map $\phi^U(\mathbf{x}) = \langle \mathbf{0}, \mathbf{x}, -\mathbf{x} \rangle$.

Sun et al. [2016] offer another pre-processing method that is applicable to unsupervised domain adaptation. Their approach aligns the distributions of source and target features by aligning the covariances. It then whitens the source data and re-colors it with the target covariance—both common preprocessing methods in CV. Lynn et al. [2017] propose a continuous extension of EasyAdapt to adapt to real-valued user factors. In a similar user-centric setting, Michel and Neubig [2018] propose to adapt the bias of the output softmax to a particular user for machine translation.

Dai et al. [2009] and Pan et al. [2010] both construct a graph based on co-occurrence statistics of the features and perform spectral clustering on this graph to induce a new feature representation. Ponomareva and Thelwall [2012] compare two graph algorithms, one based on node ranking and the other solving a graph optimisation problem and find that they are competitive with state-of-the-art approaches.

Neural network-based methods Glorot et al. [2011] proposed the first approach that applied a deep neural network to learn a common representation for domain adaptation. Their proposed model is a stacked denoising autoencoder (SDA) that is trained to reconstruct the input example. In a second step, a linear SVM is trained on the latent representation of the source domain training examples. Chen et al. [2012] proposed a marginalized denoising autoencoder (MDA) to address the high computational cost of the original model. Zhou et al. [2016a] propose to enforce a notion of cycle consistency (§3.2.6.1) by learning a transformation from target domain examples to the source domain and vice versa in a deep autoencoder. Ziser and Reichart [2017] combine SCL and autoencoders. The model learns to encode non-pivot features so that pivot features can be reconstructed from the induced representation.

Several approaches present more task-specific solutions. Schnabel and Schütze [2014] proposed distributional domain-agnostic features for part-of-speech tagging. Yin et al. [2015] propose to update the distributional features of the previous model in an online setting as new data comes in. Yang and Eisenstein [2014] propose a dropout variant that drops out all but one feature per template for part-of-speech tagging. Søgaard [2013] employs a similar mechanism in a Structured Perceptron where an adversary corrupts the most predictive features for part-of-speech tagging. Ma et al. [2014] use a neural

network to learn a latent representation from both words as well as sparse features for part-of-speech tagging. Qu et al. [2016] propose a model for NER that not only adapts to a domain but also to a label mismatch. They first learn a CRF on source data and then train a neural network that learns the correlation between source and target types for NER.

The most influential recent direction has been to combine the minimization of the distance between two domains of distributional similarity approaches with the embedding capabilities of latent feature learning approaches. An early approach in this line is by Tzeng et al. [2014] who add a bottleneck layer to an existing architecture. The bottleneck layer is then regularized to minimize the MMD between source and target domain representations. A similar approach by Zhuang et al. [2015] minimizes the KL divergence between the representations of source and target domain learned with a deep autoencoder. Wang et al. [2018b] minimize a label-aware extension of MMD between the source and target domain hidden representations in an LSTM for medical NER. He et al. [2018b] minimize the MMD between the source and target domain and then apply semi-supervised learning techniques to the domain-invariant representation.

Domain-adversarial approaches The most widely used approach in this line employs an adversarial loss [Ganin and Lempitsky, 2015, Ganin et al., 2016]. It can be seen as a multi-task learning model where an auxiliary classifier is trained to predict the domain of the input example (§3.2.6.1). Importantly, the gradients of this classifier are reversed. This encourages the model to learn representations that will maximally confuse the classifier and will not allow it to differentiate between the domains. Csurka and Chidlovskii [2016] propose to adapt this domain-adversarial loss for use with denoising autencoders so that it can be computed in closed form with MDA.

The domain-adversarial loss has been used for many NLP tasks, such as relevancy detection of social media posts during a crisis [Alam et al., 2018], language identification [Li et al., 2018a], relation extraction [Fu et al., 2017], and duplicate question detection [Shah et al., 2018]. Minimizing the model's loss while confusing the domain classifier is similar to the minimax game in generative adversarial networks [Goodfellow et al., 2014], which can be seen as minimizing the JS divergence between source distribution P_S and target distribution P_T [Arjovsky et al., 2017]. Alternatively, a similar result can be achieved by minimizing the Wasserstein distance between the source and target domain distributions, which can lead to more stable training [Shah et al., 2018].

Bousmalis et al. [2016] introduce an orthogonality constraint that has found frequent application in later work. This constraint encourages independence between domain-specific or *private* and general or *shared* representations in the encoder. Kim et al. [2017a]

combine many of the previous advances, specifically the autoencoder reconstruction loss, the domain-adversarial loss, and the orthogonality constraint in a single BiLSTM. Liu et al. [2018b] similarly combine adversarial training, attention, self-attention, and a memory network that stores training examples.

Building on the model by Bousmalis et al. [2016], Li et al. [2018a] propose a domain-generative model that computes the private representation using a single CNN rather than several domain-specific CNNs. In addition, the private representation is used to predict the domain, which further encourages a split between domain-specific and general aspects of the representation.

Multi-task learning Much of the work on domain adaptation is related to ideas from multi-task learning (§3.2.5.2). The framework of Daumé III [2007] is similar to regularized multi-task learning algorithm [Evgeniou and Pontil, 2004], which represents each task parameter as a sum of a mean parameter and its deviation from this mean. Lu et al. [2016] propose a framework that generalizes both and enables tuning of hyper-parameters to avoid negative transfer. Inspired by this framework, Lu and Zheng [2017] propose to learn cross-domain word embeddings by first learning embeddings for the source domain and then learning target domain word embeddings by encouraging words that are frequent in both domains to have similar representations. Kim et al. [2016] propose neural extensions of EasyAdapt that train multiple LSTMs where one captures global patterns, while the others capture domain-specific information.

Multi-task learning is often employed as a useful tool in domain adaptation, for instance by using an auxiliary domain classifier [Ganin and Lempitsky, 2015]. SCL in particular uses an array of auxiliary classifiers. Yu and Jiang [2016] propose a model inspired by SCL that uses the prediction of whether the input sentence contains a positive or negative domain-independent sentiment word as auxiliary tasks during learning.

In many cases, the boundaries between supervised domain adaptation and multi-task learning become blurred as multi-domain datasets originally designed for domain adaptation [Blitzer et al., 2006] are appropriated for multi-task learning and domains are treated as separate tasks [Long and Wang, 2015, Liu et al., 2017]. Such multi-task models, however, cannot be easily extended to unsupervised domain adaptation.

3.4.3 Weighting and selecting data

Rather than assigning a measure of importance to particular features or learning a new representation that focuses on particular aspects of the domains, other approaches perform weighting (§3.4.3.1) and selection (§3.4.3.2) at the instance level.

3.4.3.1 Instance weighting

In machine learning, we typically minimize the expected error of some loss function $\mathcal{L}$ with regard to the empirical distribution of our data (§2.2.2):

$$J(\theta) = \mathbb{E}_{\mathbf{x},y\sim P_{\text{data}}}\mathcal{L}(\mathbf{x}, y, \theta) = \sum_{i=1}^{n} P_{\text{data}}(\mathbf{x}, y)\mathcal{L}(\mathbf{x}, y, \theta). \tag{3.21}$$

In the domain adaptation setting, we would like to do the same for examples sampled from the target domain:

$$J(\theta) = \sum_{i=1}^{n} P_T(\mathbf{x}, y)\mathcal{L}(\mathbf{x}, y, \theta). \tag{3.22}$$

However, we do not have access to labelled examples from the target domain and only have examples from the source domain at our disposal. We can instead expand Equation 3.22 by multiplying and dividing by the empirical source domain distribution $P_S(\mathbf{x}, y)$:

$$J(\theta) = \sum_{i=1}^{n} \frac{P_T(\mathbf{x}, y)}{P_S(\mathbf{x}, y)} P_S(\mathbf{x}, y)L(\mathbf{x}, y, \theta). \tag{3.23}$$

We can now write this again as an expectation over the empirical source domain distribution:

$$J(\theta) = \mathbb{E}_{\mathbf{x},y\sim P_S} \frac{P_T(\mathbf{x}, y)}{P_S(\mathbf{x}, y)} L(\mathbf{x}, y, \theta). \tag{3.24}$$

For a principled instance weighting method for domain adaptation, we would thus need to weight each example from the source domain with $\frac{P_T(\mathbf{x},y)}{P_S(\mathbf{x},y)}$. In practice, exact computation of this term is infeasible, mainly because no target domain examples are available to compute $P_T(\mathbf{x}, y)$. Existing approaches thus make simplifying assumptions by assuming that the conditional probability distributions between source and target domain are equal. In particular, one category of approaches tackles the scenario of *class imbalance* and assumes $P_T(X \mid Y) = P_S(X \mid Y)$. The other category deals with covariate shift, where $P_T(Y \mid X) = P_S(Y \mid X)$. These approaches are reviewed in [Jiang, 2008] and will not be discussed further here as they deal with a weaker form of domain adaptation.

For the more interesting setting where $P_T(Y \mid X)$ differs from $P_S(Y \mid X)$, Jiang and Zhai [2007] propose a set of heuristics, which, however, require access to some labelled instances of the target domain. They first propose to remove "misleading" source domain examples where $P_T(y \mid \mathbf{x})$ differs from $P_S(y \mid \mathbf{x})$ too much. Secondly, they propose to assign higher weight to the labelled target instances.

Other instance weighting approaches compute the weights for each example by minimizing the distance between the weight-corrected source domain and target domain examples.

Huang et al. [2007] minimize MMD, while Sugiyama et al. [2008] and Tsuboi et al. [2009] minimize the KL divergence. Similar to the computation of the proxy $\mathcal{A}$ distance, Bickel et al. [2007] use the probabilities produced by a domain classifier as instance weights. In their case, both the domain classifier and the task model are learned jointly. Søgaard and Haulrich [2011] and Plank et al. [2014] similarly use instance weights based on the probabilities of a domain classifier, which, however is trained separately. Foster et al. [2010] weight phrase pairs in MT based on features that indicate whether they belong to general language or are similar to the target domain.

3.4.3.2 Instance selection

Instance weighting approaches can be seen as performing "soft" data selection. Examples with a weight $w = 0$ are ignored, $w = 1$ means normal treatment, and $w > 1$ and $w < 1$ indicates increased or reduced attention respectively. Imposing a hard selection is both more efficient and makes it easier to ignore examples that are potentially harmful. Most of the discussed approaches select examples by first obtaining a weight or relevance score for each example as in instance weighting. They then define a threshold and discard all examples whose score is lower than this threshold.

Plank and van Noord [2011] show that using similarity metrics based on topic representations works well for data selection for parsing. They find that JS divergence is the best similarity metric for word-based selection and that automatic selection performs better than selection based on human gerne and topic labels for WSJ data. Remus [2012] similarly uses JS divergence for data selection for sentiment analysis.

Xia et al. [2015] use PCA to obtain a representation of the data. They then use Hotelling's T^2 statistic [Hotelling, 1992], which indicates how much a sample deviates from the PCA space, as distance metric to select examples. In [Ruder et al., 2017a], we study different data selection metrics and representations for sentiment analysis. In [Ruder et al., 2017b], we propose a new data selection metric inspired by the clustering assumption in semi-supervised learning. We define the maximum cluster difference (MCD) as the absolute difference in similarity of an example's representation h with the cluster centroids of the positive and negative classes c_p and c_p, i.e. the mean representation of all examples assigned to the respective cluster by the model:

$$MCD = |\mathsf{cos}(c_p, h) - \mathsf{cos}(c_n, h)|. \tag{3.25}$$

The measure is motivated by the observation that incorrect predictions are frequent along the decision boundary—where the distance to both cluster centroids is small (low MCD)—but frequent along cluster edges—where the distance to the corresponding cluster

centroid is small but the distance to the other cluster centroid is large (large MCD). Guo et al. [2018] use the same metric to weight their mixture-of-experts model. In Section 4.1, we go beyond hand-crafted measures and automatically learn a data selection metric using Bayesian Optimization.

Data selection is particularly common in machine translation. Moore and Lewis [2010] use the difference between the cross-entropies of domain-specific and general n-gram language models as the score for each sentence. Axelrod et al. [2011] propose to additionally sum cross-entropy differences over each side of the corpus. Duh et al. [2013] use neural language models instead. Mirkin and Besacier [2014] explore data selection techniques and propose methods to address both similarity and coverage considerations. More recently, van der Wees et al. [2017] investigate data selection for neural MT and propose to vary the selected data between training epochs. Language model perplexity was also used to select training data for dependency parsing [Søgaard, 2011]. In general, there are fewer studies on data selection for other tasks, e.g., constituent parsing [McClosky et al., 2010], dependency parsing [Plank and van Noord, 2011, Søgaard, 2011] and sentiment analysis [Remus, 2012].

Both instance weighting and instance selection are closely related to semi-supervised learning, in particular self-labelling approaches, which will be discussed in the next section. While instance weighting and selection are typically performed as a pre-processing step, self-labelling approaches aim to select useful examples and estimate likely labels jointly during training.

3.4.4 Self-labelling approaches

Semi-supervised learning, learning from both labelled and unlabelled data has a long history. For an overview, refer to [Zhu, 2005] and [Chapelle et al., 2006]. Self-labelling approaches are a particular category of semi-supervised learning approaches that train a model on labelled examples and then use this model to assign *pseudo* or *proxy* labels to unlabelled examples. In the next iteration, these labels are then used to estimate a better model.

While semi-supervised learning approaches generally assume i.i.d. data, many of them have been used in scenarios with a domain shift. In the following section, we first detail different self-labelling approaches and then explore how they have been applied to domain adaptation.

3.4.4.1 Self-training

Self-training [Yarowsky, 1995, McClosky et al., 2006a] is one of the earliest and simplest approaches to semi-supervised learning. As the name implies, self-training leverages a model's own predictions on unlabelled data in order to obtain additional information that can be used during training. Typically, unlabelled examples with confident predictions (i.e. that have a probability higher than a threshold) are used as labelled instances during the next iteration.

Self-training has been used for domain adaptation for parsing [Roark and Bacchiani, 2003, McClosky et al., 2006b, Reichart and Rappoport, 2007, Sagae and Tsujii, 2007, Sagae, 2010, Petrov and McDonald, 2012], conversation summarization [Sandu et al., 2010], named entity recognition [Ciaramita and Chapelle, 2010], sentiment analysis [He and Zhou, 2011], and other tasks. Its main downside is that the model is not able to correct its own mistakes and errors are amplified, an effect that is increased under domain shift.

Self-training is closely connected to the Expectation Maximization (EM) algorithm, which we use for learning a bilingual lexicon (§5.2). In the domain adaptation setting, EM can be used both as "soft" and "hard" EM, with the main difference being that hard EM computes a hard assignment of hidden variables in the E step, whereas soft EM computes a distribution. Dai et al. [2007] and Tan et al. [2009] both use EM for adapting a Naive Bayes classifier for sentiment analysis. The former use the KL divergence between the source and target distributions to determine the trade-off between source and target data terms, while the latter gradually increase the weight of the target domain data.

3.4.4.2 Multi-view training

Multi-view training aims to train different models with different *views* of the data. These views can differ in various ways such as in the features they use, in the architectures of the models, or in the data on which the models are trained. Ideally the views complement each other and the models can collaborate in improving each other's performance.

Co-training Co-training [Blum and Mitchell, 1998] is a classic multi-view training method, which makes comparatively strong assumptions. It requires that the data can be represented using two conditionally independent feature sets and that each feature set is sufficient to train a good model. After the initial models are trained on their respective feature sets, at each iteration, only inputs that are confident according to *exactly one* of the two models are moved to the training set of *the other* model. One model thus provides the labels to the inputs on which the *other* model is uncertain.

In the original co-training paper [Blum and Mitchell, 1998], co-training is used to classify web pages using the text on the page as one view and the anchor text of hyperlinks on other pages pointing to the page as the other view. Nigam and Ghani [2000] propose Co-EM, a combination of EM and Co-training that alternately uses one model to assign labels to all the unlabelled data, from which the second model learns from. As two conditionally independent views are not always available, Chen et al. [2011b] propose pseudo-multiview regularization in order to split the features into two mutually exclusive views so that co-training is effective. To this end, pseudo-multiview regularization constrains the models so that at least one of them has a zero weight for each feature. This is similar to an orthogonality constraint [Bousmalis et al., 2016] (§3.4.2.2). A second constraint requires the models to be confident on different subsets of the unlabeled data. Chen et al. [2011a] use pseudo-multiview regularization to adapt co-training to domain adaptation.

Democratic Co-learning Rather than treating different feature sets as views, democratic co-learning [Zhou and Goldman, 2004] employs models with *different inductive biases*. These can be different network architectures in the case of neural networks or completely different learning algorithms.

Tri-training Tri-training [Zhou and Li, 2005] is one of the best known multi-view training methods. It can be seen as an instantiation of democratic co-learning, which leverages the agreement of three independently trained models to reduce the bias of predictions on unlabeled data. The main requirement for tri-training is that the initial models are diverse, which can be achieved using different model architectures as in democratic co-learning. The most common way to obtain diversity for tri-training is to obtain different variations of the original training data using bootstrap sampling. The three models are then trained on these bootstrap samples. An unlabelled data point is added to the training set of a model if the other two models and agree on its label. Training stops when the classifiers do not change anymore.

Søgaard and Rishøj [2010] and Søgaard [2010] incorporate disagreement into tri-training, which they apply to both dependency parsing and POS tagging, while Huang et al. [2010] use tri-training for word alignment in machine translation. In [Ruder and Plank, 2018], we propose multi-task tri-training, a more efficient extension of tri-training. Self-training and tri-training will be discussed further in Section 4.2 where we find that classic tri-training is a strong baseline for neural semi-supervised learning with and without domain shift for NLP and that it outperforms even recent state-of-the-art methods. In a similar vein, Oliver et al. [2018] analyze several state-of-the-art semi-supervised learning models and

find that performance degrades severely under a domain shift. More recently, Clark et al. [2018] propose a similar approach that uses auxiliary classifiers with restricted views of the input.

3.4.5 Multi-source domain adaptation

Multi-source domain adaptation is a special case of the regular unsupervised domain adaptation setting where data from not one but multiple source domains are available for training.

Combining source models The simplest method in this setting is to combine the training data from all source domains to train a single model, which is used by Aue and Gamon [2005] as a baseline. Alternatively, a separate model can be trained for each source domain and the models can be combined. Training each base model can be seen as a separate case of domain adaptation. For combining the classifiers, techniques from ensembling can be used. Aue and Gamon [2005] use stacking [Džeroski and Ženko, 2004] for combination, which trains a meta-model on the outputs of the base models. This approach is also used by Li and Zong [2008b]. Li and Zong [2008a] combine the ensemble approach with self-training: The ensemble model is used to assign pseudo labels to unlabelled examples, which are then used for training the base models during subsequent iterations.

Mansour [2009] propose to use a linear combination of the base models weighted based on the target distribution and provide theoretical guarantees for this rule. Chattopadhyay et al. [2012] similarly combine base models based on a linear combination. Blitzer et al. [2008] also propose learning bounds in the multi-source setting. Duan et al. [2009] learn a least-squares SVM with additional smoothness and sparsity constraints on top of base models. McClosky et al. [2010] train a meta-model that predicts the performance of source domain models based on features of the domain. Given a new target domain, the base models are then interpolated based on scores produced by the meta-model. Bollegala et al. [2011] construct a sentiment sensitive thesaurus from multiple source domains, which is then used to expand the feature vectors of a sentiment classifier.

Yoshida et al. [2011] propose a Bayesian model that associates each word with three factors: its domain label, whether it is general or domain-specific, and its polarity. Wu and Huang [2016] train both a shared sentiment model and domain-specific models using multi-task learning. The model also integrates an automatically learned sentiment graph. In a second step, a target domain model is learned based on the target domain's similarity

to the source domains. The final classifier then is a linear combination of the target domain and the global shared model.

Neural network-based methods In more recent deep neural network approaches, Yang and Eisenstein [2015] learn embeddings for each feature for POS tagging based on an adapted SGNS. Each example is additionally associated with a vector of binary domain attributes, which is crossed with each feature. Kim et al. [2017b] use attention based on the base models' representation for an example to compute their interpolation weights. Su and Yan [2017] frame semantic parsing as paraphrasing by converting logical forms to canonical natural language utterances. They then train a sequence-to-sequence model with attention on the source domains and fine-tune it on target domain data. Chen and Cardie [2018] extend the model of Bousmalis et al. [2016] to the multi-domain case. Rather than being a binary predictor, the domain classifier now has a multinomial output. In [Ruder et al., 2017b], we propose to weight source domain models with the similarity of the corresponding source domain to the target domain. Similar to earlier methods, Guo et al. [2018] express the target model as a mixture of source domain experts. We finally learn to select instances from multiple source domains in Section 4.1.

3.4.6 Summary

In this section, we have reviewed the most common domain adaptation methods relying on representations, instance weighting and selection, and semi-supervised learning. We have also discussed how these approaches can be extended to incorporate multiple source domains. Domain adaptation typically assumes that the feature spaces rely on the same features. In NLP, this means that domains use words of the same language. In the following section, we will describe the most common method for cross-lingual adaptation where source and target domain belong to different languages.

3.5 Cross-lingual learning

* While different approaches for learning across languages are possible, we focus on learning cross-lingual word embeddings, which has recently received increasing attention. Cross-lingual representations of words enable us to reason about word meaning in multilingual contexts and are a key facilitator of cross-lingual transfer when developing natural language processing models for low-resource languages. In this section, we

*This section is adapted from: **Ruder, S.**, Vulić, I., and Søgaard, A. (2019). A Survey of Cross-lingual Word Embedding Models. To be published in *Journal of Artificial Intelligence Research*.

provide a comprehensive typology of cross-lingual word embedding models. We compare their data requirements and objective functions. The recurring theme of this section is that many of the models presented in the literature optimize for similar objectives, and that seemingly different models are often equivalent and differ only in the optimization strategy and hyper-parameters.

3.5.1 Introduction

The usefulness of monolingual word embeddings (§3.3.2.3) together with a public awareness of the digital language divide[1] and the availability of multilingual benchmarks Hovy et al. [2006], Sylak-Glassman et al. [2015], Nivre et al. [2016] has made cross-lingual transfer a popular research topic. The need to transfer lexical knowledge across languages has given rise to *cross-lingual word embedding models*, i.e., cross-lingual representations of words in a joint embedding space, as illustrated in Figure 3.9.

Many of the high-level ideas that motivate current research in this area pre-date the popular introduction of word embeddings. This includes work on learning cross-lingual word representations from seed lexica, parallel data, or document-aligned data, as well as ideas on learning from limited bilingual supervision. We direct the reader to [Ruder et al., 2019b] for an overview of the history of this area.

Cross-lingual word embeddings are appealing for two reasons: First, they enable us *to compare the meaning of words across languages*, which is key to bilingual lexicon induction, machine translation, or cross-lingual information retrieval, for example. Second, cross-lingual word embeddings *enable model transfer between languages*, e.g., between resource-rich and low-resource languages, by providing a common representation space.

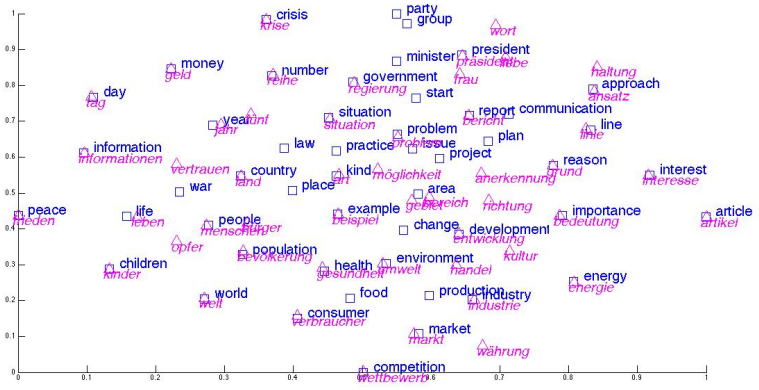


FIGURE 3.9: A shared embedding space between two languages Luong et al. [2015]

[1] E.g., `http://labs.theguardian.com/digital-language-divide/`

Many models for learning cross-lingual embeddings have been proposed in recent years. In this section, we will give a comprehensive overview of existing cross-lingual word embedding models. We will show the similarities and differences between these approaches. To facilitate this, we first introduce a common notation and terminology (§3.5.2). We then motivate and present a typology of cross-lingual embedding models (§3.5.3). The typology is based on the main differentiating aspect of cross-lingual embedding models: the nature of the data they require, in particular the type of alignment across languages (alignment of words, sentences, or documents), and whether data is assumed to be parallel or just comparable (about the same topic). The typology allows us to outline similarities and differences more concisely, but also starkly contrasts focal points of research with fruitful directions that have so far gone mostly unexplored.

We discuss the three types of alignments for learning cross-lingual word embeddings in each of the following sections:

1. word-level alignment (§3.5.4);
2. sentence-level alignment (§3.5.5);
3. and document-level alignment (§3.5.6).

We finally discuss evaluation scenarios (§3.5.7) and provide a conclusion (§3.5.8).

3.5.2 Notation and terminology

For clarity, we show all notation used throughout this survey in Table 3.1. Let $\mathbf{X}^l \in \mathbb{R}^{|V^l| \times d}$ be a word embedding matrix that is learned for the l-th of L languages where V^l is the corresponding vocabulary and d is the dimensionality of the word embeddings. We will furthermore refer to $\mathbf{X}^l_{i,:}$, that is, the word embedding of the i-th word in language l with the shorthand $\mathbf{x}^l_i$ or $\mathbf{x}_i$ if the language is unambiguous. We will refer to the word corresponding to the i-th word embedding $\mathbf{x}_i$ as w_i. Some monolingual word embedding models use a separate embedding for words that occur in the context of other words. We will use $\tilde{x}_i$ as the embedding of the i-th context word and detail its meaning in the next section. Most approaches only deal with two languages, a source language S and a target language T.

Some approaches learn a matrix $\mathbf{W}^{S \to T}$ that can be used to transform the word embedding matrix $\mathbf{X}^S$ of the source language S to that of the target language T. We will designate such a matrix by $\mathbf{W}^{S \to T} \in \mathbb{R}^{d \times d}$ and $\mathbf{W}$ if the language pairing is unambiguous. These approaches often use n source words and their translations as seed words. In addition, we will use τ as a function that maps from source words w^S_i to their translation w^T_i:

Symbol	Meaning
$\mathbf{X}$	word embedding matrix
V	vocabulary
d	word embedding dimensionality
$\mathbf{X}^l_{i,:}$ / $\mathbf{x}^l_i$ / $\mathbf{x}_i$	word embedding of the i-th word in language l
$\tilde{x}_i$	word embedding of the i-th context word
w_i	word pertaining to embedding $\mathbf{x}_i$
S	source language
T	target language
$\mathbf{W}^{S\to T}$ / $\mathbf{W}$	learned transformation matrix between space of S and T
n	number of words used as seed words for learning $\mathbf{W}$
τ	function mapping from source words to their translations
$\mathbf{C}^S$	monolingual co-occurrence matrix in language S
C	size of context window around a center word
$\mathcal{C}$	corpus of words / aligned sentences used for training
$\mathbf{A}^{S\to T}$	cross-lingual co-occurrence matrix / alignment matrix
$sent^S_i$	i-th sentence in language S
$\mathbf{y}^S_i$	representation of i-th sentence in language S
doc^S_i	i-th document in language S
$\mathbf{z}^S_i$	representation of i-th document in language S
$\underline{\mathbf{X}^S}$	$\mathbf{X}^S$ is kept fixed during optimization
$\underbrace{\mathcal{L}^1}_{1} + \underbrace{\mathcal{L}^2}_{2}$	$\mathcal{L}^1$ is optimized before $\mathcal{L}^2$

TABLE 3.1: Notation related to cross-lingual word embeddings

$\tau : V^S \to V^T$. Approaches that learn a transformation matrix are usually referred to as *offline* or *mapping* methods.

Some approaches require a monolingual word-word co-occurrence matrix $\mathbf{C}^S$ in language S. In such a matrix, every row corresponds to a word w^S_i and every column corresponds to a context word w^S_j. $\mathbf{C}^S_{ij}$ then captures the number of times word w_i occurs with context word w_j usually within a window of size C to the left and right of word w_i. In a cross-lingual context, we obtain a matrix of alignment counts $\mathbf{A}^{S\to T} \in \mathbb{R}^{|V^T|\times|V^S|}$, where each element $\mathbf{A}^{S\to T}_{ij}$ captures the number of times the $i-$th word in language T was aligned with the j-th word in language S, with each row normalized to 1.

Finally, as some approaches rely on pairs of aligned sentences, we designate $sent^S_1, \ldots, sent^S_n$ as sentences in source language S with representations $\mathbf{y}^S_1, \ldots, \mathbf{y}^S_n$, and analogously refer to their aligned sentences in the target language T as $sent^T_1, \ldots, sent^T_n$ with representations $\mathbf{y}^T_1, \ldots, \mathbf{y}^T_n$. We adopt an analogous notation for representations obtained by approaches based on alignments of documents in S and T: $doc^S_1, \ldots, doc^S_n$ and $doc^T_1, \ldots, doc^T_n$ with document representations $\mathbf{z}^S_1, \ldots, \mathbf{z}^S_n$ and $\mathbf{z}^T_1, \ldots, \mathbf{z}^T_n$ respectively.

Different notations make similar approaches appear different. Using the same notation across our survey facilitates recognizing similarities between the various cross-lingual

word embedding models. Specifically, we intend to demonstrate that cross-lingual word embedding models are trained by minimizing roughly the same objective functions, and that differences in objective are unlikely to explain the observed performance differences Levy et al. [2017].

The class of objective functions minimized by most cross-lingual word embedding methods (if not all), can be formulated as follows:

$$J = \mathcal{L}^1 + \ldots + \mathcal{L}^L + \Omega \tag{3.26}$$

where $\mathcal{L}^l$ is the monolingual loss of the l-th language and Ω is a regularization term. A similar loss was also defined by Upadhyay et al. [2016]. In this section, we will aim to condense the difference between approaches into a regularization term and detail underlying assumptions.

Importantly, how this objective function is optimized is a key differentiating factor between approaches. As the joint optimization of multiple non-convex losses is difficult, most approaches take a step-wise approach and optimize one loss at a time while keeping certain variables fixed. In most cases, we will thus use a more explicit formulation such as the one below, which makes clear in what order the losses are optimized and which variables they depend on:

$$J = \underbrace{\mathcal{L}(\mathbf{X}^S) + \mathcal{L}(\mathbf{X}^T)}_{1} + \underbrace{\Omega(\underline{\mathbf{X}^S}, \underline{\mathbf{X}^T}, \mathbf{W})}_{2} \tag{3.27}$$

The underbraces indicate that the two monolingual loss terms on the left, which depend on $\mathbf{X}^S$ and $\mathbf{X}^T$ respectively, are optimized first. Subsequently, Ω is optimized, which depends on $\underline{\mathbf{X}^S}, \underline{\mathbf{X}^T}, \mathbf{W}$. Underlined variables are kept fixed during optimization of the loss. The monolingual objectives are optimized by training one of several monolingual embedding models (§3.3.2.3) on a monolingual corpus.

3.5.3 A typology for cross-lingual word embedding models

Recent work on cross-lingual embedding models suggests that the actual choice of bilingual supervision signal—that is, the data a method requires to learn to align a cross-lingual representation space—is more important for the final model performance than the actual underlying architecture Levy et al. [2017]. Similar conclusions can be drawn from empirical work in comparing different cross-lingual embedding models Upadhyay et al. [2016]. In other words, large differences between models typically stem from their data requirements, while other fine-grained differences are artifacts of the chosen architecture, hyper-parameters, and additional tricks and fine-tuning employed.

	Parallel	Comparable
Word	Dictionaries	Images
Sentence	Translations	Captions
Document	-	Wikipedia

TABLE 3.2: Nature and alignment level of bilingual data sources required by cross-lingual embedding models.

(A) Word, par. (B) Word, comp. (C) Sentence, par. (D) Sentence, comp. (E) Doc., comp.

FIGURE 3.10: Examples for the nature and type of alignment of data sources. Par.: parallel. Comp.: comparable. Doc.: document. From left to right, word-level parallel alignment with a bilingual lexicon (3.10a); word-level comparable alignment using images from Google Image Search (3.10b); sentence-level parallel alignment with translations (3.10c); sentence-level comparable alignment using translations of image captions (3.10d); and document-level comparable alignment using similar documents (3.10e).

This directly mirrors the argument raised by Levy et al. [2015] regarding monolingual embedding models: They observe that the architecture is less important as long as the models are trained under identical conditions on the same type (and amount) of data.

We therefore base our typology on the data requirements of the cross-lingual word embedding methods. Methods differ with regard to the data they employ along the following two dimensions:

1. **Type of alignment**: Methods use different types of bilingual supervision signals (at the level of words, sentences, or documents), which introduce stronger or weaker supervision.
2. **Comparability**: Methods require either *parallel* data sources, that is, exact translations in different languages or *comparable* data that is only similar in some way.

There are three different types of possible alignments. We discuss the typical data sources for both parallel and comparable data based on the following alignment signals:

1. **Word alignment**: Most approaches use parallel word-aligned data in the form of bilingual or cross-lingual dictionary with pairs of translations between words in different languages [Mikolov et al., 2013c, Faruqui and Dyer, 2014]. Comparable word-aligned data, even though more plentiful, has been leveraged less often and

typically involves other modalities such as images [Bergsma and Van Durme, 2011, Kiela et al., 2015].

2. **Sentence alignment**: Sentence alignment requires a parallel corpus, as commonly used in MT. Methods typically use the Europarl corpus [Koehn, 2005], which consists of sentence-aligned text from the proceedings of the European parliament, and is perhaps the most common source of training data for MT models [Hermann and Blunsom, 2013, Lauly et al., 2013]. Other methods use available word-level alignment information [Zou et al., 2013, Shi et al., 2015]. There has been some work on extracting parallel data from comparable corpora [Munteanu and Marcu, 2006], but no-one has so far trained cross-lingual word embeddings on such data. Comparable data with sentence alignment may again leverage another modality, such as captions of the same image or similar images in different languages, which are not translations of each other [Calixto et al., 2017, Gella et al., 2017].
3. **Document alignment**: Parallel document-aligned data requires documents in different languages that are translations of each other. This is rare, as parallel documents typically consist of aligned sentences [Hermann and Blunsom, 2014]. Comparable document-aligned data is more common and can occur in the form of documents that are topic-aligned (e.g. Wikipedia) or class-aligned (e.g. sentiment analysis and multi-class classification datasets) [Vulić and Moens, 2013, Mogadala and Rettinger, 2016].

We summarize the different types of data required by cross-lingual embedding models along these two dimensions in Table 3.2 and provide examples for each in Figure 3.10. We present our complete typology of cross-lingual embedding models in Table 3.3, aiming to provide an exhaustive overview by classifying each model (we are aware of) into one of the corresponding model types. We also provide a more detailed overview of the monolingual objectives and regularization terms used by every approach towards the end of this section in Table 3.5.

3.5.4 Word-level alignment models

In the following, we will now discuss different types of the current generation of cross-lingual embedding models, starting with models based on word-level alignment. Among these, models based on parallel data are more common, while there are only a few approaches focusing on comparable data (§3.5.4.2).

3.5.4.1 Word alignment methods with parallel data

We distinguish three methods that use parallel word-aligned data:

	Parallel	**Comparable**
Word	Mikolov et al. (2013) Dinu et al. (2015) Lazaridou et al. (2015) Xing et al. (2015) Zhang et al. (2016) Artexte et al. (2016) Smith et al. (2016) Vulić and Korhonen (2016) Artexte et al. (2017) Hauer et al. (2017) Mrkšić et al. (2017) Faruqui and Dyer (2014) Lu et al. (2015) Ammar et al. (2016) Xiao and Guo (2014) Gouws and Søgaard (2015) Duong et al. (2016) Adams et al. (2017) Klementiev et al. (2012) Kočiský et al. (2014)	Bergsma and Van Durme (2011) Vulić et al. (2016) Kiela et al. (2015) Vulić et al. (2016) Gouws and Søgaard (2015) Duong et al. (2015)
Sentence	Zou et al. (2013) Shi et al. (2015) Gardner et al. (2015) Vyas and Carpuat (2016) Guo et al. (2015) Hermann and Blunsom (2013) Hermann and Blunsom (2014) Soyer et al. (2015) Lauly et al. (2013) Chandar et al. (2014) Gouws et al. (2015) Luong et al. (2015) Coulmance et al. (2015) Pham et al. (2015) Levy et al. (2017) Rajendran et al. (2016)	Calixto et al. (2017) Gella et al. (2017)
Document		Vulić and Moens (2016) Vulić and Moens (2013) Vulić and Moens (2014) Søgaard et al. (2015) Mogadala and Rettinger (2016)

Table 3.3: Cross-lingual embedding models ordered by data requirements.

a) **Mapping-based approaches** that first train monolingual word representations independently on large monolingual corpora and then seek to learn a transformation matrix that maps representations in one language to the representations of the other language. They learn this transformation from word alignments or bilingual dictionaries.

b) **Pseudo-multi-lingual corpora-based approaches** that use monolingual word embedding methods on automatically constructed (or corrupted) corpora that contain words from both the source and the target language.

c) **Joint methods** that take parallel text as input and minimize the source and target language monolingual losses jointly with the cross-lingual regularization term.

We will show that these approaches are mostly equivalent, except for the order in which the loss terms are optimized.

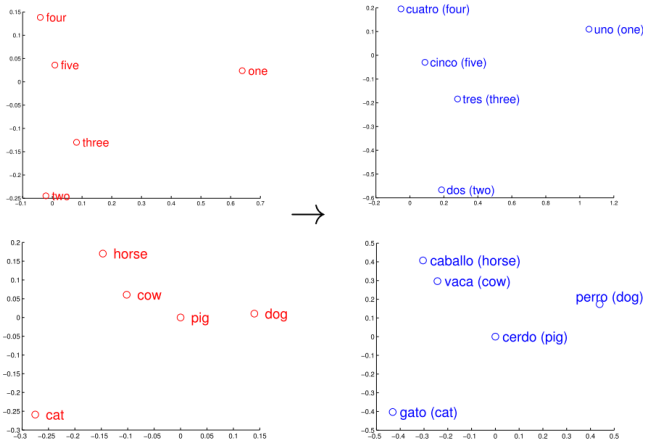


FIGURE 3.11: Similar geometric relations between numbers and animals in English and Spanish [Mikolov et al., 2013c]. Embeddings are projected into two dimensions using PCA and spaces have been rotated to emphasize the correspondence.

Mapping-based approaches

Minimizing mean squared error Mikolov et al. [2013c] notice that the geometric relations that hold between words are similar across languages: for instance, numbers and animals in English show a similar geometric constellation as their Spanish counterparts, as illustrated in Figure 3.11. This suggests that it is possible to transform the vector space of a source language S to the vector space of the target language t by learning a linear projection with a transformation matrix $\mathbf{W}$.

They use the $n = 5000$ most frequent words from the source language $w_1^S, \ldots, w_n^S$ and their translations $w_1^T, \ldots, w_n^T$ as seed words. They then learn $\mathbf{W}$ using stochastic gradient descent by minimising the MSE (§2.2.2) between the previously learned monolingual

representations of the source seed word $\mathbf{x}_i^S$ that is transformed using $\mathbf{W}$ and its translation $\mathbf{x}_i^T$ in the bilingual dictionary:

$$\Omega_{\text{MSE}} = \sum_{i=1}^{n} \|\mathbf{W}\mathbf{x}_i^S - \mathbf{x}_i^T\|^2 \tag{3.28}$$

This can also be written in matrix form as minimizing the squared Frobenius norm of the residual matrix:

$$\Omega_{\text{MSE}} = \|\mathbf{W}\mathbf{X}^S - \mathbf{X}^T\|_F^2 \tag{3.29}$$

where $\mathbf{X}^S$ and $\mathbf{X}^T$ are the embedding matrices of the seed words in the source and target language respectively. In this form, it is also known as the *orthogonal Procrustes problem* or *Procrustes Analysis* and has an analytical solution based on SVD: $\mathbf{X}^+ = (\mathbf{X}^{S\top}\mathbf{X}^S)^{-1}\mathbf{X}^{S\top}$ as $\mathbf{W} = \mathbf{X}^+\mathbf{X}^T$ [Artetxe et al., 2016].

In our notation, the MSE mapping approach can be seen as optimizing the following objective:

$$J = \underbrace{\mathcal{L}_{\text{SGNS}}(\mathbf{X}^S) + \mathcal{L}_{\text{SGNS}}(\mathbf{X}^T)}_{1} + \underbrace{\Omega_{\text{MSE}}(\underline{\mathbf{X}^S}, \underline{\mathbf{X}^T}, \mathbf{W})}_{2} \tag{3.30}$$

First, each of the monolingual losses is optimized independently. Second, the regularization term Ω_{MSE} is optimized while keeping the induced monolingual embeddings fixed. Several extensions to the basic mapping model as framed by Mikolov et al. [2013c] have been proposed.

Max-margin with intruders Dinu et al. [2015] discover that using MSE as the subobjecive for learning a projection matrix leads to the issue of *hubness*: some words tend to appear as nearest neighbours of many other words (i.e., they are hubs). As translations are typically generated by choosing the nearest neighbour of a source embedding, hubs reduce the quality of the embedding space. They propose a globally corrected neighbour retrieval method to overcome this issue. Lazaridou et al. [2015] show that optimizing a max-margin loss (§2.2.2) instead of MSE reduces hubness and consequently improves performance. The max-margin loss assigns a higher cosine similarity to word pairs that are translations of each other ($\mathbf{x}_i^S, \mathbf{x}_i^T$; first term below) than random word pairs ($\mathbf{x}_i^S, \mathbf{x}_j^T$; second term):

$$\Omega_{\text{MM}} = \sum_{i=1}^{n}\sum_{j\neq i}^{k} \max\{0, \gamma - \cos(\mathbf{W}\mathbf{x}_i^S, \mathbf{x}_i^T) + \cos(\mathbf{W}\mathbf{x}_i^S, \mathbf{x}_j^T)\} \tag{3.31}$$

The choice of negative examples is crucial. Dinu et al. [2015] propose to select negative examples that are more informative by being near the current projected vector $\mathbf{W}\mathbf{x}_i^S$ but far from the actual translation vector $\mathbf{x}_i^T$. Unlike random intruders, such intelligently

chosen intruders help the model identify training instances where the model considerably fails to approximate the target function.

Smith et al. [2017] propose a similar solution to the hubness issue in the framework of mapping-based approaches: they invert the softmax used for finding the translation of a word at test time and normalize the probability over source words rather than target words.

Normalization and orthogonality constraint Xing et al. [2015] argue that there is a mismatch between the comparison function used for training word embeddings with SGNS, that is, the dot product and the function used for evaluation, which is cosine similarity. They suggest to normalize word embeddings to be unit length to address this discrepancy. In order to preserve unit length after mapping, they propose, in addition, to constrain $\mathbf{W}$ to be orthogonal: $\mathbf{W}^\top\mathbf{W} = \mathbf{I}$. The exact solution under this constraint is $\mathbf{W} = \mathbf{V}\mathbf{U}^\top$ and can be efficiently computed in linear time with respect to the vocabulary size using SVD where $\mathbf{X}^{T\top}\mathbf{X}^S = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^\top$. An orthogonality constraint is also used by Zhang et al. [2016b] for learning cross-lingual embeddings for POS tagging.

Artetxe et al. [2016] further demonstrate the similarity between different approaches by showing that the mapping model variant of Xing et al. [2015] optimizes the same loss as Mikolov et al. [2013c] with an orthogonality constraint and unit vectors. In addition, they empirically show that orthogonality is more important for performance than length normalization. They also propose dimension-wise mean centering in order to capture the intuition that two randomly selected words are generally expected not to be similar and their cosine similarity should thus be zero. Smith et al. [2017] also learn a linear transformation with an orthogonality constraint and use identical character strings as their seed lexicon.

Using highly reliable seed lexicon entries The previous mapping approaches used a bilingual dictionary as an inherent component of their model, but did not pay much attention to the quality of the dictionary entries, using either automatic translations of frequent words or word alignments of all words. Vulić and Korhonen [2016] emphasize the role of the seed lexicon that is used for learning the projection matrix. They propose a hybrid model that initially learns a first shared bilingual embedding space based on an existing cross-lingual embedding model. They then use this initial vector space to obtain translations for a list of frequent source words by projecting them into the space and using the nearest neighbor in the target language as translation and use these translation pairs as seed words for learning a mapping. In addition, they propose a symmetry

constraint: it enforces that words are included in the seed lexicon if and only if their projections are nearest neighbors of each other in the first embedding space.

Bootstrapping from few seed entries Recently, there have been initiatives towards enabling embedding induction using only a small number of seed translation pairs. The core idea behind these bootstrapping approaches is to start from a few seed words initially, which are then iteratively expanded. Artetxe et al. [2017] propose a mapping model that relies only on a small set of shared words (e.g., identically spelled words or only shared numbers) to seed the procedure. The model has multiple bootstrapping rounds where it gradually adds more and more bilingual translation pairs to the original seed lexicon and refines it. In Section 5.2, we cast this method as a latent variable method and propose a novel latent variable model with iterative refinement [Ruder et al., 2018].

Smith et al. [2017] and Hauer et al. [2017] propose a method that creates seed lexicons by identifying cognates in the vocabularies of related languages. In contrast to Mikolov et al. [2013c], they learn not only a transformation matrix $\mathbf{W}^{S \to T}$ that transforms embeddings of language S to embeddings of language T, but also a matrix $\mathbf{W}^{T \to S}$ that transforms embeddings in the opposite direction. Starting from a small set of automatically extracted seed translation pairs, they iteratively expand the size of the lexicon.

The bootstrapping idea is conceptually similar to the work of Peirsman and Padó [2011] and Vulić and Moens [2013], with the difference that earlier approaches were developed within the traditional cross-lingual distributional framework (mapping vectors into the count-based space using a seed lexicon).

Cross-lingual embeddings via retro-fitting Learning a mapping between monolingual embedding spaces can also be framed as retro-fitting [Faruqui et al., 2015], which is used to inject knowledge from semantic lexicons into pretrained word embeddings. Retro-fitting creates a new word embedding matrix $\hat{\mathbf{X}}$ whose embeddings $\hat{\mathbf{x}}_i$ are both close to the corresponding learned monolingual word embeddings $\mathbf{x}_i$ as well as close to neighbors $\mathbf{x}_j$ in a knowledge graph:

$$\Omega_{\text{retro}} = \sum_{i=1}^{|V|} [\alpha_i \|\hat{\mathbf{x}}_i - \mathbf{x}_i\|^2 + \sum_{(i,j) \in E} \beta_{ij} \|\hat{\mathbf{x}}_i - \mathbf{x}_j\|^2] \tag{3.32}$$

where E is the set of edges in the knowledge graph and α and β control the strength of the contribution of each term.

Mrkšić et al. [2017] similarly derive cross-lingual synonymy and antonymy constraints from BabelNet. They then use these constraints to bring the monolingual vector spaces of two

different languages together into a shared embedding space. Such retro-fitting approaches employ a max-margin loss with a careful selection of intruders, similar to [Lazaridou et al., 2015]. In contrast to previous work, retro-fitting approaches use constraints on each word rather than a translation matrix $\mathbf{W}$ to arrive at a new cross-lingual vector space. While these constraints can capture relations that are more complex than a linear transformation, they are limited to words that are contained in the semantic lexicons.

CCA-based mapping Haghighi et al. [2008] are the first to use CCA for learning translation lexicons for the words of different languages. Faruqui and Dyer [2014] later apply CCA to project words from two languages into a shared embedding space. Whereas linear projection only learns one transformation matrix $\mathbf{W}^{S\rightarrow T}$ to project the embedding space of a source language into the space of a target language, CCA learns a transformation matrix for the source and target language $\mathbf{W}^{S\rightarrow}$ and $\mathbf{W}^{T\rightarrow}$ respectively to project them into a new joint space that is different from both the space of S and of T. CCA minimizes the following:

$$\Omega_{\text{CCA}} = -\sum_{i=1}^{n} \rho(\mathbf{W}^{S\rightarrow}\mathbf{x}_i^S, \mathbf{W}^{T\rightarrow}\mathbf{x}_i^T) \tag{3.33}$$

where ρ is the correlation (§2.1.1). We can write their objective as the following:

$$J = \underbrace{\mathcal{L}_{\text{LSA}}(\mathbf{X}^S) + \mathcal{L}_{\text{LSA}}(\mathbf{X}^T)}_{1} + \underbrace{\Omega_{\text{CCA}}(\underline{\mathbf{X}^S}, \underline{\mathbf{X}^T}, \mathbf{W}^{S\rightarrow}, \mathbf{W}^{T\rightarrow})}_{2} \tag{3.34}$$

As CCA sorts the projection matrices $\mathbf{W}^{S\rightarrow}$ and $\mathbf{W}^{t\rightarrow}$ in descending order, Faruqui and Dyer [2014] find that using the 80% of the projection vectors with the highest correlation generally yields the highest performance and that CCA helps to separate synonyms and antonyms in the source language.

Lu et al. [2015] extend Bilingual CCA to Deep Bilingual CCA by introducing non-linearity in the mapping process: they train two *deep* neural networks to maximize the correlation between the projections of both monolingual embedding spaces. Finally, Artetxe et al. [2016] show that their objective, built on top of the original or "standard" Mikolov-style mapping idea, and which uses dimension-wise mean centering is directly related to bilingual CCA. The only fundamental difference is that the CCA-based model does not guarantee monolingual invariance, while this property is enforced in the model of Artetxe et al. [2016].

In a similar vein to CCA, we show in [Kementchedjhieva et al., 2018] that projecting embeddings into a third latent space is beneficial. Specifically, we extend Procrustes

Analysis to Generalized Procrustes Analysis [Gower, 1975]. Generalized Procrustes Analysis outperforms Procrustes Analysis consistently and also allows us to incorporate additional support languages in low-resource scenarios.

Word-level approaches based on pseudo-bilingual corpora

Rather than learning a mapping between the source and the target language, some approaches use the word-level alignment of a seed bilingual dictionary to construct a pseudo-bilingual corpus by randomly replacing words in a source language corpus with their translations. Xiao and Guo [2014] propose the first such method. They first construct a seed bilingual dictionary by translating all words that appear in the source language corpus into the target language using Wiktionary, filtering polysemous words as well as translations that do not appear in the target language corpus. From this seed dictionary, they create a joint vocabulary, in which each translation pair occupies the same vector representation. They train this model using the max-margin loss by feeding it context windows of both the source and target language corpus.

Other approaches explicitly create a pseudo-bilingual corpus: Gouws and Søgaard [2015] concatenate the source and target language corpus and replace each word that is part of a translation pair with its translation equivalent with a probability of $\frac{1}{2k_T}$, where k_T is the total number of possible translation equivalents for a word, and train CBOW on this corpus. Ammar et al. [2016b] extend this approach to multiple languages: Using bilingual dictionaries, they determine clusters of synonymous words in different languages. They then concatenate the monolingual corpora of different languages and replace tokens in the same cluster with the cluster ID. Finally, they train SGNS on the concatenated corpus.

Duong et al. [2016] propose a similar approach. Instead of randomly replacing every word in the corpus with its translation, they replace each center word with a translation on-the-fly during CBOW training. In addition, they handle polysemy explicitly by proposing an EM-inspired method that chooses as replacement the translation w_i^T whose representation is most similar to the sum of the source word representation $\mathbf{x}_i^S$ and the sum of the context embeddings $\mathbf{x}_s^S$ (see Equation 3.9):

$$w_i^T = \text{argmax}_{w' \in \tau(w_i)} \cos(\mathbf{x}_i + \mathbf{x}_s^S, \mathbf{x}') \tag{3.35}$$

They jointly learn to predict both the words and their appropriate translations using PanLex as the seed bilingual dictionary. PanLex covers around 1,300 language with about 12 million expressions. Consequently, translations are high coverage but often

noisy. Adams et al. [2017] use the same approach for pretraining cross-lingual word embeddings for low-resource language modeling.

Joint models

While the previous approaches either optimize a set of monolingual losses and then the cross-lingual regularization term (mapping-based approaches) or optimize a monolingual loss and implicitly—via data manipulation—a cross-lingual regularization term, joint models optimize monolingual and cross-lingual objectives at the same time jointly. In what follows, we discuss a few illustrative example models which sparked this sub-line of research.

Bilingual language model Klementiev et al. [2012b] cast learning cross-lingual representations as a multi-task learning problem. They jointly optimize a source language and target language model together with a cross-lingual regularization term that encourages words that are often aligned with each other in a parallel corpus to be similar. The monolingual objective is the classic LM objective of minimizing the negative log likelihood of the current word w_i given its C previous context words:

$$\mathcal{L} = -\log P(w_i \mid w_{i-C+1:i-1}) \tag{3.36}$$

For the cross-lingual regularization term, they first obtain an alignment matrix $\mathbf{A}^{S \to T}$ that indicates how often each source language word was aligned with each target language word from parallel data such as the Europarl corpus [Koehn, 2009]. They then blow this matrix up to a symmetrical double-size version $\mathbf{A} \in \mathbb{R}^{|V| \times |V|}$ where $V = V^S \cup V^T$. They furthermore concatenate the representations of all source and target words $\mathbf{X} = [\mathbf{x}_1^S, \ldots, \mathbf{x}_{|V^S|}^S, \mathbf{x}_1^T, \ldots, \mathbf{x}_{|V^T|}^T] \in \mathbb{R}^{d \times |V|}$. The cross-lingual regularization term then encourages the representations of source and target language words that are often aligned in $\mathbf{A}$ to be similar:

$$\Omega_s = \frac{1}{2}\mathbf{X}^\top(\mathbf{A} \otimes \mathbf{I}_d)\mathbf{X} \tag{3.37}$$

where $\mathbf{I}_d \in \mathbb{R}^{d \times d}$ is the identity matrix and $\otimes$ is the Kronecker product, which intuitively "blows up" each element of $\mathbf{A}$ to the size of a word embedding $\mathbf{x} \in \mathbb{R}^d$. Equation 3.37 is a weighted (by word alignment scores) average of inner products, and hence, for unit length normalized embeddings, equivalent to approaches that maximize the sum of the cosine similarities of aligned word pairs. Using $\mathbf{A} \otimes \mathbf{I}$ to encode interaction is borrowed from linear multi-task learning models [Cavallanti et al., 2010] where $\mathbf{A}$ encodes task

relatedness (§3.2.5.2). The complete objective is the following:

$$J = \mathcal{L}(\mathbf{X}_s) + \mathcal{L}(\mathbf{X}_t) + \Omega(\underline{\mathbf{A}}, \mathbf{X}) \tag{3.38}$$

Joint learning of word embeddings and word alignments Kočiský et al. [2014] simultaneously learn word embeddings and word-level alignments using a distributed version of FastAlign [Dyer et al., 2013] together with a language model.[2] Similar to other bilingual approaches, they use the word in the source language sentence of an aligned sentence pair to predict the word in the target language sentence.

They replace the standard multinomial translation probability of FastAlign with an energy function that tries to bring the representation of a target word w_i^T close to the sum of the context words around the word w_i^S in the source sentence:

$$E(w_i^S, w_i^T,) = -(\sum_{j=-C}^{C} \mathbf{x}_{i+j}^S{}^\top \mathbf{T}) \mathbf{x}_i^T - \mathbf{b}^\top \mathbf{x}_i^T - b_{w_i^T} \tag{3.39}$$

where $\mathbf{x}_{i+j}^S$ and $\mathbf{x}_i^T$ are the representations of source word w_{i+j}^S and target word w_i^T respectively, $\mathbf{T} \in \mathbb{R}^{d \times d}$ is a projection matrix, and $\mathbf{b} \in \mathbb{R}^d$ and $b_{w_i^T} \in \mathbb{R}$ are representation and word biases respectively. The authors speed up training by using a class factorization strategy similar to the hierarchical softmax and predict frequency-based class representations instead of word representations. For training, they use EM but fix the alignment counts of the E-step learned by FastAlign that was initially trained for 5 epochs. They then optimize the word embeddings in the M-step only. Note that this model is conceptually very similar to bilingual models that discard word-level alignment information and learn solely on the basis of sentence-aligned information, which we discuss in Section 3.5.5.

3.5.4.2 Word alignment methods with comparable data

All previous methods required word-level *parallel* data. We categorize methods that employ word-level alignment with *comparable* data into two types:

a) **Language grounding models** anchor language in images and use image features to obtain information with regard to the similarity of words in different languages.
b) **Comparable feature models** that rely on the comparability of some other features. The main feature that has been explored in this context is part-of-speech (POS) tag equivalence.

[2] FastAlign is a fast and effective variant of IBM Model 2 [Brown et al., 1993c].

Grounding language in images Most methods employing word-aligned comparable data ground words from different languages in image data. The idea in all of these approaches is to use the image space as the shared cross-lingual signals. For instance, bicycles always look like bicycles even if they are referred to as "fiets", "Fahrrad", "bicikl", "bicicletta", or "velo". A set of images for each word is typically retrieved using Google Image Search. Bergsma and Van Durme [2011] calculate a similarity score for a pair of words based on the visual similarity of their associated image sets. They propose two strategies to calculate the cosine similarity between the color and SIFT features of two image sets, either taking the average or the maximum of the maximum similarity scores. Kiela et al. [2015] propose to do the same but use CNN-based image features. Vulić et al. [2016] in addition propose to combine image and word representations either by interpolating and concatenating them or by interpolating the linguistic and visual similarity scores.

A similar idea of grounding language for learning multimodal multilingual representations can be applied for sensory signals beyond vision, e.g. auditive or olfactory signals [Kiela and Clark, 2015]. This line of work, however, is currently under-explored. Moreover, it seems that signals from other modalities are typically more useful as an additional source of information to complement the uni-modal signals from text, rather than using other modalities as the single source of information.

POS tag equivalence Other approaches rely on comparability between certain features of a word, such as its part-of-speech tag. Gouws and Søgaard [2015] create a pseudo-cross-lingual corpus by replacing words based on part-of-speech equivalence, that is, words with the same part-of-speech in different languages are replaced with one another. Instead of using the POS tags of the source and target words as a bridge between two languages, we can also use the POS tags of their contexts. This makes strong assumptions about the word orders in the two languages, and their similarity, but Duong et al. [2015a] use this to obtain cross-lingual word embeddings for several language pairs. They use POS tags as context features and run SGNS on the concatenation of two monolingual corpora. Note that under the overly simplistic assumptions that all instances of a part-of-speech have the same distribution and each word belongs to a single part-of-speech class, this approach is equivalent to the pseudo-cross-lingual corpus approach described before.

3.5.5 Sentence-level alignment models

Research on MT has produced large amounts of sentence-aligned parallel data for European languages, which has led to much work focusing on learning cross-lingual

representations from sentence-aligned parallel data. For low-resource languages or new domains, sentence-aligned parallel data is more expensive to obtain than word-aligned data as it requires fine-grained supervision. Only recently have methods started leveraging sentence-aligned comparable data (§3.5.5.2).

3.5.5.1 Sentence alignment methods with parallel data

Methods leveraging sentence-aligned parallel data are generally extensions of successful monolingual models. We have detected four main types:

a) **Word-alignment based matrix factorization approaches** (§3.5.5.1) apply matrix factorization techniques to the bilingual setting and typically require additional word alignment information.
b) **Compositional sentence models** use word representations to construct sentence representations of aligned sentences, which are trained to be close to each other.
c) **Bilingual autoencoder models** reconstruct source and target sentences using an autoencoder.
d) **Bilingual skip-gram models** use the skip-gram objective to predict words in both source and target sentences.

Word-alignment based matrix factorization

Several methods directly leverage the information contained in an alignment matrix $\mathbf{A}^{S \to T}$ between source language S and target language T respectively. $\mathbf{A}^{S \to T}$ is generally automatically derived from sentence-aligned parallel text using an unsupervised word alignment model such as FastAlign [Dyer et al., 2013]. $\mathbf{A}_{ij}^{S \to T}$ captures the number of times the i-th word in language T was aligned with the j-th word in language S, with each row normalized to 1. The intuition is that if a word in the source language is only aligned with one word in the target language, then those words should have the same representation. If the target word is aligned with more than one source word, then its representation should be a combination of the representations of its aligned words. Zou et al. [2013] represent the embeddings $\mathbf{X}^S$ in the target language as the product of the source embeddings $\mathbf{X}^S$ with the corresponding alignment matrix $\mathbf{A}^{S \to T}$. They then minimize the squared difference between these two terms in both directions:

$$\Omega_{S \to T} = ||\mathbf{X}^T - \mathbf{A}^{S \to T}\mathbf{X}^S||^2 \tag{3.40}$$

Note that $\Omega_{S \to T}$ can be seen as a variant of Ω_{MSE}, which incorporates soft weights from alignments. In contrast to mapping-based approaches, the alignment matrix, which

transforms source to target embeddings, is fixed in this case, while the corresponding source embeddings $\mathbf{X}^S$ are learned:

$$J = \underbrace{\mathcal{L}_{\text{MM}}(\mathbf{X}^T)}_{1} + \underbrace{\Omega_{S \to T}(\underline{\mathbf{X}^T}, \underline{\mathbf{A}^{S \to T}}, \mathbf{X}^S)}_{2} \tag{3.41}$$

Shi et al. [2015] also take into account monolingual data by placing cross-lingual constraints on the monolingual representations and propose two alignment-based cross-lingual regularization objectives. The first one treats the alignment matrix $\mathbf{A}^{S \to T}$ as a cross-lingual co-occurrence matrix and factorizes it using the GloVe objective. The second one is similar to the objective by Zou et al. [2013] and minimizes the squared distance of the representations of words in two languages weighted by their alignment probabilities.

Gardner et al. [2015] extend LSA as translation-invariant LSA to learn cross-lingual word embeddings. They factorize a multilingual co-occurrence matrix with the restriction that it should be invariant to translation, i.e., it should stay the same if multiplied with the respective word or context dictionary.

Vyas and Carpuat [2016] propose another method based on matrix factorization that enables learning sparse cross-lingual embeddings. As the sparse cross-lingual embeddings are different from the monolingual embeddings $\mathbf{X}$, we diverge slightly from our notation and designate them as $\mathbf{S}$. They propose two constraints: The first constraint induces monolingual sparse representations from pre-trained monolingual embedding matrices $\mathbf{X}^S$ and $\mathbf{X}^T$ by factorizing each embedding matrix $\mathbf{X}$ into two matrices $\mathbf{S}$ and $\mathbf{E}$ with an additional ℓ_1 constraint on $\mathbf{S}$ for sparsity:

$$\mathcal{L} = \sum_{i=1}^{|V|} \|\mathbf{S}_i \mathbf{E}^\top - \mathbf{X}_i\|_2^2 + \lambda \|\mathbf{S}_i\|_1 \tag{3.42}$$

To learn bilingual embeddings, they add another constraint based on the alignment matrix $\mathbf{A}^{S \to T}$ that minimizes the ℓ_2 reconstruction error between words that were strongly aligned to each other in a parallel corpus:

$$\Omega = \sum_{i=1}^{|V^S|} \sum_{j=1}^{|V^T|} \frac{1}{2} \lambda_x \mathbf{A}_{ij}^{S \to T} \|\mathbf{S}_i^S - \mathbf{S}_j^T\|_2^2 \tag{3.43}$$

The complete optimization then consists of first pre-training monolingual embeddings $\mathbf{X}^S$ and $\mathbf{X}^T$ with GloVe and in a second step factorizing the monolingual embeddings with the cross-lingual constraint to induce cross-lingual sparse representations $\mathbf{S}^S$ and

$\mathbf{S}^T$:

$$J = \underbrace{\mathcal{L}_{\text{GloVe}}(\mathbf{X}^S) + \mathcal{L}_{\text{GloVe}}(\mathbf{X}^T)}_{1} + \underbrace{\mathcal{L}(\underline{\mathbf{X}^S}, \mathbf{S}^S, \mathbf{E}^S) + \mathcal{L}(\underline{\mathbf{X}^T}, \mathbf{S}^T, \mathbf{E}^T) + \Omega(\underline{\mathbf{A}^{S \to T}}, \mathbf{S}^S, \mathbf{S}^T)}_{2} \tag{3.44}$$

Guo et al. [2015] similarly create a target language word embedding $\mathbf{x}_i^T$ of a source word w_i^S by taking the average of the embeddings of its translations $\tau(w_i^S)$ weighted by their alignment probability with the source word:

$$\mathbf{x}_i^T = \sum_{w_j^T \in \tau(w_i^S)} \frac{\mathbf{A}_{i,j}}{\mathbf{A}_{i,\cdot}} \cdot \mathbf{x}_j^T \tag{3.45}$$

They propagate alignments to OOV words using edit distance as an approximation for morphological similarity and set the target word embedding $\mathbf{x}_k^T$ of an OOV source word w_k^S as the average of the projected vectors of source words that are similar to it based on the edit distance measure:

$$\mathbf{x}_k^T = \frac{1}{|E_k|} \sum_{w^S \in E_k} \mathbf{x}^T \tag{3.46}$$

where $\mathbf{x}^T$ is the target language word embedding of a source word w^S as created above, $E_k = \{w^S \mid EditDist(w_k^S, w^S) \leq \chi\}$, and χ is set empirically to 1.

Compositional sentence model

Hermann and Blunsom [2013] train a model to bring the sentence representations of aligned sentences $sent^S$ and $sent^T$ in source and target language S and T respectively close to each other. The representation $\mathbf{y}^S$ of sentence $sent^S$ in language S is the sum of the embeddings of its words:

$$\mathbf{y}^S = \sum_{i=1}^{|sent^S|} \mathbf{x}_i^S \tag{3.47}$$

They seek to minimize the distance between aligned sentences $sent^S$ and $sent^T$:

$$E_{dist}(sent^S, sent^T) = \|\mathbf{y}^S - \mathbf{y}^T\|^2 \tag{3.48}$$

They optimize this distance using the max-margin loss by learning to bring aligned sentences closer together than randomly sampled negative examples:

$$\mathcal{L} = \sum_{(sent^S, sent^T) \in \mathcal{C}} \sum_{i=1}^{k} \max(0, 1 + E_{dist}(sent^S, sent^T) - E_{dist}(sent^S, s_i^T)) \tag{3.49}$$

where k is the number of negative examples. In addition, they use an ℓ_2 regularization term for each language $\Omega = \frac{\lambda}{2}\|\mathbf{X}\|^2$ so that the final loss they optimize is the following:

$$J = \mathcal{L}(\mathbf{X}^S, \mathbf{X}^T) + \Omega(\mathbf{X}^S) + \Omega(\mathbf{X}^T) \tag{3.50}$$

Note that compared to most previous approaches, there is no dedicated monolingual objective and all loss terms are optimized jointly. Note that in this case, the ℓ_2 norm is applied to representations $\mathbf{X}$, which are computed as the difference of sentence representations.

This regularization term *approximates* minimizing the mean squared error between the pair-wise interacting source and target words in a way similar to [Gouws et al., 2015].

Hermann and Blunsom [2014] extend this approach to documents, by applying the composition and objective function recursively to compose sentences into documents. They additionally propose a non-linear composition function based on bigram pairs, which outperforms simple addition on large training datasets, but underperforms it on smaller data:

$$\mathbf{y} = \sum_{i=1}^{n} \tanh(\mathbf{x}_{i-1} + \mathbf{x}_i) \tag{3.51}$$

Soyer et al. [2015] augment this model with a monolingual objective that operates on the phrase level. The objective uses the max-margin loss and is based on the assumption that phrases are typically more similar to their sub-phrases than to randomly sampled phrases:

$$\mathcal{L} = [\max(0, m + \|\mathbf{a}_o - \mathbf{a}_i\|^2 - \|\mathbf{a}_o - \mathbf{b}_n\|^2) + \|\mathbf{a}_o - \mathbf{a}_i\|^2]\frac{n_i}{n_o} \tag{3.52}$$

where m is a margin, $\mathbf{a}_o$ is a phrase of length n_o sampled from a sentence, $\mathbf{a}_i$ is a sub-phrase of $\mathbf{a}_o$ of length n_i, and $\mathbf{b}_n$ is a phrase sampled from a random sentence. The additional loss terms are meant to reduce the influence of the margin as a hyper-parameter and to compensate for the differences in phrase and sub-phrase length.

Bilingual autoencoder

Instead of minimizing the distance between two sentence representations in different languages, Lauly et al. [2013] aim to reconstruct the target sentence from the original source sentence. Analogously to Hermann and Blunsom [2013], they also encode a sentence as the sum of its word embeddings. They then train an auto-encoder with language-specific encoder and decoder layers and hierarchical softmax to reconstruct from each sentence the sentence itself and its translation. In this case, the encoder parameters are the word embedding matrices $\mathbf{X}^S$ and $\mathbf{X}^T$, while the decoder parameters are transformation matrices that project the encoded representation to the output language space. The loss they optimize can be written as follows:

$$J = \mathcal{L}_{\text{AUTO}}^{S \to S} + \mathcal{L}_{\text{AUTO}}^{T \to T} + \mathcal{L}_{\text{AUTO}}^{S \to T} + \mathcal{L}_{\text{AUTO}}^{T \to S} \tag{3.53}$$

where $\mathcal{L}_{\text{AUTO}}^{S \to T}$ denotes the loss for reconstructing from a sentence in language S to a sentence in language T. Aligned sentences are sampled from parallel text and all losses are optimized jointly.

Chandar et al. [2014] use a binary BOW instead of the hierarchical softmax. To address the increase in complexity due to the higher dimensionality of the BOW, they propose to merge the bags-of-words in a mini-batch into a single BOW and to perform updates based on this merged bag-of-words. They also add a term to the objective function that encourages correlation between the source and target sentence representations by summing the scalar correlations between all dimensions of the two vectors.

Bilingual skip-gram

Several authors propose extensions of the monolingual SGNS model to learn cross-lingual embeddings. We show their similarities and differences in Table 3.4. All of these models jointly optimize monolingual SGNS losses for each language together with one more cross-lingual regularization terms:

$$J = \mathcal{L}_{\text{SGNS}}^{S} + \mathcal{L}_{\text{SGNS}}^{T} + \Omega \tag{3.54}$$

Another commonality is that these models do not require word alignments of aligned sentences. Instead, they make different assumptions about the alignment of the data. Bilingual Bag-of-Words without Word Alignments [BilBOWA; Gouws et al., 2015] assumes each word in a source sentence is aligned with *every* word in the target sentence. If we knew the word alignments, we would try to bring the embeddings of aligned words in source and target sentences close together. Instead, under a uniform alignment model

Model	Alignment model	Monolingual losses	Cross-lingual regularizer
BilBOWA Gouws et al. [2015]	Uniform	$\mathcal{L}^S_{\text{SGNS}} + \mathcal{L}^T_{\text{SGNS}}$	Ω_{BILBOWA}
Trans-gram Coulmance et al. [2015]	Uniform	$\mathcal{L}^S_{\text{SGNS}} + \mathcal{L}^T_{\text{SGNS}}$	$\Omega^{S\to t}_{\text{SGNS}} + \Omega^{t\to s}_{\text{SGNS}}$
BiSkip Luong et al. [2015]	Monotonic	$\mathcal{L}^S_{\text{SGNS}} + \mathcal{L}^T_{\text{SGNS}}$	$\Omega^{S\to t}_{\text{SGNS}} + \Omega^{t\to s}_{\text{SGNS}}$

TABLE 3.4: A comparison of similarities and differences of the three bilingual skip-gram variants.

which matches the intuition behind the simplest (lexical) word alignment IBM Model 1 [Brown et al., 1993a], we try to bring the *average* alignment close together. In other words, we use the means of the word embeddings in a sentence as the sentence representations $\mathbf{y}$ and seek to minimize the distance between aligned sentence representations:

$$\mathbf{y}^S = \frac{1}{|sent^S|} \sum_{i=1}^{|sent^S|} \mathbf{x}^S_i \tag{3.55}$$

$$\Omega_{\text{BILBOWA}} = \sum_{(sent^S, sent^T)\in\mathcal{C}} \|\mathbf{y}^S - \mathbf{y}^T\|^2 \tag{3.56}$$

Note that this regularization term is very similar to the objective used in the compositional sentence model [Hermann and Blunsom, 2013] (Equations 3.47 and 3.48); the only difference is that we use the mean rather than the sum of word embeddings as sentence representations.

Trans-gram [Coulmance et al., 2015] also assumes uniform alignment but uses the SGNS objective as cross-lingual regularization term. In the cross-lingual SGNS setting, we aim to predict words in the aligned target language sentence based on words in the source sentence. Under uniform alignment, we aim to predict *all* words in the target sentence based on each word in the source sentence:

$$\Omega^{S\to T}_{\text{SGNS}} = -\sum_{(sent^S, sent^T)\in\mathcal{C}} \frac{1}{|sent^S|} \sum_{t=1}^{|sent^S|} \sum_{j=1}^{|sent^T|} \log P(w_{t+j} \mid w_t) \tag{3.57}$$

where $P(w_{t+j} \mid w_t)$ is computed via negative sampling (see Equation 3.8).

BiSkip [Luong et al., 2015] uses the same cross-lingual regularization terms as Trans-gram, but only aims to predict monotonically aligned target language words: Each source word at position i in the source sentence $sent^S$ is aligned to the target word at position $i \cdot \frac{|sent^S|}{|sent^T|}$ in the target sentence $sent^T$. In practice, all bilingual skip-gram models are trained by sampling a pair of aligned sentences from a parallel corpus and the respective loss terms minimizing for the source and target language sentence .

In a similar vein, Pham et al. [2015] propose an extension of paragraph vectors [Le and Mikolov, 2014] to the multilingual setting by forcing aligned sentences of different languages to share the same vector representation.

Other sentence-level approaches Levy et al. [2017] use another sentence-level bilingual signal: IDs of the aligned sentence pairs in a parallel corpus. Their model provides a strong baseline for cross-lingual embeddings that is inspired by the Dice aligner commonly used for producing word alignments for MT. Observing that sentence IDs are already a powerful bilingual signal, they propose to apply SGNS to the word-sentence ID matrix. They show that this method can be seen as a generalization of the Dice Coefficient.

Rajendran et al. [2016] propose a method that exploits the idea of using pivot languages, also tackled in previous work, e.g., [Shezaf and Rappoport, 2010]. The model requires parallel data between each language and a pivot language and is able to learn a shared embedding space for two languages without any direct alignment signals as the alignment is implicitly learned via their alignment with the pivot language. The model optimizes a correlation term with neural network encoders and decoders that is similar to the objective of the CCA-based approaches [Faruqui and Dyer, 2014, Lu et al., 2015].

3.5.5.2 Sentence alignment methods with comparable data

Grounding language in images Similarly to approaches based on word-level aligned comparable data, methods that learn cross-lingual representations using sentence alignment with comparable data do so by associating sentences with images [Elliott and Kádár, 2017]. The associated image captions/annotations can be direct translations of each other, but are not expected to be in general. The images are then used as pivots to induce a shared multimodal embedding space. These approaches typically use Multi30k [Elliott et al., 2016], a multilingual extension of the Flickr30k dataset [Young et al., 2014], which contains 30k images and 5 English sentence descriptions and their German translations for each image. Calixto et al. [2017] represent images using features from a pre-trained CNN and model sentences using a GRU. They then use the max-margin loss to assign a higher score to image-description pairs compared to images with a random description. Gella et al. [2017] augment this objective with another max-margin term that also brings the representations of translated descriptions closer together, thus effectively combining learning signals from both visual and textual modality.

3.5.6 Document-level alignment models

Models that require parallel document alignment presuppose that sentence-level parallel alignment is also present. Such models thus reduce to parallel sentence-level alignment methods, which have been discussed in the previous section. Comparable document-level alignment, on the other hand, is more appealing as it is often cheaper to obtain. Existing

approaches generally use Wikipedia documents, which they either automatically align, or they employ already theme-aligned Wikipedia documents discussing similar topics.

3.5.6.1 Document alignment methods with comparable data

We divide models using document alignment with comparable data into three types, some of which employ similar general ideas to previously discussed word and sentence-level parallel alignment models:

a) **Approaches based on pseudo-bilingual document-aligned corpora** automatically construct a pseudo-bilingual corpus containing words from the source and target language by mixing words from document-aligned documents.
b) **Concept-based methods** leverage information about the distribution of latent topics or concepts in document-aligned data to represent words.
c) **Extensions of sentence-aligned models** extend methods using sentence-aligned parallel data to also work without parallel data.

Pseudo-bilingual document-aligned corpora

The approach of Vulić and Moens [2016] is similar to existing pseudo-bilingual corpora approaches (§3.5.4.1). In contrast to previous methods, they propose a *merge and shuffle* strategy to merge two aligned documents of different languages into a pseudo-bilingual document. This is done by concatenating the documents and then randomly shuffling them by permuting words. The intuition is that as most methods rely on learning word embeddings based on their context, shuffling the documents will lead to robust bilingual contexts for each word. As the shuffling step is completely random, it might lead to sub-optimal configurations.

For this reason, they propose another strategy for merging the two aligned documents, called *length-ratio shuffle*. It assumes that the structures of the two documents are similar: words are inserted into the pseudo-bilingual document by alternating between the source and the target document relying on the order in which they appear in their monolingual document and based on the monolingual documents' length ratio.

Concept-based models

Some methods for learning cross-lingual word embeddings leverage the insight that words in different languages are similar if they are used to talk about or evoke the same multilingual concepts or topics. Vulić and Moens [2013] base their method on the

cognitive theory of semantic word responses. Their method centres on the intuition that words in source and target language are similar if they are likely to generate similar words as their top semantic word responses. They utilise a probabilistic multilingual topic model again trained on aligned Wikipedia documents to learn and quantify semantic word responses. The embedding $\mathbf{x}_i^S \in \mathbb{R}^{|V^S|+|V^T|}$ of source word w_i is the following vector:

$$\mathbf{x}_i^S = [P(w_1^S|w_i), \ldots, P(w_{|V^S|}^S|w_i), P(w_1^T|w_i) \ldots, P(w_{|V^T|}^T|w_i)] \tag{3.58}$$

where $[\cdot, \cdot]$ represents concatenation and $P(w_j|w_i)$ is the probability of w_j given w_i under the induced bilingual topic model. The sparse representations may be turned into dense vectors by factorizing the constructed word-response matrix.

Søgaard et al. [2015] propose an approach that relies on the structure of Wikipedia itself. Their method is based on the intuition that similar words are used to describe the same concepts across different languages. Instead of representing every Wikipedia concept with the terms that are used to describe it, they use an inverted index and represent a word by the concepts it is used to describe. As a post-processing step, dimensionality reduction on the produced word representations in the word-concept matrix is performed. A very similar model by Vulić et al. [2011] uses a bilingual topic model to perform the dimensionality reduction step and learns a shared cross-lingual topical space.

Extensions of sentence-alignment models

Mogadala and Rettinger [2016] extend the approach of Pham et al. [2015] to also work without parallel data and adjust the regularization term Ω based on the nature of the training corpus. Similar to previous work [Hermann and Blunsom, 2013, Gouws et al., 2015], they use the mean of the word embeddings of a sentence as the sentence representation $\mathbf{y}$ and constrain these to be close together. In addition, they propose to constrain the sentence paragraph vectors $\mathbf{p}^S$ and $\mathbf{p}^T$ of aligned sentences $sent^S$ and $sent^T$ to be close to each other. These vectors are learned via paragraph vectors [Le and Mikolov, 2014] for each sentence and stored in embedding matrices $\mathbf{P}^S$ and $\mathbf{P}^T$. The complete regularizer then uses elastic net regularization to combine both terms:

$$\Omega = \sum_{(sent^S, sent^T) \in \mathcal{C}} \alpha||\mathbf{p}^S - \mathbf{p}^T||^2 + (1-\alpha)\|\mathbf{y}^S - \mathbf{y}^T\|^2 \tag{3.59}$$

The monolingual paragraph vector objectives $\mathcal{L}_{\text{SGNS-P}}$ are then optimized jointly with the cross-lingual regularization term:

$$J = \mathcal{L}^S_{\text{SGNS-P}}(\mathbf{P}^S, \mathbf{X}^S) + \mathcal{L}^T_{\text{SGNS-P}}(\mathbf{P}^T, \mathbf{X}^T) + \Omega(\mathbf{P}^S, \mathbf{P}^T, \mathbf{X}^S, \mathbf{X}^T) \tag{3.60}$$

To leverage data that is not sentence-aligned, but where an alignment is still present on the document level, they propose a two-step approach: They use Procrustes analysis to find the most similar document in language T for each document in language S. In the second step, they then simply use the previously described method to learn cross-lingual word representations from the alignment documents, this time treating the entire documents as paragraphs.

As a final overview, we list all approaches with their monolingual objectives and regularization terms in Table 3.5. The table is meant to reveal the *high-level* objectives and losses each model is optimizing. It also indicates for each method whether all objectives are jointly optimized; if they are, both monolingual losses and regularization terms are optimized jointly; otherwise the monolingual losses are optimized first and the monolingual variables are frozen, while the cross-lingual regularization constraint is optimized. The table obscures smaller differences and implementation details, which can be found in the corresponding sections of this survey or by consulting the original papers. We use Ω_∞ to represent an infinitely stronger regularizer that enforces equality between representations. Regularizers with a * imply that the regularization is achieved in the limit, e.g. in the pseudo-bilingual case, where examples are randomly sampled with some equivalence, we obtain the same representation in the limit, without strictly enforcing it to be the same representation [Ruder et al., 2019b].

Most approaches can be seen as optimizing a combination of monolingual losses with a regularization term. As we can see, some approaches do not employ a regularization term; notably, a small number of approaches, i.e., those that ground language in images, do not optimize a loss but rather use pre-trained image features and a set of similarity heuristics to retrieve translations.

3.5.7 Evaluation

We now discuss the tasks that we will use for evaluation of cross-lingual word embeddings in this thesis (§5). For a more comprehensive overview of evaluation tasks and applications, we refer the reader to Ruder et al. [2019b].

Word similarity This task evaluates how well the notion of word similarity according to humans is emulated in the vector space. Multi-lingual word similarity datasets are

Approach	Monolingual	Regularizer	Joint?	Description
Klementiev et al. (2012)	$\mathcal{L}_{\text{MLE}}$	Ω_{MSE}	✓	Joint
Mikolov et al. (2013)	$\mathcal{L}_{\text{SGNS}}$	Ω_{MSE}		Projection-based
Zou et al. (2013)	$\mathcal{L}_{\text{C\&W}}$	Ω_{MSE}		Matrix factorization
Hermann and Blunsom (2013)	$\mathcal{L}_{\text{C\&W}}$	Ω^*_{MSE}	✓	Sentence-level, joint
Hermann and Blunsom (2014)	$\mathcal{L}_{\text{C\&W}}$	Ω^*_{MSE}	✓	Sentence-level + bigram composition
Soyer et al. (2015)	$\mathcal{L}_{\text{C\&W}}$	Ω^*_{MSE}	✓	Phrase-level
Shi et al. (2015)	$\mathcal{L}_{\text{C\&W}}$	Ω_{MSE}		Matrix factorization
Dinu et al. (2015)	$\mathcal{L}_{\text{SGNS}}$	Ω_{MSE}		Better neighbour retrieval
Gouws et al. (2015)	$\mathcal{L}_{\text{SGNS}}$	Ω_{MSE}	✓	Sentence-level
Vyas and Carpuat (2016)	$\mathcal{L}_{\text{GloVe}}$	Ω_{MSE}		Sparse matrix factorization
Hauer et al. (2017)	$\mathcal{L}_{\text{SGNS}}$	Ω_{MSE}		Cognates
Mogadala and Rettinger (2016)	$\mathcal{L}_{\text{SGNS-P}}$	Ω_{MSE}	✓	Elastic net, Procrustes analysis
Xing et al. (2015)	$\mathcal{L}_{\text{SGNS}}$	Ω_{MSE} s.t. $\mathbf{W}^\top\mathbf{W} = \mathbf{I}$		Normalization, orthogonality
Zhang et al. (2016)	$\mathcal{L}_{\text{SGNS}}$	Ω_{MSE} s.t. $\mathbf{W}^\top\mathbf{W} = \mathbf{I}$		Orthogonality constraint
Artexte et al. (2016)	$\mathcal{L}_{\text{SGNS}}$	Ω_{MSE} s.t. $\mathbf{W}^\top\mathbf{W} = \mathbf{I}$		Normalization, orthogonality, mean centering
Smith et al. (2017)	$\mathcal{L}_{\text{SGNS}}$	Ω_{MSE} s.t. $\mathbf{W}^\top\mathbf{W} = \mathbf{I}$		Orthogonality, inverted softmax identical character strings
Artexte et al. (2017)	$\mathcal{L}_{\text{SGNS}}$	Ω_{MSE} s.t. $\mathbf{W}^\top\mathbf{W} = \mathbf{I}$		Normalization, orthogonality, mean centering, bootstrapping
Lazaridou et al. (2015)	$\mathcal{L}_{\text{CBOW}}$	$\Omega_{\text{C\&W}}$		Max-margin with intruders
Mrkšić et al. (2017)	$\mathcal{L}_{\text{SGNS}}$	$\Omega_{\text{C\&W}}$		Semantic specialization
Calixto et al. (2017)	$\mathcal{L}_{\text{RNN}}$	$\Omega_{\text{C\&W}}$	✓	Image-caption pairs
Gella et al. (2017)	$\mathcal{L}_{\text{RNN}}$	$\Omega_{\text{C\&W}}$	✓	Image-caption pairs
Faruqui and Dyer (2014)	$\mathcal{L}_{\text{LSA}}$	Ω_{CCA}		-
Lu et al. (2015)	$\mathcal{L}_{\text{LSA}}$	Ω_{CCA}		Neural CCA
Rajendran et al. (2016)	$\mathcal{L}_{\text{AUTO}}$	Ω_{CCA}		Pivots
Ammar et al. (2016)	$\mathcal{L}_{\text{LSA}}$	Ω_{CCA}		Multilingual CCA
Søgaard et al. (2015)	-	Ω_{SVD}	✓	Inverted indexing
Levy et al. (2017)	$\mathcal{L}_{\text{PMI}}$	Ω_{SVD}	✓	
Levy et al. (2017)	-	Ω_{SGNS}	✓	Inverted indexing
Lauly et al. (2013)	$\mathcal{L}_{\text{AUTO}}$	Ω_{AUTO}	✓	Autoencoder
Chandar et al. (2014)	$\mathcal{L}_{\text{AUTO}}$	Ω_{AUTO}	✓	Autoencoder
Vulić and Moens (2013a)	$\mathcal{L}_{\text{LDA}}$	Ω^*_∞	✓	Document-level
Vulić and Moens (2014)	$\mathcal{L}_{\text{LDA}}$	Ω^*_∞	✓	Document-level
Xiao and Guo (2014)	$\mathcal{L}_{\text{C\&W}}$	Ω_∞	✓	Pseudo-multilingual
Gouws and Søgaard (2015)	$\mathcal{L}_{\text{CBOW}}$	Ω^*_∞	✓	Pseudo-multilingual
Luong et al. (2015)	$\mathcal{L}_{\text{SGNS}}$	Ω^*_∞		Monotonic alignment
Gardner et al. (2015)	$\mathcal{L}_{\text{LSA}}$	Ω^*_∞		Matrix factorization
Pham et al. (2015)	$\mathcal{L}_{\text{SGNS-P}}$	Ω_∞	✓	Paragraph vectors
Guo et al. (2015)	$\mathcal{L}_{\text{CBOW}}$	Ω_∞		Weighted by word alignments
Coulmance et al. (2015)	$\mathcal{L}_{\text{SGNS}}$	Ω^*_∞	✓	Sentence-level
Ammar et al. (2016)	$\mathcal{L}_{\text{SGNS}}$	Ω_∞	✓	Pseudo-multilingual
Vulić and Korhonen (2016)	$\mathcal{L}_{\text{SGNS}}$	Ω_∞		Highly reliable seed entries
Duong et al. (2016)	$\mathcal{L}_{\text{CBOW}}$	Ω_∞	✓	Pseudo-multilingual, polysemy
Vulić and Moens (2016)	$\mathcal{L}_{\text{SGNS}}$	Ω_∞	✓	Pseudo-multilingual documents
Adams et al. (2017)	$\mathcal{L}_{\text{CBOW}}$	Ω_∞	✓	Pseudo-multilingual, polysemy
Bergsma and Van Durme (2011)	-	-	✓	SIFT image features, similarity
Kiela et al. (2015)	-	-	✓	CNN image features, similarity
Vulić et al. (2016)	-	-	✓	CNN features, similarity, interpolation
Gouws and Søgaard (2015)	$\mathcal{L}_{\text{CBOW}}$	POS-level Ω^*_∞	✓	Pseudo-multilingual
Duong et al. (2015)	$\mathcal{L}_{\text{CBOW}}$	POS-level Ω^*_∞	✓	Pseudo-multilingual

TABLE 3.5: Overview of approaches with monolingual objectives and regularization terms, with an indication whether the order of optimization matters and short descriptions. Ω_∞ represents an infinitely strong regularizer that enforces equality between representations. * implies that the regularization is achieved in the limit.

multilingual extensions of datasets that have been used for evaluating English word representations. Many of these originate from psychology research and consist of word pairs—ranging from synonyms (e.g., car–automobile) to unrelated terms (e.g., noon–string)—that have been annotated with a relatedness score by human subjects.

Cross-lingual embeddings are evaluated on these datasets by first computing the cosine similarity of the representations of the cross-lingual word pairs. The Spearman's rank correlation coefficient [Myers et al., 2010] is then computed between the cosine similarity score and the human judgement scores for every word pair. Cross-lingual word similarity datasets are affected by the same problems as their monolingual variants Faruqui et al. [2016]: the annotated notion of word similarity is subjective and is often confused with relatedness; the datasets evaluate semantic rather than task-specific similarity, which is arguably more useful; they do not have standardized splits; they correlate only weakly with performance on downstream tasks; past models do not use statistical significance; and they do not account for polysemy, which is even more important in the cross-lingual setting.

Bilingual dictionary induction After the shared cross-lingual embedding space is induced, given a list of N source language words $x_{u,1}, \ldots, x_{u,N}$, the task is to find a target language word t for each *query word* x_u relying on the representations in the space. t_i is the target language word closest to the source language word $x_{u,i}$ in the induced cross-lingual space, also known as the *cross-lingual nearest neighbor*. The set of learned N $(x_{u,i}, t_i)$ pairs is then run against a gold standard dictionary.

Bilingual dictionary induction is appealing as an evaluation task, as high-quality, freely available, wide-coverage manually constructed dictionaries are still rare, especially for non-European languages. The task also provides initial intrinsic evidence on the quality of the shared space. Most previous work [Vulić and Moens, 2013, Gouws et al., 2015, Mikolov et al., 2013c] filters source and target words based on part-of-speech, though this simplifies the task and introduces bias in the evaluation. Each cross-lingual embedding model is then evaluated on its ability to select the closest target language word to a given source language word as the translation of choice and measured based on precision-at-one (P@1).

3.5.8 Summary

This survey has focused on providing an overview of cross-lingual word embedding models. It has introduced standardized notation and a typology that demonstrated the similarity of many of these models. It provided proofs that connect different word-level embedding

models and has described ways to evaluate cross-lingual word embeddings as well as how to extend them to the multilingual setting. It finally outlined challenges and future directions.

3.6 Conclusions

In this chapter, we have introduced transfer learning and reviewed the four transfer learning settings that are prevalent in NLP and that will be dealt with throughout this thesis. The first of these, domain adaptation, will be tackled in the next chapter. Throughout this chapter, we have furthermore emphasized the need to overcome a discrepancy between source and target scenarios, which is of particular importance in domain adaptation.

Chapter 4

Selecting Data for Domain Adaptation

In transfer learning, the similarity of the source and target setting is of key importance. A discrepancy may lead to negative transfer (§3.3.5). In this chapter, we will investigate the domain adaptation scenario, where source and target domains are different, i.e. $\mathcal{D}_S \neq \mathcal{D}_T$, while the source and target tasks are assumed to be the same, i.e. $\mathcal{T}_S = \mathcal{T}_T$.

To overcome the discrepancy between domains, we propose algorithms that automatically select both relevant and informative labelled and unlabelled examples for unsupervised domain adaptation. We evaluate our systems on a combination of sentiment analysis, part-of-speech tagging, and dependency parsing tasks. We empirically show that an appropriate selection of data helps in overcoming a domain shift, resulting in higher performance on the target domain.

In Section 4.1, we propose a novel method that uses Bayesian Optimization to automatically select relevant training data from multiple source domains for unsupervised domain adaptation. The approach learns an optimal combination of several existing domain similarity metrics (§3.4.2.1). It also considers the informativeness of an example by means of different measures of diversity on an example level.

In Section 4.2, we consider the problem of selecting unlabelled examples for domain adaptation. In this setting, we not only need to select relevant examples but also need to consider whether our model's predictions are reliable. To this end, we consider classic self-labelling approaches (§3.4.4) through the lens of current neural models. We adapt these classic methods to neural networks and compare them against state-of-the-art approaches. In addition, we propose multi-task tri-training, a more efficient method that combines the best of both worlds.

4.1 Learning to Select Data for Transfer Learning With Bayesian Optimization

* Domain similarity measures can be used to gauge adaptability and select suitable data for transfer learning, but existing approaches define ad hoc measures that are deemed suitable for respective tasks. Inspired by work on curriculum learning, we propose to *learn* data selection measures using Bayesian Optimization and evaluate them across models, domains and tasks. Our learned measures outperform existing domain similarity measures significantly on three tasks: sentiment analysis, part-of-speech tagging, and parsing. We show the importance of complementing similarity with diversity, and that learned measures are—to some degree—transferable across models, domains, and even tasks.

4.1.1 Introduction

Natural Language Processing models suffer considerably when applied in the wild. The distribution of the test data is typically very different from the data used during training, causing a model's performance to deteriorate substantially. Domain adaptation is a prominent approach to transfer learning that can help to bridge this gap; many approaches were suggested so far [Blitzer et al., 2007, Daumé III, 2007, Jiang and Zhai, 2007, Ma et al., 2014, Schnabel and Schütze, 2014].

However, most work focused on one-to-one scenarios: Given a set of source domains A and a set of target domains B, a model is evaluated based on its ability to adapt between all pairs (a, b) in the Cartesian product $A \times B$ where $a \in A$ and $b \in B$ [Remus, 2012]. However, adaptation between two dissimilar domains is often undesirable, as it may lead to negative transfer [Rosenstein et al., 2005]. Only recently, many-to-one adaptation [Mansour, 2009, Wu and Huang, 2016] has received some attention, as it replicates the realistic scenario of multiple source domains where performance on the target domain is the foremost objective. These approaches, however, are typically limited to a particular model or task.

Inspired by work on curriculum learning [Bengio et al., 2009, Tsvetkov et al., 2016], we instead propose—to the best of our knowledge—the first model-agnostic *data selection* approach to transfer learning. Contrary to curriculum learning that aims at speeding up learning (see §4.1.5), we aim at *learning to select* the most relevant data from multiple sources using data metrics. While several measures have been proposed in the past [Moore

*This section is adapted from: **Ruder, S.** and Plank, B. (2017). Learning to select data for transfer learning with Bayesian Optimization. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing.*

and Lewis, 2010, Axelrod et al., 2011, Van Asch and Daelemans, 2010, Plank and van Noord, 2011, Remus, 2012], prior work is limited by studying metrics mostly in isolation, using only the notion of similarity [Ben-David et al., 2007] and focusing on a single task (see §4.1.5). Our hypothesis is that different tasks or even different domains demand different notions of similarity. In this paper we go beyond prior work by i) studying a range of similarity metrics, including diversity; and ii) testing the robustness of the learned weights across models (e.g., whether a more complex model can be approximated with a simpler surrogate), domains and tasks (to delimit the transferability of the learned weights).

The contributions of this work are threefold. First, we present the first model-independent approach to *learn* a data selection measure for transfer learning. It outperforms baselines across three tasks and multiple domains and is competitive with state-of-the-art domain adaptation approaches. Second, prior work on transfer learning mostly focused on similarity. We demonstrate empirically that diversity is as important as—and complements—domain similarity for transfer learning. Note that rather than making the selected set of examples as diverse as possible, we focus on examples that have a diverse set of words. Finally, we show—for the first time—to what degree learned measures *transfer* across models, domains and tasks.

4.1.2 Data selection model

In order to select training data for adaptation for a task $\mathcal{T}$, existing approaches rank the available n training examples $X = \{x_1, x_2, \cdots, x_n\}$ of k source domains $D = \{\mathcal{D}_1, \mathcal{D}_2, \cdots, \mathcal{D}_k\}$ according to a domain similarity measure $\mathcal{S}$ and choose the top m samples for training their algorithm. While this has been shown to work empirically [Moore and Lewis, 2010, Axelrod et al., 2011, Plank and van Noord, 2011, Van Asch and Daelemans, 2010, Remus, 2012], using a pre-existing metric leaves us unable to adapt to the characteristics of our task $\mathcal{T}$ and target domain $\mathcal{D}_T$ and foregoes additional knowledge that may be gleaned from the interaction of different metrics. For this reason, we propose to *learn* the following linear domain similarity measure $\mathcal{S}$ as a linear combination of feature values:

$$\mathcal{S} = \phi(X) \cdot w^\intercal \tag{4.1}$$

where $\phi(X) \in \mathbb{R}^{n \times l}$ are the similarity and diversity features further described in §4.1.2.2 for each training example, with l being the number of features, while $w \in \mathbb{R}^l$ are the weights learned by Bayesian Optimization.

We aim to learn weights w in order to optimize the objective function J of the respective task $\mathcal{T}$ on a small number of validation examples of the corresponding target domain $\mathcal{D}_T$.

4.1.2.1 Bayesian Optimization for data selection

As the learned measure $\mathcal{S}$ should be agnostic of the particular objective function J, we cannot use gradient-based methods for optimization. Similar to [Tsvetkov et al., 2016], we use Bayesian Optimization [Brochu et al., 2010], which has emerged as an efficient framework to optimize any function. For instance, it has repeatedly found better settings of neural network hyperparameters than domain experts [Snoek et al., 2012].

Given a black-box function $f : \mathbb{X} \rightarrow \mathbb{R}$, Bayesian Optimization aims to find an input $\hat{x} \in \arg\min_{x \in \mathbb{X}} f(x)$ that globally minimizes f. For this, it requires a prior $p(f)$ over the function and an acquisition function $a_{p(f)} : \mathbb{X} \rightarrow \mathbb{R}$ that calculates the *utility* of any evaluation at any x.

Bayesian Optimization then proceeds iteratively. At iteration t, 1) it finds the most promising input $x_t \in \arg\max a_p(x)$ through numerical optimization; 2) it then evaluates the surrogate function $y_t \sim f(x_t) + \mathcal{N}(0, \sigma^2)$ on this input and adds the resulting data point (x_t, y_t) to the set of observations $\mathcal{O}_{t-1} = (x_j, y_j)_{j=1...t-1}$; 3) finally, it updates the prior $p(f|\mathcal{O}_t)$ and the acquisition function $a_{p(f|\mathcal{O}_t)}$.

For data selection, the black-box function f looks as follows: 1) It takes as input a set of weights w that should be evaluated; 2) the training examples of all source domains are then scored and sorted according to Equation 4.1; 3) the model for the respective task $\mathcal{T}$ is trained on the top n samples; 4) the model is evaluated on the validation set according to the evaluation measure J and the value of J is returned.

Gaussian Processes (GP) are a popular choice for $p(f)$ due to their descriptive power [Rasmussen, 2006]. We use GP with Monte Carlo acquistion and Expected Improvement (EI) [Močkus, 1974] as acquisition function as this combination has been shown to outperform comparable approaches [Snoek et al., 2012].[1]

4.1.2.2 Features

Existing work on data selection for domain adaptation selects data based on its *similarity* to the target domain. Several measures have been proposed in the literature [Van Asch and Daelemans, 2010, Plank and van Noord, 2011, Remus, 2012], but were so far only used in isolation.

Only selecting training instances with respect to the target domain also fails to account for instances that are richer and better suited for knowledge acquisition. For this reason,

[1] We also experimented with `FABOLAS` [Klein et al., 2017], but found its ability to adjust the training set size during optimization to be inconclusive for our relatively small training sets.

we consider—to our knowledge for the first time—whether intrinsic qualities of the training data accounting for *diversity* are of use for domain adaptation in NLP.

Similarity We use a range of similarity metrics. Some metrics might be better suited for some tasks, while different measures might capture complementary information. We thus use the following measures as features for learning a more effective domain similarity metric.

We define similarity features over probability distributions in accordance with existing literature [Plank and van Noord, 2011]. Let P be the representation of a source training example, while Q is the corresponding target domain representation. Let further $M = \frac{1}{2}(P+Q)$, i.e. the average distribution between P and Q and let $D_{KL}(P||Q) = \sum_{i=1}^{n} p_i \log \frac{p_i}{q_i}$, i.e., the KL divergence between the two domains. We do not use D_{KL} as a feature as it is undefined for distributions where some event $q_i \in Q$ has probability 0, which is common for term distributions. Our features are:

- Jensen-Shannon divergence [Lin, 1991]: $\frac{1}{2}[D_{KL}(P||M) + D_{KL}(Q||M)]$. Jensen-Shannon divergence is a smoothed, symmetric variant of D_{KL} that has been successfully used for domain adaptation [Plank and van Noord, 2011, Remus, 2012].
- Rényi divergence [Rényi, 1961]: $\frac{1}{\alpha-1} \log(\sum_{i=1}^{n} \frac{p_i^\alpha}{q_i^{\alpha-1}})$. Rényi divergence reduces to D_{KL} as $\alpha \to 1$. We set $\alpha = 0.99$ following [Van Asch and Daelemans, 2010].
- Bhattacharyya distance [Bhattacharya, 1943]: $\ln(\sum_i \sqrt{P_i Q_i})$
- Cosine similarity [Lee, 2001]: $\frac{P \cdot Q}{\|P\| \, \|Q\|}$. We can treat the distributions alternatively as vectors and consider geometrically motivated distance functions such as cosine similarity as well as the following.
- Euclidean distance [Lee, 2001]: $\sqrt{\sum_i (P_i - Q_i)^2}$.
- Variational dist. [Lee, 2001]: $\sum_i |P_i - Q_i|$.

We consider three different representations for calculating the above domain similarity measures:

- Term distributions [Plank and van Noord, 2011]: $t \in \mathbb{R}^{|V|}$ where t_i is the probability of the i-th word in the vocabulary V.
- Topic distributions [Plank and van Noord, 2011]: $t \in \mathbb{R}^n$ where t_i is the probability of the i-th topic as determined by an LDA model [Blei et al., 2003] trained on the data and n is the number of topics.

- Word embeddings [Mikolov et al., 2013a]: $\frac{1}{n}\sum_i v_{w_i}\sqrt{\frac{a}{p(w_i)}}$ where n is the number of words with embeddings in the document, v_{w_i} is the pre-trained embedding of the i-th word, $p(w_i)$ its probability, and a is a smoothing factor used to discount frequent probabilities. A similar weighted sum has recently been shown to outperform supervised approaches for other tasks [Arora et al., 2017]. As embeddings may be negative, we use them only with the latter three geometric features above.

Diversity For each training example, we calculate its diversity based on the words in the example. These diversity metrics operate on the example level and aim to measure an example's intrinsic informativeness. They do not seek to measure the diversity of the entire set of examples.[2]

Let p_i and p_j be probabilities of the word types t_i and t_j in the training data and $\cos(v_{t_i}, v_{t_j})$ the cosine similarity between their word embeddings. We employ measures that have been used in the past for measuring diversity [Tsvetkov et al., 2016]:

- Number of word types: $\#types$.
- Type-token ratio: $\frac{\#types}{\#tokens}$.
- Entropy [Shannon, 1948]: $-\sum_i p_i \ln(p_i)$.
- Simpson's index [Simpson, 1949]: $-\sum_i p_i^2$.
- Rényi entropy [Rényi, 1961]: $\frac{1}{\alpha-1}\log(\sum_i p_i^\alpha)$
- Quadratic entropy [Rao, 1982]: $\sum_{i,j}\cos(v_{t_i}, v_{t_j})p_i p_j$.

4.1.3 Experiments

Tasks and datasets We evaluate our approach on three tasks: sentiment analysis, part-of speech (POS) tagging, and dependency parsing. We use the n examples with the highest score as determined by the learned data selection measure for training our models.[3] We show statistics for all datasets in Table 4.1.

Sentiment analysis For sentiment analysis, we evaluate on the Amazon reviews dataset [Blitzer et al., 2006]. We use tf-idf-weighted unigram and bigram features and a linear SVM classifier [Blitzer et al., 2007]. We set the vocabulary size to 10,000 and the number of training examples $n = 1600$ to conform with existing approaches [Bollegala et al., 2011] and stratify the training set.

[2]Future work may consider the diversity of all examples jointly, for instance by employing an algorithm that greedily selects examples in order to maximize a set-level diversity metric.

[3]All code is available at `https://github.com/sebastianruder/learn-to-select-data`.

$\mathcal{T}$	**Domain**	**# labeled**	**# unlabeled**
Sentiment	Book	2000	4465
	DVD	2000	3586
	Electronics	2000	5681
	Kitchen	2000	5945
POS/Parsing	Answers	3489	27274
	Emails	4900	1194173
	Newsgroups	2391	1000000
	Reviews	3813	1965350
	Weblogs	2031	524834
	WSJ	2976	30060

TABLE 4.1: Statistics for the Amazon Reviews dataset for sentiment analysis (top) and the SANCL 2012 dataset for POS tagging and parsing (bottom).

POS tagging For POS tagging and parsing, we evaluate on the coarse-grained POS data (12 universal POS) of the SANCL 2012 shared task [Petrov and McDonald, 2012]. Each domain—except for WSJ—contains around 2000-5000 labeled sentences and more than 100,000 unlabeled sentences. In the case of WSJ, we use its dev and test data as labeled samples and treat the remaining sections as unlabeled. We set $n = 2000$ for POS tagging and parsing to retain enough examples for the most-similar-domain baseline.

To evaluate the impact of model choice, we compare two models: a Structured Perceptron (in-house implementation with commonly used features pertaining to tags, words, case, prefixes, as well as prefixes and suffixes) trained for 5 iterations with a learning rate of 0.2; and a state-of-the-art Bi-LSTM tagger [Plank et al., 2016] with word and character embeddings as input. We perform early stopping on the validation set with patience of 2 and use otherwise default hyperparameters[4] as provided by the authors.

Parsing For parsing, we evaluate the state-of-the-art Bi-LSTM parser by [Kiperwasser and Goldberg, 2016] [Kiperwasser and Goldberg, 2016] with default hyperparameters.[5] We use the same domains as used for POS tagging, i.e., the dependency parsing data with gold POS as made available in the SANCL 2012 shared task.[6]

Training details In practice, as feature values occupy different ranges, we have found it helpful to apply z-normalisation similar to Tsvetkov et al. [2016]. We moreover constrain the weights w to $[-1, 1]$.

[4] `https://github.com/bplank/bilstm-aux`
[5] `https://github.com/elikip/bist-parser`
[6] We leave investigating the effect of the adapted taggers on parsing for future work.

For each dataset, we treat each domain as target domain and all other domains as source domains. Similar to Bousmalis et al. [2016], we chose to use a small number (100) target domain examples as validation set. We optimize each similarity measure using Bayesian Optimization with 300 iterations according to the objective measure J of each task (accuracy for sentiment analysis and POS tagging; LAS for parsing) with respect to the validation set of the corresponding target domain.

Unlabeled data is used in addition to calculate the representation of the target domain and to calculate the source domain representation for the most similar domain baseline. We train an LDA model [Blei et al., 2003] with 50 topics and 10 iterations for topic distribution-based representations and use GloVe embeddings [Pennington et al., 2014] trained on 42B tokens of Common Crawl data[7] for word embedding-based representations.

For sentiment analysis, we conduct 10 runs of each feature set for every domain and report mean and variance. For POS tagging and parsing, we observe that variance is low and perform one run while retaining random seeds for reproducibility.

Baselines and features We compare the learned measures to three baselines: i) a random baseline that randomly selects n training samples from all source domains (*random*); ii) the top n examples selected using Jensen-Shannon divergence (*JS – examples*), which outperformed other measures in previous work [Plank and van Noord, 2011, Remus, 2012]; iii) n examples randomly selected from the most similar source domain determined by Jensen-Shannon divergence (*JS – domain*). We additionally compare against training on all available source data (6,000 examples for sentiment analysis; 14,700-17,569 examples for POS tagging and parsing depending on the target domain).

We optimize data selection using Bayesian Optimization with every feature set: similarity features respectively based on i) word embeddings, ii) term distributions, and iii) topic distributions; and iv) diversity features. In addition, we investigate how well different representations help each other by using similarity features with the two best-performing representations, term distributions and topic distributions. Finally, we explore whether diversity and similarity-based features complement each other by in turn using each similarity-based feature set together with diversity features.

4.1.4 Results

Sentiment analysis We show results for sentiment analysis in Table 4.2. The best results are highlighted in bold and the second-best are underlined. First of all, the

[7] `https://nlp.stanford.edu/projects/glove/`

		Target domains			
	Feature set	**Book**	**DVD**	**Electronics**	**Kitchen**
Base	Random	71.17 (± 4.41)	70.51 (± 3.33)	76.75 (± 1.77)	77.94 (± 3.72)
	Jensen-Shannon divergence – examples	72.51 (± 0.42)	68.21 (± 0.34)	76.51 (± 0.63)	77.47 (± 0.44)
	Jensen-Shannon divergence – domain	75.26 (± 1.25)	73.74 (± 1.36)	72.60 (± 2.19)	80.01 (± 1.93)
Learned measures	Similarity (word embeddings)	75.06 (± 1.38)	74.96 (± 2.12)	80.79 (± 1.31)	83.45 (± 0.96)
	Similarity (term distributions)	75.39 (± 0.98)	76.25 (± 0.96)	81.91 (± 0.57)	83.39 (± 0.84)
	Similarity (topic distributions)	76.04 (± 1.10)	75.89 (± 0.81)	81.69 (± 0.96)	83.09 (± 0.95)
	Diversity	76.03 (± 1.28)	77.48 (± 1.33)	81.15 (± 0.67)	83.94 (± 0.99)
	Sim (term dists) + sim (topic dists)	75.76 (± 1.30)	76.62 (± 0.95)	81.73 (± 0.63)	83.43 (± 0.75)
	Sim (word embeddings) + diversity	75.52 (± 0.98)	77.50 (± 0.61)	80.97 (± 0.83)	84.28 (± 1.02)
	Sim (term dists) + diversity	76.20 (± 1.45)	77.60 (± 1.01)	**82.67** (± 0.73)	**84.98** (± 0.60)
	Sim (topic dists) + diversity	**77.16** (± 0.77)	**79.00** (± 0.93)	81.92 (± 1.32)	84.29 (± 1.00)
	All source data (6,000 training examples)	70.86 (± 0.51)	68.74 (± 0.32)	77.39 (± 0.32)	73.09 (± 0.41)

TABLE 4.2: Accuracy scores for data selection for sentiment analysis domain adaptation on the Amazon reviews dataset.

baselines show that the sentiment review domains are clearly delimited. Adapting between two similar domains such as Book and DVD is more productive than adaptation between dissimilar domains, e.g. Books and Electronics, as shown in previous work [Blitzer et al., 2007]. This explains the strong performance of the most-similar-domain baseline.

In contrast, selecting individual examples based on a domain similarity measure performs only as good as chance. Thus, when domains are more clear-cut, selecting from the closest domain is a stronger baseline than selecting from the entire pool of source data.

If we learn a data selection measure using Bayesian Optimization, we are able to outperform the baselines with almost all feature sets. Performance gains are considerable for all domains with individual feature sets (term similarity, word embeddings similarity, diversity and topic similarity), except for Books were improvements for some single feature sets are smaller. Term distributions and topic distributions are the best-performing representations for calculating similarity, with term distributions performing slightly better across all domains. Combining term distribution-based and topic distribution-based features only provides marginal gains over the individual feature sets, demonstrating that most of the information is contained in the similarity features rather than the representations.

Diversity features perform comparatively to the best similarity features and outperform them on two domains. Furthermore, the combination of diversity and similarity features yields another sizable gain of around 1 percentage point for almost all domains over the best similarity features, which shows that diversity and similarity features capture complementary information. Term distribution and topic distribution-based similarity features in conjunction with diversity features finally yield the best performance, outperforming the baselines by 2-6 points in absolute accuracy.

	Trg domains →	Answers			Emails			Newsgroups			Reviews			Weblogs			WSJ		
	Task →	POS		Pars	POS		Pars	POS		Pars	POS		Pars	POS		Pars	POS		Pars
	Feat ↓ Model →	P	B	BIST	P	B	BIST	P	B	BIST	P	B	BIST	P	B	BIST	P	B	BIST
Base	Random	91.34	92.55	81.02	91.80	93.25	79.09	92.50	93.26	80.61	92.08	92.12	82.30	92.76	93.03	82.39	91.08	92.54	78.31
	JS – examples	92.42	93.16	82.80	91.75	93.77	80.53	92.96	94.29	83.25	92.77	93.32	84.35	94.33	94.92	85.36	92.85	94.08	82.43
	JS – domain	90.84	91.13	80.37	91.64	93.16	79.93	92.23	92.67	81.77	92.27	92.67	82.11	93.19	94.34	83.44	91.20	92.99	80.61
Learned measures	W2v sim	92.53	93.22	82.74	92.94	94.14	81.18	93.41	94.09	81.62	93.51	93.30	82.98	94.41	94.83	84.30	93.02	94.66	81.57
	Term sim	**93.13**	93.43	**83.79**	92.96	94.04	81.09	93.58	**94.55**	82.68	93.53	**93.73**	**84.66**	94.42	**95.09**	84.85	**93.44**	94.11	82.57
	Topic sim	92.50	93.16	82.87	92.70	**94.48**	81.43	93.97	94.09	82.07	93.21	93.22	83.98	94.42	93.71	84.98	93.09	94.02	**82.90**
	Diversity	92.33	92.58	83.01	93.08	93.56	80.93	**94.37**	93.97	83.98	93.33	93.05	83.92	94.62	94.94	85.84	93.33	93.44	82.80
	Term+topic sim	92.80	**93.69**	82.87	92.70	92.28	81.13	93.57	93.76	82.97	**93.56**	93.61	84.65	94.41	94.23	84.43	93.07	**94.68**	82.43
	W2v sim+div	92.76	92.38	82.34	**93.51**	94.19	80.77	93.96	94.10	**84.26**	93.45	93.39	84.47	94.36	94.95	85.53	93.32	93.20	82.32
	Term sim+div	92.73	93.46	83.72	92.90	93.81	**81.60**	94.03	93.47	82.80	93.47	93.29	84.62	**94.76**	95.06	85.44	93.32	93.68	82.87
	Topic sim+div	92.93	93.62	82.60	92.62	93.93	80.83	93.85	94.06	84.04	93.16	93.59	84.45	94.42	94.45	**85.89**	93.38	94.23	82.33
	All source data	94.30	95.16	86.34	94.34	95.90	85.57	95.40	95.90	87.18	94.90	95.03	87.51	95.53	95.79	88.23	94.19	95.64	85.20

TABLE 4.3: Results for data selection for part-of-speech tagging (POS) and parsing domain adaptation on the SANCL 2012 dataset.

Finally, we compare data selection to training on all available source data (in this setup, 6,000 instances). The result complements the findings of the most-similar baseline: as domains are dissimilar, training on all available sources is detrimental.

POS tagging Results for POS tagging are given in Table 4.3 using accuracy as evaluation metric. Best results are again highlighted in bold and the second-best are underlined.

Using Bayesian Optimization, we are able to outperform the baselines with almost all feature sets, except for a few cases (e.g., diversity and word embeddings similarity, topic and term distributions). Overall term distribution-based similarity emerges as the most powerful individual feature. Combining it with diversity does not prove as beneficial as in the sentiment analysis case, however, often yields the second-best results.

Notice that for POS tagging/parsing, in contrast to sentiment analysis, the most-similar domain baseline is not effective, it often performs only as good as chance, or even hurts. In contrast, the baseline that selects instances (*JS – examples*) rather than a domain performs better. This makes sense as in SA topically closer domains express sentiment in more similar ways, while for POS tagging having more varied training instances is intuitively more beneficial. In fact, when inspecting the domain distribution of our approach, we find that the best SA model chooses more instances from the closest domain, while for POS tagging instances are more balanced across domains. This suggests that the Web treebank domains are less clear-cut. In fact, training a model on all sources, which is considerably more and varied data (in this setup, 14-17.5k training instances) is beneficial. This is in line with findings in machine translation [Mirkin and Besacier, 2014], which show that similarity-based selection works best if domains are very different. Results are thus less pronounced for POS tagging, and we leave experimenting with larger n for future work.

To gain some insight into the optimization procedure, Figure 4.1 shows the development accuracy for the Structured Perceptron exemplarily on the Reviews domain. The top-right and bottom graphs show the hypothesis space exploration of Bayesian Optimization for different single feature sets, while the top-left graph displays the overall best dev accuracy for different feature sets. We observe again that term similarity is among the best feature sets and results in a larger explored space (more variance), in contrast to the diversity features whose development accuracy increases less and results in an overall less explored space. Exploration plots for other features/models looks similar.

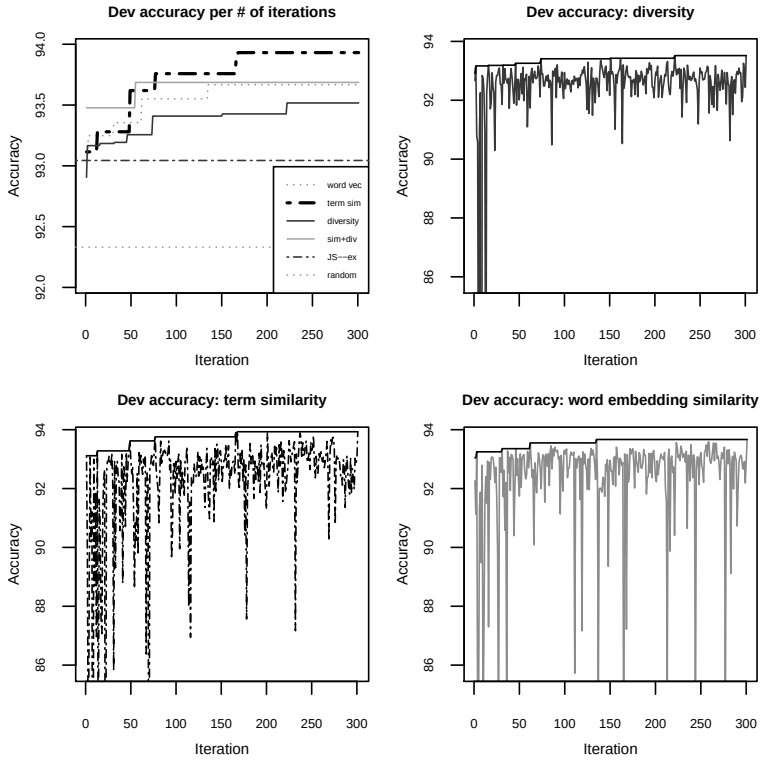


FIGURE 4.1: Dev accuracy curves of Bayes Optimization for POS tagging on the Reviews domain.

Parsing The results for parsing are given in Table 4.3 using Labeled Attachment Score (LAS) as evaluation metric. Diversity features are stronger than for POS tagging and outperform the baselines for all except the Reviews domain. Similarly to POS tagging, term distribution-based similarity as well as its combination with diversity features yield the best results across most domains.

Transfer across models In addition, we are interested how well the metric learned for one target domain transfers to other settings. We first investigate its ability to transfer to another model. In practice, a metric can be learned using a model that is cheap to evaluate and serves as proxy for a state-of-the-art model, in a way similar to

uptraining [Petrov et al., 2010]. For this, we employ the data selection features learned using the Structured Perceptron model for POS tagging and use them to select data for the Bi-LSTM tagger. The data selection weights are learned using model $\mathcal{M}_\mathcal{S}$; the Bi-LSTM tagger (B) is then trained using the learned weights.

We show results in Table 4.4. Performance that is better than the baselines is underlined. The results indicate that cross-model transfer is indeed possible, with most transferred feature sets achieving similar results or even outperforming features learned with the Bi-LSTM. In particular, transferred diversity significantly outperforms its in-model equivalent. This is encouraging, as it allows to learn a data selection metric using less complex models.

		Target domains											
		Answers		**Emails**		**Newsgroups**		**Reviews**		**Weblogs**		**WSJ**	
Feature set ↓	$\mathcal{M}_\mathcal{S}$ →	**B**	$\mathbf{P}_{proxy}$	**B**	$\mathbf{P}_{proxy}$	**B**	$\mathbf{P}_{proxy}$	**B**	$\mathbf{P}_{proxy}$	**B**	$\mathbf{P}_{proxy}$	**B**	$\mathbf{P}_{proxy}$
Term similarity		93.43	93.67	94.04	93.88	94.55	93.77	**93.73**	93.54	**95.09**	95.06	94.11	94.30
Diversity		92.58	93.19	93.56	**94.40**	93.97	**94.96**	93.05	93.52	94.94	94.91	93.44	**94.14**
Term similarity+diversity		**93.46**	93.18	93.81	94.29	93.47	94.28	93.29	93.35	95.06	94.67	93.68	93.92

TABLE 4.4: Accuracy scores for cross-model transfer of learned data selection weights on the SANCL 2012 dataset.

Transfer across domains We explore whether data selection parameters learned for one target domain transfer to other target domains. For each domain, we use the weights with the highest performance on the validation set and use them for data selection with the remaining domains as target domains. We conduct 10 runs for the best-performing feature sets for sentiment analysis and report the average accuracy scores in Table 4.5 (for POS tagging using the Structured Perceptron, see Table 4.6). We indicate the target domain used for learning the metric $\mathcal{S}$ with $\mathcal{D}_\mathcal{S}$ and highlight the best performance per feature set in bold and in-domain results in gray. We list the state-of-the-art domain adaptation approach SDAMS [Wu and Huang, 2016] as comparison.

The transfer of the weights learned with Bayesian Optimization is quite robust in most cases. Feature sets like Similarity or Diversity trained on Books outperform the strong JS – $\mathcal{D}$ baseline in all 6 cases, for Electronics and Kitchen in 4/6 cases (off-diagonals for box 2 and 3 in Table 4.5). In some cases, the transferred weights even outperform the data selection metric learned for the respective domain, such as on D→E with sim and sim+div features and by almost 2 pp on E→D.

Transferred similarity+diversity features mostly achieve higher performance than other feature sets, but the higher number of parameters runs the risk of overfitting to the domain as can be observed with two instances of negative transfer with sim+div features.

As a reference, we also list the performance of the state-of-the-art multi-domain adaptation approach [Wu and Huang, 2016], which shows that task-independent data selection is

		Target domains			
Feature	$\mathcal{D}_S$	**B**	**D**	**E**	**K**
Sim	B	**75.39**	75.22	80.74	80.41
Sim	D	75.30	76.25	**82.68**	82.29
Sim	E	74.55	76.65	81.91	82.23
Sim	K	73.64	**76.66**	81.09	**83.39**
Div	B	**76.03**	75.16	80.16	80.01
Div	D	75.68	**77.48**	65.74	72.48
Div	E	74.69	76.60	**81.15**	81.97
Div	K	75.03	76.23	80.71	**83.94**
Sim+div	B	**76.20**	64.81	65.06	79.87
Sim+div	D	74.17	77.60	**83.26**	**85.19**
Sim+div	E	74.14	**79.32**	82.67	84.53
Sim+div	K	75.54	76.11	78.72	84.98
SDAMS	-	78.29	79.13	84.06	86.29

TABLE 4.5: Accuracy scores for cross-domain transfer of learned data selection weights on Amazon reviews.

		Target domains					
Feature set	$\mathcal{D}_S$	**Answers (A)**	**Emails (E)**	**Newsgroups (N)**	**Reviews (R)**	**Weblogs (W)**	**WSJ**
Term similarity	A	**93.13**	91.60	93.94	**93.63**	94.26	92.42
Term similarity	E	92.35	**92.96**	93.42	**93.63**	93.75	92.24
Term similarity	N	92.48	92.28	93.58	93.35	93.95	93.00
Term similarity	R	92.06	92.18	93.38	93.53	94.26	91.88
Term similarity	W	92.69	92.12	**93.65**	93.12	**94.42**	92.63
Term similarity	WSJ	92.50	92.51	93.53	93.00	94.29	**93.44**
Diversity	A	92.33	92.14	93.46	92.00	94.01	92.56
Diversity	E	92.11	**93.08**	93.81	92.67	94.16	93.13
Diversity	N	**92.67**	92.22	**94.37**	92.44	94.05	92.96
Diversity	R	92.65	92.72	93.67	**93.33**	94.18	93.28
Diversity	W	92.19	92.31	93.31	92.20	**94.62**	92.04
Diversity	WSJ	92.26	92.31	93.75	92.70	94.32	**93.33**
Term similarity+diversity	A	92.73	92.63	93.16	92.58	93.88	92.23
Term similarity+diversity	E	92.55	92.90	93.78	92.73	93.54	92.57
Term similarity+diversity	N	92.47	92.27	**94.03**	92.63	94.30	93.14
Term similarity+diversity	R	**92.80**	**93.11**	93.92	93.47	93.79	92.99
Term similarity+diversity	W	92.61	92.45	93.44	**93.52**	**94.76**	93.26
Term similarity+diversity	WSJ	91.82	92.37	93.52	92.63	94.17	**93.32**

TABLE 4.6: Accuracy scores for cross-domain transfer of learned data selection weights for part-of-speech tagging on the SANCL 2012 dataset.

in fact competitive with a task-specific, heuristic state-of-the-art domain adaptation approach. In fact our *transferred* similarity+diversity feature (E→D) outperforms the state-of-the-art [Wu and Huang, 2016] on DVD. This is encouraging as previous work [Remus, 2012] has shown that data selection and domain adaptation can be complementary.

Transfer across tasks We finally investigate whether data selection is task-specific or whether a metric learned on one task can be transferred to another one. For each feature set, we use the learned weights for each domain in the source task (for sentiment analysis, we use the best weights on the validation set; for POS tagging, we use the Structured

Perceptron model) and run experiments with them for all domains in the target task.[8] We report the averaged accuracy scores for transfer across all tasks in Table 4.7. $\mathcal{T}_{\mathcal{S}}$ is the task used for learning metric $\mathcal{S}$. We use the same features as in Table 4.5 and highlight in-task reults with gray and underline results that are better than the baselines.

		Target tasks		
Feature set	$\mathcal{T}_{\mathcal{S}}$	**POS**	**Pars**	**SA**
Sim	POS	93.51	83.11	74.19
Sim	Pars	92.78	83.27	72.79
Sim	SA	86.13	67.33	79.23
Div	POS	93.51	83.11	69.78
Div	Pars	93.02	83.41	68.45
Div	SA	90.52	74.68	79.65
Sim+div	POS	93.54	83.24	69.79
Sim+div	Pars	93.11	83.51	72.27
Sim+div	SA	89.80	75.17	80.36

TABLE 4.7: Results of cross-task transfer of learned data selection weights.

Transfer is productive between related tasks, i.e. POS tagging and parsing results are similar to those obtained with data selection learned for the particular task. We observe large drops in performance for transfer between unrelated tasks, such as sentiment analysis and POS tagging, which is expected since these are very different tasks. Between related tasks, the combination of similarity and diversity features achieves the most robust transfer and outperforms the baselines in both cases. This suggests that even in the absence of target task data, we only require data of a related task to learn a successful data selection measure.

4.1.5 Related work

Most prior work on data selection for transfer learning focuses on phrase-based machine translation [Moore and Lewis, 2010, Axelrod et al., 2011, Duh et al., 2013, Mirkin and Besacier, 2014], with fewer studies in other tasks, e.g., constituent parsing [McClosky et al., 2010], dependency parsing [Plank and van Noord, 2011, Søgaard, 2011] and sentiment analysis [Remus, 2012] (§3.4.3.2). We are not aware of another study that covers three distinct tasks. In addition, we test previously explored similarity metrics (§3.4.2.1) and complement them with diversity.

Very recently interest emerged in *curriculum learning* [Bengio et al., 2009]. It is inspired by human active learning by providing easier examples at initial learning stages (e.g., by curriculum strategies such as growing vocabulary size). Curriculum learning employs

[8] E.g., for SA-¿POS, for each feature set, we obtain one set of weights for each of 4 SA domains, which we use to select data for the 6 POS domains, yielding $4 \cdot 6 = 24$ results.

a range of data metrics, but aims at altering the order in which the entire training data is selected, rather than *selecting* data. In contrast to us, curriculum learning is mostly aimed at speeding up the learning, while we focus on learning metrics for transfer learning. Other related work in this direction include using Reinforcement Learning to learn what data to select during neural network training [Fan et al., 2017b].

There is a long history of research in adaptive data selection, with early approaches grounded in information theory using a Bayesian learning framework [MacKay, 1992]. It has also been studied extensively as active learning [El-Gamal, 1991]. Curriculum learning is related to active learning [Settles, 2012], whose view is different: active learning aims at finding the most difficult instances to label, examples typically close to the decision boundary. Confidence-based measures are prominent, but as such are less widely applicable than our model-agnostic approach.

The approach most similar to ours is by Tsvetkov et al. [2016] who use Bayesian Optimization to learn a curriculum for training word embeddings. Rather than ordering data (in their case, paragraphs), we use Bayesian Optimization for learning to *select* relevant training instances that are useful for transfer learning in order to prevent negative transfer [Rosenstein et al., 2005]. To the best of our knowledge there is no prior work that uses this strategy for transfer learning.

4.1.6 Summary

In this section, we propose to use Bayesian Optimization to learn data selection measures for transfer learning. Our results outperform existing domain similarity metrics on three tasks (sentiment analysis, POS tagging and parsing), and are competitive with a state-of-the-art domain adaptation approach. More importantly, we present the first study on the transferability of such measures, showing promising results to port them across models, domains and related tasks.

In the following section, we in addition consider unlabelled examples for data selection and focus on a particular category of semi-supervised learning methods that learns to assign labels to such unlabelled examples. While we previously incorporated a measure of informativeness explicitly via features, our models in the next section will aim to learn how reliable an example based on the agreement with each other.

4.2 Strong Baselines for Neural Semi-supervised Learning under Domain Shift

* Novel neural models have been proposed in recent years for learning under domain shift. Most models, however, only evaluate on a single task, on proprietary datasets, or compare to weak baselines, which makes comparison of models difficult. In this section, we re-evaluate classic general-purpose bootstrapping approaches in the context of neural networks under domain shifts vs. recent neural approaches and propose a novel *multi-task tri-training* method that reduces the time and space complexity of classic tri-training. Extensive experiments on two benchmarks are negative: while our novel method establishes a new state-of-the-art for sentiment analysis, it does not fare consistently the best. More importantly, we arrive at the somewhat surprising conclusion that classic tri-training, with some additions, outperforms the state of the art. We conclude that classic approaches constitute an important and strong baseline.

4.2.1 Introduction

Deep neural networks excel at learning from labeled data and have achieved state of the art in a wide array of supervised NLP tasks such as dependency parsing [Dozat and Manning, 2017], named entity recognition [Lample et al., 2016], and semantic role labeling [He et al., 2017].

In contrast, learning from unlabeled data, especially under domain shift, remains a challenge. This is common in many real-world applications where the distribution of the training and test data differs. Many state-of-the-art domain adaptation approaches leverage task-specific characteristics such as sentiment words [Blitzer et al., 2006, Wu and Huang, 2016] or distributional features [Schnabel and Schütze, 2014, Yin et al., 2015] which do not generalize to other tasks. Other approaches that are in theory more general only evaluate on proprietary datasets [Kim et al., 2017a] or on a single benchmark [Zhou et al., 2016a], which carries the risk of overfitting to the task. In addition, most models only compare against weak baselines and, strikingly, almost none considers evaluating against approaches from the extensive semi-supervised learning (SSL) literature [Chapelle et al., 2006].

In this work, we make the argument that such algorithms make strong baselines for any task in line with recent efforts highlighting the usefulness of classic approaches [Melis et al., 2017, Denkowski and Neubig, 2017]. We re-evaluate bootstrapping algorithms in

*This section is adapted from: **Ruder, S.** and Plank, B. (2018). Strong Baselines for Neural Semi-supervised Learning under Domain Shift. In *Proceedings of ACL 2018*.

the context of DNNs. These are general-purpose semi-supervised algorithms that treat the model as a black box and can thus be used easily—with a few additions—with the current generation of NLP models. Many of these methods, though, were originally developed with in-domain performance in mind, so their effectiveness in a domain adaptation setting remains unexplored.

In particular, we re-evaluate three traditional bootstrapping methods, self-training [Yarowsky, 1995], tri-training [Zhou and Li, 2005], and tri-training with disagreement [Søgaard, 2010] for neural network-based approaches on *two* NLP tasks with different characteristics, namely, a sequence prediction and a classification task (POS tagging and sentiment analysis). We evaluate the methods across multiple domains on two well-established benchmarks, without taking any further task-specific measures, and compare to the best results published in the literature.

We make the somewhat surprising observation that classic tri-training outperforms task-agnostic state-of-the-art semi-supervised learning [Laine and Aila, 2017] and recent neural adaptation approaches [Ganin et al., 2016, Saito et al., 2017].

In addition, we propose *multi-task tri-training*, which reduces the main deficiency of tri-training, namely its time and space complexity. It establishes a new state of the art on unsupervised domain adaptation for sentiment analysis but it is outperformed by classic tri-training for POS tagging.

Contributions Our contributions are: a) We propose a novel multi-task tri-training method. b) We show that tri-training can serve as a strong and robust semi-supervised learning baseline for the current generation of NLP models. c) We perform an extensive evaluation of bootstrapping[1] algorithms compared to state-of-the-art approaches on two benchmark datasets. d) We shed light on the task and data characteristics that yield the best performance for each model.

4.2.2 Neural bootstrapping methods

We first introduce three classic bootstrapping methods, self-training, tri-training, and tri-training with disagreement and detail how they can be used with neural networks. For in-depth details we refer the reader to [Abney, 2007, Chapelle et al., 2006, Zhu and Goldberg, 2009]. We introduce our novel multi-task tri-training method in §4.2.2.3.

[1]We use the term bootstrapping as used in the semi-supervised learning literature [Zhu, 2005], which should not be confused with the statistical procedure of the same name [Efron and Tibshirani, 1994].

4.2.2.1 Self-training

Self-training [Yarowsky, 1995, McClosky et al., 2006b] is one of the earliest and simplest bootstrapping approaches. In essence, it leverages the model's own predictions on unlabeled data to obtain additional information that can be used during training. Typically the most confident predictions are taken at face value, as detailed next.

Self-training trains a model m on a labeled training set L and an unlabeled data set U. At each iteration, the model provides predictions $m(x)$ in the form of a probability distribution over classes for all unlabeled examples x in U. If the probability assigned to the most likely class is higher than a predetermined threshold τ, x is added to the labeled examples with $p(x) = \arg\max m(x)$ as pseudo-label. This instantiation is the most widely used and shown in Algorithm 1.

Algorithm 1 Self-training [Abney, 2007]

```
repeat
    m ← train_model(L)
    for x ∈ U do
        if max m(x) > τ then
            L ← L ∪ {(x, p(x))}
until no more predictions are confident
```

Calibration It is well-known that output probabilities in neural networks are poorly calibrated [Guo et al., 2017]. Using a fixed threshold τ is thus not the best choice. While the *absolute* confidence value is inaccurate, we can expect that the *relative* order of confidences is more robust.

For this reason, we select the top n unlabeled examples that have been predicted with the highest confidence after every epoch and add them to the labeled data. This is one of the many variants for self-training, called *throttling* [Abney, 2007]. We empirically confirm that this outperforms the classic selection in our experiments.

Online learning In contrast to many classic algorithms, DNNs are trained online by default. We compare training setups and find that training until convergence on labeled data and then training until convergence using self-training performs best.

Classic self-training has shown mixed success. In parsing it proved successful only with small datasets [Reichart and Rappoport, 2007] or when a generative component is used together with a reranker in high-data conditions [McClosky et al., 2006b, Suzuki and Isozaki, 2008]. Some success was achieved with careful task-specific data selection [Petrov and McDonald, 2012], while others report limited success on a variety of NLP tasks [Plank

and van Noord, 2011, Van Asch and Daelemans, 2016, van der Goot et al., 2017]. Its main downside is that the model is not able to correct its own mistakes and errors are amplified, an effect that is increased under domain shift.

4.2.2.2 Tri-training

Tri-training [Zhou and Li, 2005] is a classic method that reduces the bias of predictions on unlabeled data by utilizing the agreement of three independently trained models. Tri-training (cf. Algorithm 2) first trains three models m_1, m_2, and m_3 on bootstrap samples of the labeled data L. An unlabeled data point is added to the training set of a model m_i if the other two models m_j and m_k agree on its label. Training stops when the classifiers do not change anymore.

Algorithm 2 Tri-training [Zhou and Li, 2005]

1: **for** $i \in \{1..3\}$ **do**
2: $\quad S_i \leftarrow bootstrap_sample(L)$
3: $\quad m_i \leftarrow train_model(S_i)$
4: **repeat**
5: $\quad$ **for** $i \in \{1..3\}$ **do**
6: $\quad\quad L_i \leftarrow \emptyset$
7: $\quad\quad$ **for** $x \in U$ **do**
8: $\quad\quad\quad$ **if** $p_j(x) = p_k(x)(j, k \neq i)$ **then**
9: $\quad\quad\quad\quad L_i \leftarrow L_i \cup \{(x, p_j(x))\}$
$\quad\quad m_i \leftarrow train_model(L \cup L_i)$
10: **until** none of m_i changes
11: apply majority vote over m_i

Tri-training *with disagreement* [Søgaard, 2010] is based on the intuition that a model should only be strengthened in its weak points and that the labeled data should not be skewed by easy data points. In order to achieve this, it adds a simple modification to the original algorithm (altering line 8 in Algorithm 2), requiring that for an unlabeled data point on which m_j and m_k *agree*, the other model m_i *disagrees* on the prediction. Tri-training with disagreement is more data-efficient than tri-training and has achieved competitive results on part-of-speech tagging [Søgaard, 2010].

Sampling unlabeled data Both tri-training and tri-training with disagreement can be very expensive in their original formulation as they require to produce predictions for each of the three models on all unlabeled data samples, which can be in the millions in realistic applications. We thus propose to sample a number of unlabeled examples at every epoch. For all traditional bootstrapping approaches we sample 10k candidate instances in each epoch. For the neural approaches we use a linearly growing candidate

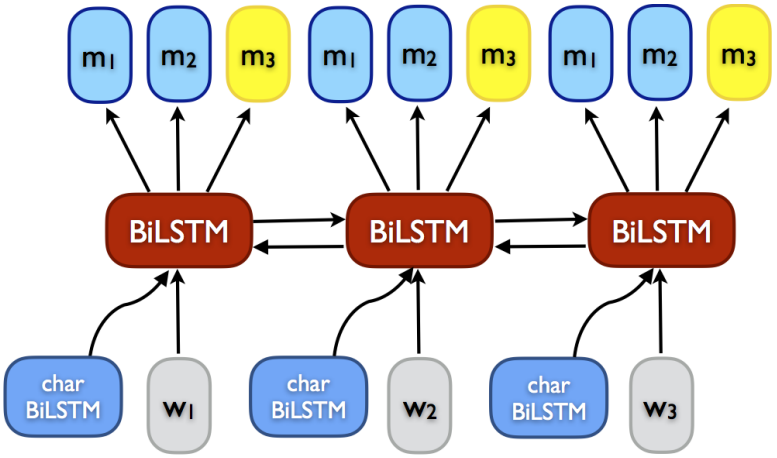


FIGURE 4.2: The Multi-task Tri-training model

sampling scheme proposed by Saito et al. [2017], increasing the candidate pool size as the models become more accurate.

Confidence thresholding Similar to self-training, we can introduce an additional requirement that pseudo-labeled examples are only added if the probability of the prediction of at least one model is higher than some threshold τ. We did not find this to outperform prediction without threshold for traditional tri-training, but thresholding proved essential for our method (§4.2.2.3).

The most important condition for tri-training and tri-training with disagreement is that the models are diverse. Typically, bootstrap samples are used to create this diversity [Zhou and Li, 2005, Søgaard, 2010]. However, training separate models on bootstrap samples of a potentially large amount of training data is expensive and takes a lot of time. This drawback motivates our approach.

4.2.2.3 Multi-task tri-training

In order to reduce both the time and space complexity of tri-training, we propose Multi-task Tri-training (MT-Tri). MT-Tri leverages insights from multi-task learning (MTL) [Caruana, 1993] to share knowledge across models and accelerate training. Rather than storing and training each model separately, we propose to share the parameters of the models and train them jointly using MTL.[2] All models thus collaborate on learning a joint representation, which improves convergence.

[2]Note: we use the term multi-task learning here albeit all tasks are of the same kind, similar to work on multi-lingual modeling treating each language (but same label space) as separate task e.g., [Fang and Cohn, 2017]. It is interesting to point out that our model is further doing implicit multi-view learning by way of the orthogonality constraint.

The output softmax layers are model-specific and are only updated for the input of the respective model. We show the model in Figure 4.2 (as instantiated for POS tagging). As the models leverage a joint representation, we need to ensure that the features used for prediction in the softmax layers of the different models are as diverse as possible, so that the models can still learn from each other's predictions. In contrast, if the parameters in all output softmax layers were the same, the method would degenerate to self-training.

To guarantee diversity, we introduce an orthogonality constraint [Bousmalis et al., 2016] as an additional loss term, which we define as follows:

$$\mathcal{L}_{orth} = \|\mathbf{W}_{m_1}^{\top}\mathbf{W}_{m_2}\|_F^2 \tag{4.2}$$

where $|\ \cdot\ \|_F^2$ is the squared Frobenius norm and $\mathbf{W}_{m_1}$ and $\mathbf{W}_{m_2}$ are the softmax output parameters of the two source and pseudo-labeled output layers m_1 and m_2, respectively. The orthogonality constraint encourages the models not to rely on the same features for prediction. As enforcing pair-wise orthogonality between three matrices is not possible, we only enforce orthogonality between the softmax output layers of m_1 and m_2,[3] while m_3 is gradually trained to be more target-specific. We parameterize $\mathcal{L}_{orth}$ by γ=0.01 following Liu et al. [2017]. We do not further tune γ.

More formally, let us illustrate the model by taking the sequence prediction task (Figure 4.2) as illustration. Given an utterance with labels $y_1, .., y_n$, our Multi-task Tri-training loss consists of three task-specific (m_1, m_2, m_3) tagging loss functions (where $\vec{h}$ is the uppermost Bi-LSTM encoding):

$$\mathcal{L}(\boldsymbol{\theta}) = -\sum_{i}\sum_{1,..,n}\log P_{m_i}(y|\vec{h}) + \gamma\mathcal{L}_{orth} \tag{4.3}$$

In contrast to classic tri-training, we can train the multi-task model with its three model-specific outputs jointly and *without* bootstrap sampling on the labeled source domain data until convergence, as the orthogonality constraint enforces different representations between models m_1 and m_2. From this point, we can leverage the pair-wise agreement of two output layers to add pseudo-labeled examples as training data to the third model. We train the third output layer m_3 only on pseudo-labeled target instances in order to make tri-training more robust to a domain shift. For the final prediction, majority voting of all three output layers is used, which resulted in the best instantiation, together with confidence thresholding ($\tau = 0.9$, except for high-resource POS where $\tau = 0.8$ performed slightly better). We also experimented with using a domain-adversarial loss [Ganin

[3]We also tried enforcing orthogonality on a hidden layer rather than the output layer, but this did not help.

Algorithm 3 Multi-task Tri-training

$m \leftarrow train_model(L)$
repeat
  for $i \in \{1..3\}$ **do**
    $L_i \leftarrow \emptyset$
    for $x \in U$ **do**
      if $p_j(x) = p_k(x)(j, k \neq i)$ **then**
        $L_i \leftarrow L_i \cup \{(x, p_j(x))\}$
    if $i = 3$ **then** $m_i = train_model(L_i)$
    else $m_i \leftarrow train_model(L \cup L_i)$
until end condition is met
apply majority vote over m_i

et al., 2016] on the jointly learned representation, but found this not to help. The full pseudo-code is given in Algorithm 3.

Computational complexity The motivation for MT-Tri was to reduce the space and time complexity of tri-training. We thus give an estimate of its efficiency gains. MT-Tri is $\approx 3\times$ more space-efficient than regular tri-training; tri-training stores one set of parameters for each of the three models, while MT-Tri only stores one set of parameters (we use three output layers, but these make up a comparatively small part of the total parameter budget). In terms of time efficiency, tri-training first requires to train each of the models from scratch. The actual tri-training takes about the same time as training from scratch and requires a separate forward pass for each model, effectively training three independent models simultaneously. In contrast, MT-Tri only necessitates one forward pass as well as the evaluation of the two additional output layers (which takes a negligible amount of time) and requires about as many epochs as tri-training until convergence (see Table 4.9, second column) while adding fewer unlabeled examples per epoch (see Section 4.2.3.4). In our experiments, MT-Tri trained about 5-6$\times$ faster than traditional tri-training.

MT-Tri can be seen as a self-ensembling technique, where different variations of a model are used to create a stronger ensemble prediction. Recent approaches in this line are *snapshot ensembling* [Huang et al., 2017a] that ensembles models converged to different minima during a training run, *asymmetric tri-training* [Saito et al., 2017] (Asym) that leverages agreement on two models as information for the third, and *temporal ensembling* [Laine and Aila, 2017], which ensembles predictions of a model at different epochs. We tried to compare to temporal ensembling in our experiments, but were not able to obtain consistent results.[4] We compare to the closest most recent method, asymmetric

[4]We suspect that the sparse features in NLP and the domain shift might be detrimental to its unsupervised consistency loss.

tri-training [Saito et al., 2017]. It differs from ours in two aspects: a) ASYM leverages only pseudo-labels from data points on which m_1 and m_2 agree, and b) it uses only one task (m_3) as final predictor. In essence, our formulation of MT-Tri is closer to the original tri-training formulation (agreements on two provide pseudo-labels to the third) thereby incorporating more diversity.

4.2.3 Experiments

In order to ascertain which methods are robust across different domains, we evaluate on widely used unsupervised domain adaptation datasets for two tasks, a sequence labeling and a classification task. We choose the SANCL 2012 dataset for POS tagging and the Amazon Reviews dataset, which we already employed in Section 4.1 (see Table 4.1 for statistics).

4.2.3.1 POS tagging

For POS tagging we use the SANCL 2012 shared task dataset [Petrov and McDonald, 2012] and compare to the top results in both low and high-data conditions [Schnabel and Schütze, 2014, Yin et al., 2015]. Both are strong baselines, as the FLORS tagger has been developed for this challenging dataset and it is based on contextual distributional features (excluding the word's identity), and hand-crafted suffix and shape features (including some language-specific morphological features). We want to gauge to what extent we can adopt a nowadays fairly standard (but more lexicalized) general neural tagger.

Our POS tagging model is a state-of-the-art Bi-LSTM tagger [Plank, 2016] with word and 100-dim character embeddings. Word embeddings are initialized with the 100-dim Glove embeddings [Pennington et al., 2014]. The BiLSTM has one hidden layer with 100 dimensions. The base POS model is trained on WSJ with early stopping on the WSJ development set, using patience 2, Gaussian noise with $\sigma = 0.2$ and word dropout with $p = 0.25$ [Kiperwasser and Goldberg, 2016].

Regarding data, the source domain is the Ontonotes 4.0 release of the Penn treebank Wall Street Journal (WSJ) annotated for 48 fine-grained POS tags. This amounts to 30,060 labeled sentences. We use 100,000 WSJ sentences from 1988 as unlabeled data, following Schnabel and Schütze [2014].[5] As target data, we use the five SANCL domains (answers, emails, newsgroups, reviews, weblogs). We restrict the amount of unlabeled data for each SANCL domain to the first 100k sentences, and do not do any

[5]Note that our unlabeled data might slightly differ from theirs. We took the first 100k sentences from the 1988 WSJ dataset from the BLLIP 1987-89 WSJ Corpus Release 1.

pre-processing. We consider the development set of ANSWERS as our only target dev set to set hyperparameters. This may result in suboptimal per-domain settings but better resembles an unsupervised adaptation scenario.

4.2.3.2 Sentiment analysis

For sentiment analysis, we evaluate on the Amazon reviews dataset [Blitzer et al., 2006]. Reviews with 1 to 3 stars are ranked as negative, while reviews with 4 or 5 stars are ranked as positive. The dataset consists of four domains, yielding 12 adaptation scenarios. We use the same pre-processing and architecture as used in [Ganin et al., 2016, Saito et al., 2017]: 5,000-dimensional tf-idf weighted unigram and bigram features as input; 2k labeled source samples and 2k unlabeled target samples for training, 200 labeled target samples for validation, and between 3k-6k samples for testing. The model is an MLP with one hidden layer with 50 dimensions, sigmoid activations, and a softmax output. We compare against the Variational Fair Autoencoder (VFAE) [Louizos et al., 2015] model and domain-adversarial neural networks (DANN) [Ganin et al., 2016].

4.2.3.3 Baselines

Besides comparing to the top results published on both datasets, we include the following baselines:

a) the task model trained on the source domain;
b) self-training (Self);
c) tri-training (Tri);
d) tri-training with disagreement (Tri-D); and
e) asymmetric tri-training [Saito et al., 2017].

Our proposed model is multi-task tri-training (MT-Tri). We implement our models in DyNet [Neubig et al., 2017]. Reporting single evaluation scores might result in biased results [Reimers and Gurevych, 2017]. Throughout the paper, we report mean accuracy and standard deviation over five runs for POS tagging and over ten runs for sentiment analysis. Significance is computed using bootstrap test. The code for all experiments is released at: `https://github.com/bplank/semi-supervised-baselines`.

4.2.3.4 Results

Sentiment analysis We show results for sentiment analysis for all 12 domain adaptation scenarios in Figure 4.3. For clarity, we also show the accuracy scores averaged across

each target domain as well as a global macro average in Table 4.8. The domains are Books (B), DVD (DVD), Electronics (E), and Kitchen (K). Results for VFAE, DANN, and Asym (indicated with a * in Table 4.8 are from Saito et al. [2017]).

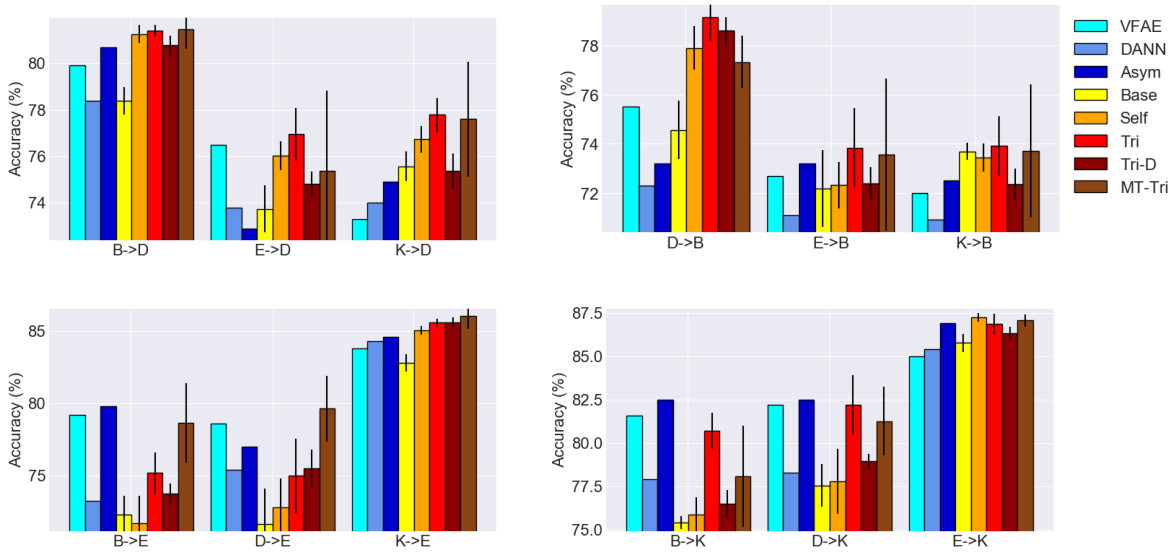


FIGURE 4.3: Average results for unsupervised domain adaptation on the Amazon dataset.

Model	D	B	E	K	Avg
VFAE*	76.57	73.40	80.53	82.93	78.36
DANN*	75.40	71.43	77.67	80.53	76.26
Asym*	76.17	72.97	80.47	**83.97**	78.39
Src	75.91	73.47	75.61	79.58	76.14
Self	78.00	74.55	76.54	80.30	77.35
Tri	**78.72**	**75.64**	78.60	83.26	79.05
Tri-D	76.99	74.44	78.30	80.59	77.58
MT-Tri	78.14	74.86	**81.45**	82.14	**79.15**

TABLE 4.8: Average accuracy scores for each SA target domain.

Self-training achieves surprisingly good results but is not able to compete with tri-training. Tri-training with disagreement is only slightly better than self-training, showing that the disagreement component might not be useful when there is a strong domain shift. Tri-training achieves the best average results on two target domains and clearly outperforms the state of the art on average.

MT-Tri finally outperforms the state of the art on 3/4 domains, and even slightly traditional tri-training, resulting in the overall best method. This improvement is mainly due to the B→E and D→E scenarios, on which tri-training struggles. These domain pairs are among those with the highest $\mathcal{A}$-distance [Blitzer et al., 2007], which highlights that tri-training has difficulty dealing with a strong shift in domain. Our method is able to

mitigate this deficiency by training one of the three output layers only on pseudo-labeled target domain examples.

In addition, MT-Tri is more efficient as it adds a smaller number of pseudo-labeled examples than tri-training at every epoch. For sentiment analysis, tri-training adds around 1800-1950/2000 unlabeled examples at every epoch, while MT-Tri only adds around 100-300 in early epochs. This shows that the orthogonality constraint is useful for inducing diversity. In addition, adding fewer examples poses a smaller risk of swamping the learned representations with useless signals and is more akin to fine-tuning, the standard method for supervised domain adaptation Howard and Ruder [2018].

We observe an asymmetry in the results between some of the domain pairs, e.g. B→D and D→B. We hypothesize that the asymmetry may be due to properties of the data and that the domains are relatively far apart e.g., in terms of $\mathcal{A}$-distance. In fact, asymmetry in these domains is already reflected in the results of Blitzer et al. [2007] and is corroborated in the results for asymmetric tri-training [Saito et al., 2017] and our method.

We note a weakness of this dataset is high variance. Existing approaches only report the mean, which makes an objective comparison difficult. For this reason, we believe it is essential to evaluate proposed approaches also on other tasks.

POS tagging Results for tagging in the low-data regime (10% of WSJ) are given in Table 4.9. Avg means the average over the 5 SANCL domains. The hyperparameter *ep* (epochs) is tuned on the validation set of the Answers domain. μ_{pseudo} indicates the average amount of added pseudo-labeled data. The results for FLORS Batch (u:big) are from [Yin et al., 2015] (see §4.2.3).

		Target domains							
Model	*ep*	**Answers**	**Emails**	**Newsgroups**	**Reviews**	**Weblogs**	**Avg**	**WSJ**	μ_{pseudo}
Src (+glove)		87.63 ±.37	86.49 ±.35	**88.60** ±.22	90.12 ±.32	92.85 ±.17	89.14 ±.28	95.49 ±.09	—
Self	(5)	87.64 ±.18	86.58 ±.30	88.42 ±.24	90.03 ±.11	92.80 ±.19	89.09 ±.20	95.36 ±.07	.5k
Tri	(4)	88.42 ±.16	87.46 ±.20	87.97 ±.09	90.72 ±.14	93.40 ±.15	89.56 ±.16	95.94 ±.07	20.5k
Tri-D	(7)	**88.50** ±.04	**87.63** ±.15	88.12 ±.05	**90.76** ±.10	**93.51** ±.06	**89.70** ±.08	**95.99** ±.03	7.7K
Asym	(3)	87.81 ±.19	86.97 ±.17	87.74 ±.24	90.16 ±.17	92.73 ±.16	89.08 ±.19	95.55 ±.12	1.5k
MT-Tri	(4)	87.92 ±.18	87.20 ±.23	87.73 ±.37	90.27 ±.10	92.96 ±.07	89.21 ±.19	95.50 ±.06	7.6k
FLORS		89.71	88.46	89.82	92.10	94.20	90.86	95.80	—

TABLE 4.9: Accuracy scores on dev set of target domain for POS tagging for 10% labeled data.

Self-training does not work for the sequence prediction task. We report only the best instantiation (throttling with n=800). Our results contribute to negative findings regarding self-training [Plank and van Noord, 2011, Van Asch and Daelemans, 2016].

In the low-data setup, tri-training *with disagreement* works best, reaching an overall average accuracy of 89.70, closely followed by classic tri-training, and significantly

outperforming the baseline on 4/5 domains. The exception is newsgroups, a difficult domain with high OOV rate where none of the approches beats the baseline (see §4.2.3.4). Our proposed MT-Tri is better than asymmetric tri-training, but falls below classic tri-training. It beats the baseline significantly on only 2/5 domains (answers and emails). The FLORS tagger [Yin et al., 2015] fares better. Its contextual distributional features are particularly helpful on unknown word-tag combinations (see §4.2.3.4), which is a limitation of the lexicalized generic bi-LSTM tagger.

We show results for the high-data setup in Table 4.10 where models were trained on the full source data. Values for methods with * are from Schnabel and Schütze [2014]. Results are similar.

	Target domains dev sets					**Avg on**	
Model	**Answers**	**Emails**	**Newsgroups**	**Reviews**	**Weblogs**	**targets**	**WSJ**
TnT*	88.55	88.14	88.66	90.40	93.33	89.82	95.75
Stanford*	88.92	88.68	89.11	91.43	94.15	90.46	96.83
Src	88.84 ±.15	88.24 ±.12	89.45 ±.23	91.24 ±.03	93.92 ±.17	90.34 ±.14	96.69 ±.08
Tri	89.34 ±.18	88.83 ±.07	89.32 ±.21	91.62 ±.06	94.40 ±.06	90.70 ±.12	96.84 ±.04
Tri-D	89.35 ±.16	88.66 ±.09	89.29 ±.12	91.58 ±.05	94.32 ±.05	90.62 ±.09	96.85 ±.06
Src (+glove)	89.35 ±.16	88.55 ±.14	**90.12** ±.31	91.48 ±.15	94.48 ±.07	90.80 ±.17	96.90 ±.04
Tri	**90.00** ±.03	**89.06** ±.16	90.04 ±.25	**91.98** ±.11	**94.74** ±.06	**91.16** ±.12	**96.99** ±.02
Tri-D	89.80 ±.19	88.85 ±.10	90.03 ±.22	**91.98** ±.09	94.70 ±.05	91.01 ±.13	96.95 ±.05
Asym	89.51 ±.15	88.47 ±.19	89.26 ±.16	91.60 ±.20	94.28 ±.15	90.62 ±.17	96.56 ±.01
MT-Tri	89.45 ±.05	88.65 ±.04	89.40 ±.22	91.63 ±.23	94.41 ±.05	90.71 ±.12	97.37 ±.07
FLORS*	90.30	89.44	90.86	92.95	94.71	91.66	96.59
	Target domains test sets					**Avg on**	
Model	**Answers**	**Emails**	**Newsgroups**	**Reviews**	**Weblogs**	**targets**	**WSJ**
TnT*	89.36	87.38	90.85	89.67	91.37	89.73	96.57
Stanford*	89.74	87.77	91.25	90.30	92.32	90.28	97.43
Src (+glove)	90.43 ±.13	87.95 ±.18	91.83 ±.20	90.04 ±.11	92.44 ±.14	90.54 ±.15	**97.50** ±.03
Tri	**91.21** ±.06	**88.30** ±.19	**92.18** ±.19	**90.06** ±.10	**92.85** ±.02	**90.92** ±.11	97.45 ±.03
Asym	90.62 ±.26	87.71 ±.07	91.40 ±.05	89.89 ±.22	92.37 ±.27	90.39 ±.17	97.19 ±.03
MT-Tri	90.53 ±.15	87.90 ±.07	91.45 ±.19	89.77 ±.26	92.35 ±.09	90.40 ±.15	97.37 ±.07
FLORS*	91.17	88.67	92.41	92.25	93.14	91.53	97.11

Table 4.10: Accuracy for POS tagging on the dev and test sets of the SANCL domains.

Disagreement, however, is only favorable in the low-data setups; the effect of avoiding easy points no longer holds in the full data setup. Classic tri-training is the best method. In particular, traditional tri-training is complementary to word embedding initialization, pushing the non-pre-trained baseline to the level of Src with Glove initalization. Tri-training pushes performance even further and results in the best model, significantly outperforming the baseline again in 4/5 cases, and reaching FLORS performance on weblogs. Multi-task tri-training is often slightly more effective than asymmetric tri-training [Saito et al., 2017]; however, improvements for both are not robust across domains, sometimes performance even drops. The model likely is too simplistic for such a high-data POS setup, and exploring shared-private models might prove more fruitful [Liu et al., 2017]. On the test sets, tri-training performs consistently the best.

	Ans	**Email**	**Newsg**	**Rev**	**Webl**
% unk tag	0.25	0.80	0.31	0.06	0.0
% OOV	8.53	10.56	10.34	6.84	8.45
% UWT	2.91	3.47	2.43	2.21	1.46
Accuracy on OOV tokens					
Src	54.26	57.48	**61.80**	59.26	**80.37**
Tri	**55.53**	**59.11**	61.36	**61.16**	79.32
Asym	52.86	56.78	56.58	59.59	76.84
MT-Tri	52.88	57.22	57.28	58.99	77.77
Accuracy on unknown word-tag (UWT) tokens					
Src	**17.68**	**11.14**	**17.88**	**17.31**	**24.79**
Tri	16.88	10.04	17.58	16.35	23.65
Asym	17.16	10.43	17.84	16.92	22.74
MT-Tri	16.43	11.08	17.29	16.72	23.13
FLORS*	17.19	15.13	21.97	21.06	21.65

TABLE 4.11: Accuracy scores on dev sets for OOV and unknown word-tag (UWT) combinations.

POS analysis We analyze POS tagging accuracy with respect to word frequency[6] and unseen word-tag combinations (UWT) on the dev sets. Table 4.11 (top rows) provides percentage of unknown tags, OOVs and unknown word-tag (UWT) rate.

The SANCL dataset is overall very challenging: OOV rates are high (6.8-11% compared to 2.3% in WSJ), so is the unknown word-tag (UWT) rate (answers and emails contain 2.91% and 3.47% UWT compared to 0.61% on WSJ) and almost all target domains even contain unknown tags [Schnabel and Schütze, 2014] (unknown tags: ADD,GW,NFP,XX), except for weblogs. Email is the domain with the highest OOV rate and highest unknown-tag-for-known-words rate. We plot accuracy with respect to word frequency on email in Figure 4.4, analyzing how the three methods fare in comparison to the baseline on this difficult domain.

Regarding OOVs, the results in Table 4.11 (second part) show that classic tri-training outperforms the source model (trained on only source data) on 3/5 domains in terms of OOV accuracy, except on two domains with high OOV rate (newsgroups and weblogs). In general, we note that tri-training works best on OOVs and on low-frequency tokens, which is also shown in Figure 4.4 (leftmost bins). Both other methods fall typically below the baseline in terms of OOV accuracy, but MT-Tri still outperforms Asym in 4/5 cases. Table 4.11 (last part) also shows that no bootstrapping method works well on unknown word-tag combinations. UWT tokens are very difficult to predict correctly using an unsupervised approach; the less lexicalized and more context-driven approach

[6]The binned log frequency was calculated with base 2 (bin 0 are OOVs, bin 1 are singletons and rare words etc).

FIGURE 4.4: Comparison of POS accuracy per binned log frequency for three methods in the email domain

taken by FLORS is clearly superior for these cases, resulting in higher UWT accuracies for 4/5 domains.

4.2.4 Related work

Learning under Domain Shift There is a large body of work on domain adaptation (§3.4). There is almost no work on bootstrapping approaches for recent neural NLP, in particular under domain shift. Tri-training is less studied, and only recently re-emerged in the vision community [Saito et al., 2017], albeit is not compared to classic tri-training.

Neural network ensembling Related work on self-ensembling approaches includes snapshot ensembling [Huang et al., 2017a] or temporal ensembling [Laine and Aila, 2017]. In general, the line between "explicit" and "implicit" ensembling [Huang et al., 2017a], like dropout [Srivastava et al., 2014] or temporal ensembling [Saito et al., 2017], is more fuzzy. As we noted earlier our multi-task learning setup can be seen as a form of self-ensembling.

Multi-task learning in NLP Neural networks are particularly well-suited for MTL allowing for parameter sharing [Caruana, 1993]. Recent NLP conferences witnessed a "tsunami" of deep learning papers [Manning, 2015], followed by a multi-task learning "wave" [Ruder, 2017]. For sentiment analysis we found tri-training and our MT-Tri model to outperform DANN [Ganin et al., 2016]. Our MT-Tri model lends itself well to shared-private models such as those proposed recently [Liu et al., 2017, Kim et al., 2017a], which extend upon [Ganin et al., 2016] by having separate source and target-specific encoders.

4.2.5 Summary

We re-evaluate a range of traditional general-purpose bootstrapping algorithms in the context of neural network approaches to semi-supervised learning under domain shift. For the two examined NLP tasks classic tri-training works the best and even outperforms a recent state-of-the-art method. The drawback of tri-training it its time and space complexity. We therefore propose a more efficient multi-task tri-training model, which outperforms both traditional tri-training and recent alternatives in the case of sentiment analysis. For POS tagging, classic tri-training is superior, performing especially well on OOVs and low frequency tokens, which suggests it is less affected by error propagation. Overall we emphasize the importance of comparing neural approaches to strong baselines and reporting results across several runs.

4.3 Conclusions

In this chapter, we have proposed algorithms that automatically select both relevant and informative labelled and unlabelled examples for domain adaptation and evaluated them on different extrinsic tasks. We have demonstrated that our methods outperform source domain models and the state-of-the-art in the majority of cases. In the next chapter, we will focus on methods that learn representations for heterogeneous, i.e. cross-lingual, domain adaptation. We will analyse the deficiencies of such representations and evaluate them on intrinsic tasks.

Chapter 5

Unsupervised and Weakly Supervised Cross-lingual Learning

Domain adaptation is most often applied in the unsupervised scenario where no labelled data in the target domain is available. Similarly, cross-lingual learning is most useful for low-resource languages where labelled data is rare or hard to obtain. For this reason, recent approaches that learn cross-lingual embeddings without any labelled data are particularly promising. However, transfer to low-resource languages is often difficult due to the dissimilarity between the language pairs. In this chapter, we will seek to gain a better understanding of such methods and characterize the distance between languages.

In Section 5.1, we first analyze the limitations of existing unsupervised cross-lingual word embedding methods. We propose a metric that measures the distance between monolingual embedding spaces. This metric can be seen as an analogue to the domain similarity metrics employed in the previous chapter (§4.1) for heterogeneous feature spaces and correlates well with downstream performance. In particular, we find that if spaces are too dissimilar such as between distant language pairs, e.g. English and Finnish, unsupervised approaches completely fail to produce an alignment. We propose a weakly supervised method that overcomes this weakness.

We take a latent variable view of cross-lingual word embedding models in Section 5.2, viewing the existing model by Artetxe et al. [2017] as a latent variable model. In a similar vein to the multi-task tri-training method presented in the previous chapter (§4.2), we propose a model that combines both traditional and current neural approaches. As a latent variable model, we can view the alignment of words with their translations as a combinatorial optimization problem, allowing us to use an efficient algorithm to solve it. We show that our approach outperforms the state-of-the-art and particularly performs well on low-resource language pairs.

5.1 The Limitations of Unsupervised Bilingual Dictionary Induction

* Unsupervised machine translation—i.e., not assuming *any* cross-lingual supervision signal, whether a dictionary, translations, or comparable corpora—seems impossible, but nevertheless, [Lample et al., 2018a] recently proposed a fully unsupervised machine translation model. The model relies heavily on an adversarial, unsupervised alignment of word embedding spaces for *bilingual dictionary induction* [Conneau et al., 2018a], which we examine here. Our results identify the limitations of current unsupervised MT: unsupervised bilingual dictionary induction performs much worse on morphologically rich languages that are not dependent marking, when monolingual corpora from different domains or different embedding algorithms are used. We show that a simple trick, exploiting a weak supervision signal from identical words, enables more robust induction, and establish a near-perfect correlation between unsupervised bilingual dictionary induction performance and a previously unexplored graph similarity metric.

5.1.1 Introduction

Cross-lingual word representations enable us to reason about word meaning in multilingual contexts and facilitate cross-lingual transfer [Ruder et al., 2019b]. Early cross-lingual word embedding models relied on large amounts of parallel data [Klementiev et al., 2012b, Mikolov et al., 2013c], but more recent approaches have tried to minimize the amount of supervision necessary [Vulić and Korhonen, 2016, Levy et al., 2017, Artetxe et al., 2017]. Some researchers have even presented *unsupervised* methods that do not rely on any form of cross-lingual supervision at all [Barone, 2016, Conneau et al., 2018a, Zhang et al., 2017].

Unsupervised cross-lingual word embeddings hold promise to induce bilingual lexicons and machine translation models in the absence of dictionaries and translations [Barone, 2016, Zhang et al., 2017, Lample et al., 2018a], and would therefore be a major step toward machine translation to, from, or even between low-resource languages.

Unsupervised approaches to learning cross-lingual word embeddings are based on the assumption that monolingual word embedding graphs are approximately isomorphic, that is, after removing a small set of vertices (words) [Mikolov et al., 2013c, Barone, 2016, Zhang et al., 2017, Conneau et al., 2018a]. In the words of Barone [2016]:

*This section is adapted from: Søgaard, A., **Ruder, S.** and Vulić, I. (2018). On the Limitations of Unsupervised Bilingual Dictionary Induction. In *Proceedings of ACL 2018*. Anders focused on the eigenvector similarity metric. Sebastian and Ivan focused on the analyses.

> *. . . we hypothesize that, if languages are used to convey thematically similar information in similar contexts, these random processes should be approximately isomorphic between languages, and that this isomorphism can be learned from the statistics of the realizations of these processes, the monolingual corpora, in principle without any form of explicit alignment.*

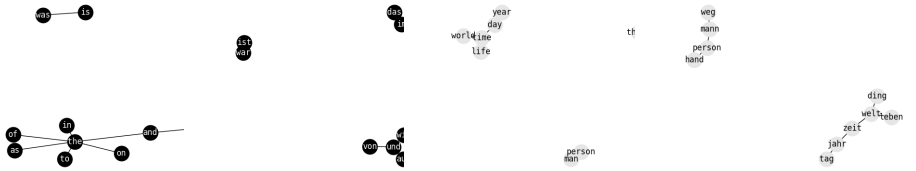
Our results indicate this assumption is not true in general, and that approaches based on this assumption have important limitations.

Contributions We focus on the recent state-of-the-art unsupervised model of Conneau et al. [2018a].[1] Our contributions are: (a) In §5.1.2, we show that the monolingual word embeddings used in Conneau et al. [2018a] are *not* approximately isomorphic, using the VF2 algorithm [Cordella et al., 2001] and we therefore introduce a metric for quantifying the similarity of word embeddings, based on Laplacian eigenvalues. (b) In §5.1.3, we identify circumstances under which the unsupervised bilingual dictionary induction (BDI) algorithm proposed in Conneau et al. [2018a] does not lead to good performance. (c) We show that a simple trick, exploiting a weak supervision signal from words that are identical across languages, makes the algorithm much more robust. Our main finding is that the performance of unsupervised BDI depends heavily on all three factors: the language pair, the comparability of the monolingual corpora, and the parameters of the word embedding algorithms.

5.1.2 How similar are embeddings across languages?

As mentioned, recent work focused on unsupervised BDI assumes that monolingual word embedding spaces (or at least the subgraphs formed by the most frequent words) are approximately isomorphic. In this section, we show, by investigating the nearest neighbor graphs of word embedding spaces, that word embeddings are far from isomorphic. We therefore introduce a method for computing the similarity of non-isomorphic graphs. In §5.1.4.7, we correlate our similarity metric with performance on unsupervised BDI.

[1] Our motivation for this is that Artetxe et al. [2017] use small dictionary seeds for supervision, and Barone [2016] seems to obtain worse performance than Conneau et al. [2018a]. Our results should extend to Barone [2016] and Zhang et al. [2017], which rely on very similar methodology.



(A) Top 10 most frequent English words

(B) German translations of top 10 most frequent English words

(C) Top 10 most frequent English nouns

(D) German translations of top 10 most frequent English nouns

FIGURE 5.1: Nearest neighbor graphs of English words and German translations.

Isomorphism To motivate our study, we first establish that word embeddings are far from graph isomorphic[2]—even for two closely related languages, English and German, and using embeddings induced from comparable corpora (Wikipedia) with the same hyper-parameters.

If we take the top k most frequent words in English, and the top k most frequent words in German, and build nearest neighbor graphs for English and German using the monolingual word embeddings used in Conneau et al. [2018a], the graphs are of course very different. This is, among other things, due to German case and the fact that *the* translates into *der*, *die*, and *das*, but unsupervised alignment does not have access to this kind of information. *Even* if we consider the top k most frequent English words *and their translations* into German, the nearest neighbor graphs are not isomorphic. Figure 5.1a-b shows the nearest neighbor graphs of the top 10 most frequent English words on Wikipedia, and their German translations.

Word embeddings are particularly good at capturing relations between nouns, but even if we consider the top k most frequent English *nouns* and their translations, the graphs are not isomorphic; see Figure 5.1c-d. We take this as evidence that word embeddings are not approximately isomorphic across languages. We also ran graph isomorphism checks on 10 random samples of frequent English nouns and their translations into Spanish, and only in 1/10 of the samples were the corresponding nearest neighbor graphs isomorphic.

Eigenvector similarity Since the nearest neighbor graphs are not isomorphic, even for frequent translation pairs in neighboring languages, we want to quantify the potential for unsupervised BDI using a metric that captures varying degrees of graph similarity. Eigenvalues are compact representations of global properties of graphs, and we introduce a spectral metric based on Laplacian eigenvalues [Shigehalli and Shettar, 2011]

[2]Two graphs that contain the same number of graph vertices connected in the same way are said to be isomorphic. In the context of weighted graphs such as word embeddings, a weak version of this is to require that the underlying nearest neighbor graphs for the most frequent k words are isomorphic.

that quantifies the extent to which the nearest neighbor graphs are *isospectral*. Note that (approximately) isospectral graphs need not be (approximately) isomorphic, but (approximately) isomorphic graphs are always (approximately) isospectral [Gordon et al., 1992]. Let A_1 and A_2 be the adjacency matrices of the nearest neighbor graphs G_1 and G_2 of our two word embeddings, respectively. Let $L_1 = D_1 - A_1$ and $L_2 = D_2 - A_2$ be the Laplacians of the nearest neighbor graphs, where D_1 and D_2 are the corresponding diagonal matrices of degrees. We now compute the eigensimilarity of the Laplacians of the nearest neighbor graphs, L_1 and L_2. For each graph, we find the smallest k such that the sum of the k largest Laplacian eigenvalues is $< 90\%$ of the Laplacian eigenvalues. We take the smallest k of the two, and use the sum of the squared differences between the largest k Laplacian eigenvalues Δ as our similarity metric.

$$\Delta = \sum_{i=1}^{k} (\lambda_{1_i} - \lambda_{2_i})^2 \tag{5.1}$$

where k is chosen s.t.

$$\min_j \{ \frac{\sum_{i=1}^{k} \lambda_{ji}}{\sum_{i=1}^{n} \lambda_{ji}} > 0.9 \}$$

Note that $\Delta = 0$ means the graphs are isospectral, and the metric goes to infinite. Thus, the higher Δ is, the *less* similar the graphs (i.e., their Laplacian spectra). We discuss the correlation between unsupervised BDI performance and approximate isospectrality or eigenvector similarity in §5.1.4.7.

5.1.3 Unsupervised cross-lingual learning

5.1.3.1 Learning scenarios

Unsupervised neural machine translation relies on BDI using cross-lingual embeddings [Lample et al., 2018a, Artetxe et al., 2018b], which in turn relies on the assumption that word embedding graphs are approximately isomorphic. The work of Conneau et al. [2018a], which we focus on here, also makes several implicit assumptions that may or may not be necessary to achieve such isomorphism, and which may or may not scale to low-resource languages. The algorithms are not intended to be limited to learning scenarios where these assumptions hold, but since they do in the reported experiments, it is important to see to what extent these assumptions are necessary for the algorithms to produce useful embeddings or dictionaries.

We focus on the work of Conneau et al. [2018a], who present a fully unsupervised approach to aligning monolingual word embeddings, induced using *fastText* [Bojanowski et al.,

2017]. We describe the learning algorithm in §5.1.3.2. Conneau et al. [2018a] consider a specific set of learning scenarios:

(a) The authors work with the following **languages**: English-{French, German, Chinese, Russian, Spanish}. These languages, except French, are dependent marking (Table 5.1).[3] We evaluate Conneau et al. [2018a] on (English to) Estonian (ET), Finnish (FI), Greek (EL), Hungarian (HU), Polish (PL), and Turkish (TR) in §5.1.4.2, to test whether the selection of languages in the original study introduces a bias.

(b) The monolingual corpora in their experiments are comparable; Wikipedia corpora are used, except for an experiment in which they include Google Gigawords. We evaluate across different **domains**, i.e., on all combinations of Wikipedia, EuroParl, and the EMEA medical corpus, in §5.1.4.3. We believe such scenarios are more realistic for low-resource languages.

(c) The monolingual embedding models are induced using the same **algorithms** with the same **hyper-parameters**. We evaluate Conneau et al. [2018a] on pairs of embeddings induced with different hyper-parameters in §5.1.4.4. While keeping hyper-parameters fixed is always possible, it is of practical interest to know whether the unsupervised methods work on any set of pre-trained word embeddings.

We also investigate the sensitivity of unsupervised BDI to the **dimensionality** of the monolingual word embeddings in §5.1.4.5. The motivation for this is that dimensionality reduction will alter the geometric shape and remove characteristics of the embedding graphs that are important for alignment; but on the other hand, lower dimensionality introduces regularization, which will make the graphs more similar. Finally, in §5.1.4.6, we investigate the impact of different types of query **test words** on performance, including how performance varies across part-of-speech word classes and on shared vocabulary items.

5.1.3.2 Summary of Conneau et al. [2018a]

We now introduce the method of Conneau et al. [2018a].[4] The approach builds on existing work on learning a mapping between monolingual word embeddings [Mikolov et al., 2013c, Xing et al., 2015] and consists of the following steps: 1) **Monolingual word embeddings:** An off-the-shelf word embedding algorithm [Bojanowski et al., 2017] is used to learn source and target language spaces X and Y. 2) **Adversarial mapping:** A translation matrix W is learned between the spaces X and Y using adversarial techniques [Ganin et al., 2016]. A discriminator is trained to discriminate samples from the translated

[3] A dependent-marking language marks agreement and case more commonly on dependents than on heads.

[4] `https://github.com/facebookresearch/MUSE`

source space WX from the target space Y, while W is trained to prevent this. This, again, is motivated by the assumption that source and target language word embeddings are approximately isomorphic. 3) **Refinement (Procrustes analysis):** W is used to build a small bilingual dictionary of frequent words, which is pruned such that only bidirectional translations are kept [Vulić and Korhonen, 2016]. A new translation matrix W that translates between the spaces X and Y of these frequent word pairs is then induced by solving the Orthogonal Procrustes problem:

$$\begin{aligned} W^* &= \text{argmin}_W \|WX - Y\|_{\text{F}} = UV^\top \\ &\text{s.t. } U\Sigma V^\top = \text{SVD}(YX^\top) \end{aligned} \tag{5.2}$$

This step can be used iteratively by using the new matrix W to create new seed translation pairs. It requires frequent words to serve as reliable anchors for learning a translation matrix. In the experiments in Conneau et al. [2018a], as well as in ours, the iterative Procrustes refinement improves performance across the board. 4) **Cross-domain similarity local scaling** (CSLS) is used to expand high-density areas and condense low-density ones, for more accurate nearest neighbor calculation, CSLS reduces the hubness problem in high-dimensional spaces [Radovanović et al., 2010, Dinu et al., 2015]. It relies on the mean similarity of a source language embedding x to its K target language nearest neighbours ($K = 10$ suggested) $nn_1, \ldots, nn_K$:

$$mnn_T(x) = \frac{1}{K} \sum_{i=1}^{K} cos(x, nn_i) \tag{5.3}$$

where *cos* is the cosine similarity. $mnn_S(y)$ is defined in an analogous manner for any target language embedding y. $CSLS(x, y)$ is then calculated as follows:

$$2cos(x, y) - mnn_T(x) - mnn_S(y) \tag{5.4}$$

5.1.3.3 A simple supervised method

Instead of learning cross-lingual embeddings completely without supervision, we can extract inexpensive supervision signals by harvesting identically spelled words as in, e.g. [Artetxe et al., 2017, Smith et al., 2017]. Specifically, we use identically spelled words that occur in the vocabularies of both languages as bilingual seeds, without employing any additional transliteration or lemmatization/normalization methods. Using this seed dictionary, we then run the refinement step using Procrustes analysis of Conneau et al. [2018a].

5.1.4 Experiments

In the following experiments, we investigate the robustness of unsupervised cross-lingual word embedding learning, varying the language pairs, monolingual corpora, hyper-parameters, etc., to obtain a better understanding of when and why unsupervised BDI works.

We use bilingual dictionaries compiled by Conneau et al. [2018a] as gold standard, and adopt their evaluation procedure: each test set in each language consists of 1500 gold translation pairs. We rely on CSLS for retrieving the nearest neighbors, as it consistently outperformed the cosine similarity in all our experiments. Following a standard evaluation practice [Vulić and Moens, 2013, Mikolov et al., 2013c, Conneau et al., 2018a], we report *Precision at 1* scores (P@1): how many times one of the correct translations of a source word w is retrieved as the nearest neighbor of w in the target language.

5.1.4.1 Experimental setup

Our default experimental setup closely follows the setup of Conneau et al. [2018a]. For each language we induce monolingual word embeddings for all languages from their respective tokenized and lowercased Polyglot Wikipedias [Al-Rfou et al., 2013] using *fastText* [Bojanowski et al., 2017]. Only words with more than 5 occurrences are retained for training. Our *fastText* setup relies on skip-gram with negative sampling [Mikolov et al., 2013a] with standard hyper-parameters: bag-of-words contexts with the window size 2, 15 negative samples, subsampling rate 10^{-4}, and character n-gram length 3-6. All embeddings are 300-dimensional.

As we analyze the impact of various modeling assumptions in the following sections (e.g., domain differences, algorithm choices, hyper-parameters), we also train monolingual word embeddings using other corpora and different hyper-parameter choices. Quick summaries of each experimental setup are provided in the respective subsections.

5.1.4.2 Impact of language similarity

Conneau et al. [2018a] present results for several target languages: Spanish, French, German, Russian, Chinese, and Esperanto. All languages but Esperanto are isolating or exclusively concatenating languages from a morphological point of view. All languages but French are dependent-marking. Table 5.1 lists three important morphological properties of the languages involved in their/our experiments.[5]

[5]We use information from the World Atlas of Language Structures (WALS; `https://wals.info/`) as basis for our characterisation of the presented languages.

	Marking	**Type**	**# Cases**
English (EN)	dependent	isolating	None
French (FR)	mixed	fusional	None
German (DE)	dependent	fusional	4
Chinese (ZH)	dependent	isolating	None
Russian (RU)	dependent	fusional	6–7
Spanish (ES)	dependent	fusional	None
Estonian (ET)	mixed	agglutinative	10+
Finnish (FI)	mixed	agglutinative	10+
Greek (EL)	double	fusional	3
Hungarian (HU)	dependent	agglutinative	10+
Polish (PL)	dependent	fusional	6–7
Turkish (TR)	dependent	agglutinative	6–7

TABLE 5.1: Languages in Conneau et al. [2018a] (top) and in our experiments (lower half)

	Unsupervised (Adversarial)	**Supervised** (Identical)	**Similarity** (Eigenvectors)
EN-ES	81.89	**82.62**	2.07
EN-ET	00.00	**31.45**	6.61
EN-FI	00.09	**28.01**	7.33
EN-EL	00.07	**42.96**	5.01
EN-HU	45.06	**46.56**	3.27
EN-PL	46.83	**52.63**	2.56
EN-TR	32.71	**39.22**	3.14
ET-FI	**29.62**	24.35	3.98

TABLE 5.2: Bilingual dictionary induction scores (P@1×100%) of unsupervised and supervised methods, together with eigenvector similarities.

Agglutinative languages with mixed or double marking show more morphological variance with content words, and we speculate whether unsupervised BDI is challenged by this kind of morphological complexity. To evaluate this, we experiment with Estonian and Finnish, and we include Greek, Hungarian, Polish, and Turkish to see how their approach fares on combinations of these two morphological traits.

We show results in the left column of Table 5.2 using **a)** the unsupervised method with adversarial training; **b)** the supervised method with a bilingual seed dictionary consisting of identical words (shared between the two languages). The third columns lists eigenvector similarities between 10 randomly sampled source language nearest neighbor subgraphs of 10 nodes and the subgraphs of their translations, all from the benchmark dictionaries in Conneau et al. [2018a].

The results are quite dramatic. The approach achieves impressive performance for Spanish, one of the languages Conneau et al. [2018a] include in their paper. For the languages we add here, performance is less impressive. For the languages with dependent marking (Hungarian, Polish, and Turkish), P@1 scores are still reasonable, with Turkish

being slightly lower (0.327) than the others. However, for Estonian and Finnish, the method fails completely. Only in less than 1/1000 cases does a nearest neighbor search in the induced embeddings return a correct translation of a query word.[6]

The sizes of Wikipedias naturally vary across languages: e.g., *fastText* trains on approximately 16M sentences and 363M word tokens for Spanish, while it trains on 1M sentences and 12M words for Finnish. However, the difference in performance cannot be explained by the difference in training data sizes. To verify that near-zero performance in Finnish is not a result of insufficient training data, we have conducted another experiment using the large **Finnish WaC corpus** [Ljubešić et al., 2016] containing 1.7B words in total (this is similar in size to the English Polyglot Wikipedia). However, even with this large Finnish corpus, the model does not induce anything useful: P@1 equals 0.0.

We note that while languages with mixed marking may be harder to align, it seems unsupervised BDI is possible between similar, mixed marking languages. So while unsupervised learning fails for English-Finnish and English-Estonian, performance is reasonable and stable for the more similar Estonian-Finnish pair (Table 5.2). In general, unsupervised BDI, using the approach in Conneau et al. [2018a], seems challenged when pairing English with languages that are not isolating and do not have dependent marking.[7]

The promise of zero-supervision models is that we can learn cross-lingual embeddings even for low-resource languages. On the other hand, a similar distribution of embeddings requires languages to be similar. This raises the question whether we need fully unsupervised methods at all. In fact, our supervised method that relies on very naive supervision in the form of identically spelled words leads to competitive performance for similar language pairs and better results for dissimilar pairs. The fact that we can reach competitive and more robust performance with such a simple heuristic questions the true applicability of fully unsupervised approaches and suggests that it might often be better to rely on available weak supervision.

5.1.4.3 Impact of domain differences

Monolingual word embeddings used in Conneau et al. [2018a] are induced from Wikipedia, a near-parallel corpus. In order to assess the sensitivity of unsupervised BDI to the

[6]We note, though, that varying our random seed, performance for Estonian, Finnish, and Greek is sometimes (approximately 1 out of 10 runs) *on par* with Turkish. Detecting main causes and remedies for the inherent instability of adversarial training is one the most important avenues for future research.

[7]One exception here is French, which they include in their paper, but French arguably has a relatively simple morphology.

comparability and domain similarity of the monolingual corpora, we replicate the experiments in Conneau et al. [2018a] using combinations of word embeddings extracted from three different domains: **1)** parliamentary proceedings from EuroParl.v7 [Koehn, 2005], **2)** Wikipedia [Al-Rfou et al., 2013], and **3)** the EMEA corpus in the medical domain [Tiedemann, 2009]. We report experiments with three language pairs: English-{Spanish, Finnish, Hungarian}.

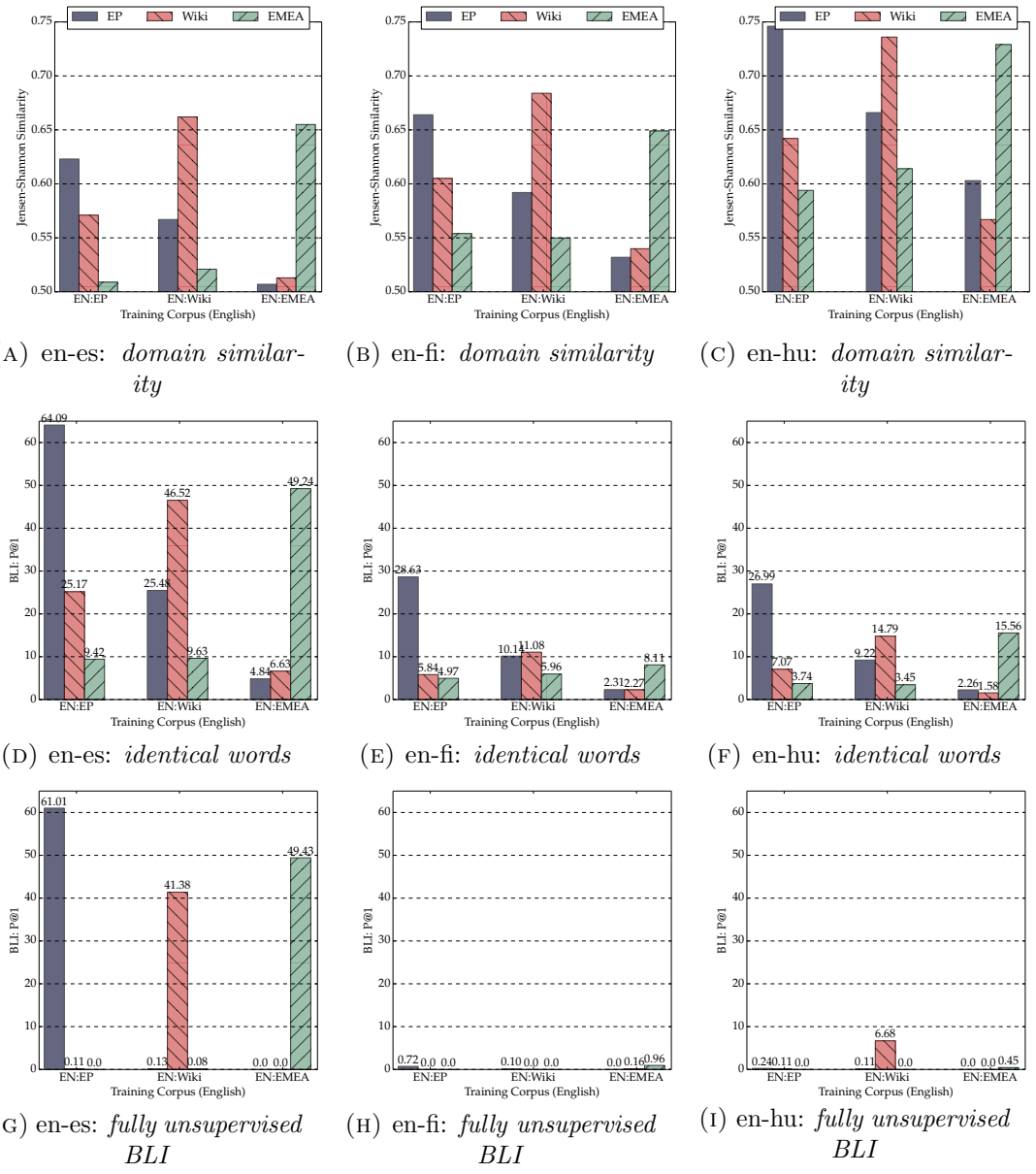
To control for the corpus size, we restrict each corpus in each language to 1.1M sentences in total (i.e., the number of sentences in the smallest, EMEA corpus). 300-dim *fastText* vectors are induced as in §5.1.4.1, retaining all words with more than 5 occurrences in the training data. For each pair of monolingual corpora, we compute their domain (dis)similarity by calculating the Jensen-Shannon divergence [El-Gamal, 1991], based on term distributions.[8] The domain similarities are displayed in Figures 5.2a–c.[9] In the top row (a)-(c), we show domain similarity (higher is more similar) computed as $dsim = 1 - JS$, where JS is Jensen-Shannon divergence; the middle row, (d)-(f), depicts the performance of the baseline BLI model which learns a linear mapping between two monolingual spaces based on a set of identical (i.e., shared) words; the bottom row, (g)-(i), displays the performance of the fully unsupervised BLI model relying on the distribution-level alignment and adversarial training. Both BLI models apply the Procrustes analysis and use CSLS to retrieve nearest neighbours.

We show the results of unsupervised BDI in Figures 5.2g–i. For Spanish, we see good performance in all three cases where the English and Spanish corpora are from the same domain. *When the two corpora are from different domains, performance is close to zero.* For Finnish and Hungarian, performance is always poor, suggesting that more data is needed, even when domains are similar. This is in sharp contrast with the results of our minimally supervised approach (Figures 5.2d–f) based on identical words, which achieves decent performance in many set-ups.

We also observe a strong decrease in P@1 for English-Spanish (from 81.19% to 46.52%) when using the smaller Wikipedia corpora. This result indicates the importance of procuring large monolingual corpora from similar domains in order to enable unsupervised dictionary induction. However, resource-lean languages, for which the unsupervised method was designed in the first place, cannot be guaranteed to have as large monolingual training corpora as available for English, Spanish or other major resource-rich languages.

[8] In order to get comparable term distributions, we translate the source language to the target language using the bilingual dictionaries provided by Conneau et al. [2018a] Conneau et al. [2018a].

[9] We also computed $\mathcal{A}$-distances [Blitzer et al., 2007] and confirmed that trends were similar.



(A) en-es: *domain similarity* (B) en-fi: *domain similarity* (C) en-hu: *domain similarity*

(D) en-es: *identical words* (E) en-fi: *identical words* (F) en-hu: *identical words*

(G) en-es: *fully unsupervised BLI* (H) en-fi: *fully unsupervised BLI* (I) en-hu: *fully unsupervised BLI*

FIGURE 5.2: Influence of language-pair *and* domain similarity on BLI performance, with three language pairs (en-es/fi/hu). **Top row, (a)-(c)**: Domain similarity (higher is more similar) computed as $dsim = 1 - JS$, where JS is Jensen-Shannon divergence; **Middle row, (d)-(f)**: baseline BLI model which learns a linear mapping between two monolingual spaces based on a set of identical (i.e., shared) words; **Bottom row, (g)-(i)**: fully unsupervised BLI model relying on the distribution-level alignment and adversarial training. Both BLI models apply the Procrustes analysis and use CSLS to retrieve nearest neighbours.

5.1.4.4 Impact of hyper-parameters

Conneau et al. [2018a] use the same hyper-parameters for inducing embeddings for all languages. This is of course always practically possible, but we are interested in seeing whether their approach works on pre-trained embeddings induced with possibly very different hyper-parameters. We focus on two hyper-parameters: context window-size

	English (skipgram, win=2, chn=3-6)	
	Spanish *(skipgram)*	**Spanish** *(cbow)*
==	81.89	00.00
≠ win=10	81.28	00.07
≠ chn=2-7	80.74	00.00
≠ win=10, chn=2-7	80.15	00.13

TABLE 5.3: Varying the underlying *fastText* algorithm and hyper-parameters.

(*win*) and the parameter controlling the number of n-gram features in the *fastText* model (*chn*), while at the same time varying the underlying algorithm: *skip-gram* vs. *cbow*. The results for English-Spanish are listed in Table 5.3. The first column lists differences in training configurations between English and Spanish monolingual embeddings.

The small variations in the hyper-parameters with the same underlying algorithm (i.e., using *skip-gram* or *cbow* for both EN and ES) yield only slight drops in the final scores. Still, the best scores are obtained with the same configuration on both sides. Our main finding here is that unsupervised BDI fails (even) for EN-ES when the two monolingual embedding spaces are induced by two different algorithms (see the results of the entire Spanish *cbow* column).[10] In sum, this means that *the unsupervised approach is unlikely to work on pre-trained word embeddings unless they are induced on same- or comparable-domain, reasonably-sized training data using the same underlying algorithm.*

5.1.4.5 Impact of dimensionality

We also perform an experiment on 40-dimensional monolingual word embeddings. This leads to reduced expressivity, and can potentially make the geometric shapes of embedding spaces harder to align; on the other hand, reduced dimensionality may also lead to less overfitting. We generally see worse performance (P@1 is 50.33 for Spanish, 21.81 for Hungarian, 20.11 for Polish, and 22.03 for Turkish) – but, very *interestingly*, we obtain *better performance* for Estonian (13.53), Finnish (15.33), and Greek (24.17) than we did with 300 dimensions. We hypothesize this indicates monolingual word embedding algorithms over-fit to some of the rarer peculiarities of these languages.

[10]We also checked if this result might be due to a lower-quality monolingual ES space. However, monolingual word similarity scores on available datasets in Spanish show performance comparable to that of Spanish *skip-gram* vectors: e.g., Spearman's ρ correlation is ≈ 0.7 on the ES evaluation set from SemEval-2017 Task 2 [Camacho-Collados et al., 2017].

	en-es	en-hu	en-fi
Noun	80.94	26.87	00.00
Verb	66.05	25.44	00.00
Adjective	85.53	53.28	00.00
Adverb	80.00	51.57	00.00
Other	73.00	53.40	00.00

TABLE 5.4: P@1 × 100% scores for query words with different parts-of-speech.

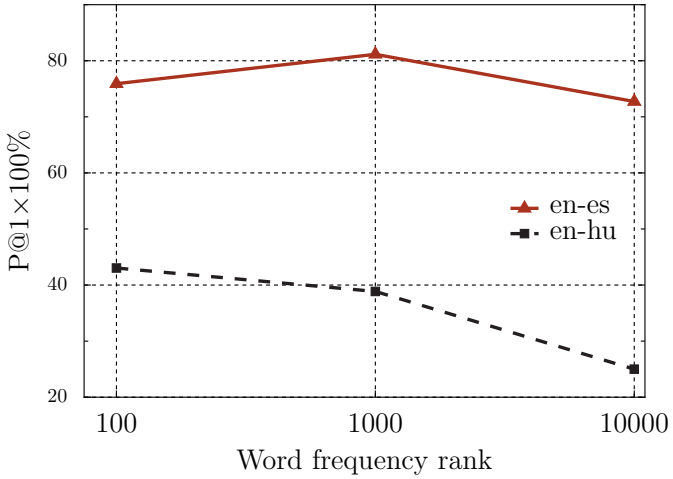


FIGURE 5.3: P@1 scores for EN-ES and EN-HU for queries with different frequency ranks.

5.1.4.6 Impact of evaluation procedure

BDI models are evaluated on a held-out set of query words. Here, we analyze the performance of the unsupervised approach across different parts-of-speech, frequency bins, and with respect to query words that have orthographically identical counterparts in the target language with the same or a different meaning.

Part-of-speech We show the impact of the part-of-speech of the query words in Table 5.4; again on a representative subset of our languages. The results indicate that performance on verbs is lowest across the board. This is consistent with research on distributional semantics and verb meaning [Schwartz et al., 2015, Gerz et al., 2016].

Frequency We also investigate the impact of the frequency of query words. We calculate the word frequency of English words based on Google's Trillion Word Corpus: query words are divided in groups based on their rank – i.e., the first group contains the top 100 most frequent words, the second one contains the 101th-1000th most frequent words, etc. – and plot performance (P@1) relative to rank in Figure 5.3. For EN-FI, P@1 was 0 across all frequency ranks. The plot shows sensitivity to frequency for HU, but less so for ES.

Spelling	Meaning	en-es	en-hu	en-fi
Same	Same	45.94	18.07	00.00
Same	Diff	39.66	29.97	00.00
Diff	Diff	62.42	34.45	00.00

TABLE 5.5: Scores (P@1 × 100%) for query words with same and different spellings and meanings.

Homographs Since we use identical word forms (homographs) for supervision, we investigated whether these are representative or harder to align than other words. Table 5.5 lists performance for three sets of query words: (a) source words that have homographs (words that are spelled the same way) with the same meaning (homonyms) in the target language, e.g., many proper names; (b) source words that have homographs that are not homonyms in the target language, e.g., many short words; and (c) other words. Somewhat surprisingly, words which have translations that are homographs, are associated with *lower* precision than other words. This is probably due to loan words and proper names, but note that using homographs as supervision for alignment, we achieve high precision for this part of the vocabulary *for free*.

5.1.4.7 Evaluating eigenvector similarity

Finally, in order to get a better understanding of the limitations of unsupervised BDI, we correlate the graph similarity metric described in §5.1.2 (right column of Table 5.2) with performance across languages (left column). Since we already established that the monolingual word embeddings are far from isomorphic—in contrast with the intuitions motivating previous work [Mikolov et al., 2013c, Barone, 2016, Zhang et al., 2017, Conneau et al., 2018a]—we would like to establish another diagnostic metric that identifies embedding spaces for which the approach in Conneau et al. [2018a] is likely to work. Differences in morphology, domain, or embedding parameters seem to be predictive of poor performance, but a metric that is independent of linguistic categorizations and the characteristics of the monolingual corpora would be more widely applicable. We plot the values in Table 5.2 in Figure 5.4. Recall that our graph similarity metric returns a value in the half-open interval $[0, \infty)$. The correlation between BDI performance and graph similarity is strong ($\rho \sim 0.89$).

5.1.5 Related work

Unsupervised cross-lingual learning Haghighi et al. [2008] were first to explore unsupervised BDI, using features such as context counts and orthographic substrings, and canonical correlation analysis. Recent approaches use adversarial learning [Goodfellow

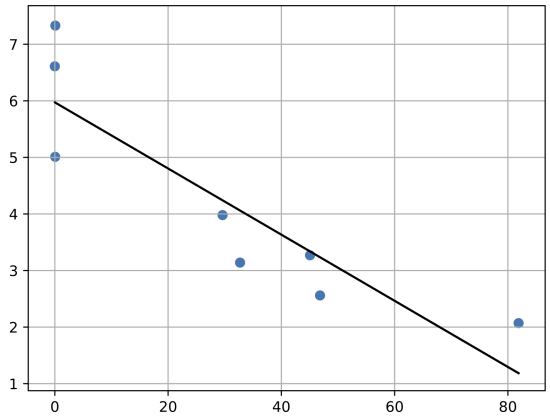


FIGURE 5.4: Strong correlation ($\rho = 0.89$) between BDI performance (x) and graph similarity (y)

et al., 2014] and employ a discriminator, trained to distinguish between the translated source and the target language space, and a generator learning a translation matrix [Barone, 2016]. Zhang et al. [2017], in addition, use different forms of regularization for convergence, while Conneau et al. [2018a] uses additional steps to refine the induced embedding space.

Unsupervised machine translation Research on unsupervised machine translation [Lample et al., 2018a, Artetxe et al., 2018b, Lample et al., 2018b] has generated a lot of interest recently with a promise to support the construction of MT systems for and between resource-poor languages. All unsupervised NMT methods critically rely on accurate unsupervised BDI and back-translation. Models are trained to reconstruct a corrupted version of the source sentence and to translate its translated version back to the source language. Since the crucial input to these systems are indeed cross-lingual word embedding spaces induced in an unsupervised fashion, in this paper we also implicitly investigate one core limitation of such unsupervised MT techniques.

5.1.6 Summary

We investigated when unsupervised BDI [Conneau et al., 2018a] is possible and found that differences in morphology, domains or word embedding algorithms may challenge this approach. Further, we found eigenvector similarity of sampled nearest neighbor subgraphs to be predictive of unsupervised BDI performance. Building on these findings, we will propose a new cross-lingual embedding model that casts existing models from a latent variable perspective in the next section. We will show that this model performs

well with the weak supervision introduced in this chapter as well as on low-resource languages.

5.2 A Discriminative Latent-Variable Model for Bilingual Lexicon Induction

* We introduce a novel discriminative latent-variable model for the task of bilingual lexicon induction. Our model combines the bipartite matching dictionary prior of Haghighi et al. [2008] with a state-of-the-art embedding-based approach. To train the model, we derive an efficient Viterbi EM algorithm. We provide empirical improvements on six language pairs under two metrics and show that the prior theoretically and empirically helps to mitigate the hubness problem. We also demonstrate how previous work may be viewed as a similarly fashioned latent-variable model, albeit with a different prior.†

5.2.1 Introduction

Is there a more fundamental bilingual linguistic resource than a dictionary? The task of bilingual lexicon induction seeks to create a dictionary in a data-driven manner directly from monolingual corpora in the respective languages and, perhaps, a small seed set of translations. From a practical point of view, bilingual dictionaries have found uses in a myriad of NLP tasks ranging from machine translation [Klementiev et al., 2012a] to cross-lingual named entity recognition [Mayhew et al., 2017]. In this work, we offer a probabilistic twist on the task, developing a novel discriminative latent-variable model that outperforms previous work.

Our proposed model is a bridge between current state-of-the-art methods in bilingual lexicon induction that take advantage of word embeddings, e.g., the embeddings induced by Mikolov et al. [2013b]'s skip-gram objective, and older ideas in the literature that build explicit probabilistic models for the task. We propose a discriminative probability model, inspired by Irvine and Callison-Burch [2013], infused with the bipartite matching dictionary prior of Haghighi et al. [2008]. However, like more recent approaches [Artetxe et al., 2017], our model operates directly over pretrained word embeddings, induces a joint cross-lingual embedding space, and scales to large vocabulary sizes. To train our

*This section is adapted from: **Ruder, S.**, Cotterell, R., Kementchedjhieva, Y., and Søgaard, A. (2018). A Discriminative Latent-Variable Model for Bilingual Lexicon Induction. In *Proceedings of EMNLP 2018*. Sebastian focused on the implementation and experiments. Ryan focused on the presentation of the model and the proofs.

†The code used to run the experiments is available at `https://github.com/sebastianruder/latent-variable-vecmap`.

model, we derive a generalized expectation-maximization algorithm [EM; Neal and Hinton, 1998] and employ an efficient matching algorithm.

Empirically, we experiment on three standard and three extremely low-resource language pairs. We evaluate intrinsically, comparing the quality of the induced bilingual dictionary, as well as analyzing the resulting bilingual word embeddings themselves. The latent-variable model yields gains over several previous approaches across language pairs. It also enables us to make implicit modeling assumptions explicit. To this end, we provide a reinterpretation of Artetxe et al. [2017] as a latent-variable model with an IBM Model 1–style [Brown et al., 1993b] dictionary prior, which allows a clean side-by-side analytical comparison. Viewed in this light, the difference between our approach and Artetxe et al. [2017], the strongest baseline, is whether one-to-one alignments or one-to-many alignments are admitted between the words of the languages' respective lexicons. Thus, we conclude that our hard constraint on one-to-one alignments is primarily responsible for the improvements over Artetxe et al. [2017].

5.2.1.1 Graph-theoretic formulation

To ease the later exposition, we will formulate the task graph-theoretically. Let ℓ_{src} denote the source language and ℓ_{trg} the target language. Suppose the source language ℓ_{src} has n_{src} word types in its lexicon V_{src} and ℓ_{trg} has n_{trg} word types in its lexicon V_{trg}. We will write $v_{src}(i)$ for the ith word type in ℓ_{src} and $v_{trg}(i)$ for the ith word type in ℓ_{trg}. We can view the elements of V_{src} and V_{trg} as sets of vertices in a graph. Now consider the bipartite set of vertices $V = V_{trg} \cup V_{src}$. In these terms, a bilingual lexicon is just a bipartite graph $G = (E, V)$ and, thus, the task of bilingual lexicon induction is a combinatorial problem: the search for a 'good' edge set $E \subseteq V_{trg} \times V_{src}$. We depict such a bipartite graph in Figure 5.5. In section 5.2.2, we will operationalize the notion of 'goodness' by assigning a weight w_{ij} to each possible edge between V_{trg} and V_{src}.

When the edge set E takes the form of a **matching**, we will denote it as $\mathbf{m}$.[1] In general, we will be interested in partial matchings, where many vertices have no incident edges. We will write $\mathcal{M}$ for the set of all partial matchings on the bipartite graph G. The set of vertices in V_{trg} (respectively V_{src}) with no incident edges will be termed $\mathbf{u}_{trg}$ (respectively $\mathbf{u}_{src}$). Note that for any matching $\mathbf{m}$, we have the identity $\mathbf{u}_{trg} = V_{trg} \setminus \{i : (i, j) \in \mathbf{m}\}$.

[1] A matching is an edge set where none of the edges share common vertices [West, 2000].

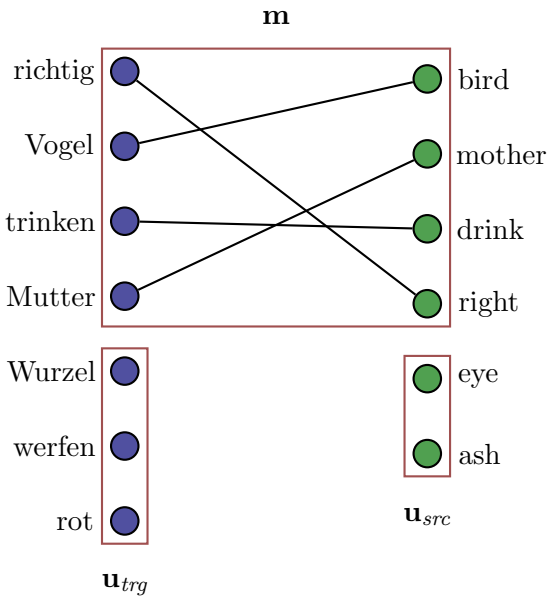


FIGURE 5.5: Partial lexicons of German and English shown as a bipartite graph. German is the target language and English is the source language. The $n_{trg} = 7$ German words are shown in blue and the $n_{src} = 6$ English words are shown in green. A bipartite matching $\mathbf{m}$ between the two sets of vertices is also depicted. The German nodes in $\mathbf{u}_{trg}$ are unmatched.

5.2.1.2 Word embeddings

Word embeddings will also play a key role in our model. For the remainder of the paper, we will assume we have access to d-dimensional embeddings for each language's lexicon—for example, those provided by a standard model such as skip-gram [Mikolov et al., 2013b]. Notationally, we define the real matrices $S \in \mathbb{R}^{d \times n_{src}}$ and $T \in \mathbb{R}^{d \times n_{trg}}$. Note that in this formulation $s_i \in \mathbb{R}^d$, the ith column of S, is the word embedding corresponding to $v_{src}(i)$. Likewise, note that $t_i \in \mathbb{R}^d$, the ith column of T, is the word embedding corresponding to $v_{trg}(i)$.

5.2.2 A latent-variable model

The primary contribution of this paper is a novel latent-variable model for bilingual lexicon induction. The latent variable will be the edge set E, as discussed in section 5.2.1.1. Given pretrained embeddings for the source and target languages, arranged into the matrices S and T, we define the density

$$p(T \mid S) \coloneqq \sum_{\mathbf{m} \in \mathcal{M}} p(T \mid S, \mathbf{m}) \cdot p(\mathbf{m}) \tag{5.5}$$

where $\mathcal{M}$ is the set of all bipartite matchings on the graph G and $\mathbf{m} \in \mathcal{M}$ is an individual matching. Note that, then, $p(\mathbf{m})$ is a *distribution* over all bipartite matchings on G such as the matching shown in Figure Figure 5.5. We will take $p(\mathbf{m})$ to be fixed as the uniform distribution for the remainder of the exposition, though more complicated distributions could be learned, of course. We further define the distribution

$$p_\theta(T \mid S, \mathbf{m}) \coloneqq \prod_{(i,j)\in\mathbf{m}} p(t_i \mid s_j) \cdot \prod_{i\in\mathbf{u}_{trg}} p(t_i) \tag{5.6}$$

Recall we write $(i, j) \in \mathbf{m}$ to denote an edge in the matching. Furthermore, for notational simplicity, we have dropped the dependence of $\mathbf{u}_{trg}$ on $\mathbf{m}$.[2]. Next, we define the two densities present in Equation 5.6 as Gaussians:

$$\begin{aligned} p_\theta(t \mid s) &\coloneqq \mathcal{N}(\Omega\, s, I) && (5.7)\\ &\propto \exp -1/2||t - \Omega\, s||_2^2 \\ p_\theta(t) &\coloneqq \mathcal{N}(\mu, I) && (5.8) \end{aligned}$$

Given a fixed matching $\mathbf{m}$, we may create matrices $S_\mathbf{m} \in \mathbb{R}^{d\times|\mathbf{m}|}$ and $T_\mathbf{m} \in \mathbb{R}^{d\times|\mathbf{m}|}$ such that the rows correspond to word vectors of matched vertices (translations under the matching $\mathbf{m}$). Now, after some algebra, we see that we can rewrite $\prod_{(i,j)\in\mathbf{m}} p(t_i \mid s_i)$ in matrix notation:

$$\begin{aligned} p_\theta(T_\mathbf{m} \mid S_\mathbf{m}, \mathbf{m}) &= \prod_{(i,j)\in\mathbf{m}} p(t_i \mid s_j) && (5.9)\\ &\propto \prod_{(i,j)\in\mathbf{m}} \exp -1/2||t_i - \Omega\, s_j||_2^2 \\ &= \exp -1/2 \sum_{(i,j)\in\mathbf{m}} ||t_i - \Omega\, s_j||_2^2 \\ &= \exp -1/2||T_\mathbf{m} - \Omega\, S_\mathbf{m}||_F^2 && (5.10) \end{aligned}$$

where $\Omega \in \mathbb{R}^{d\times d}$ is an orthogonal matrix of parameters to be learned. The result of this derivation, Equation 5.10, will become useful during the discussion of parameter estimation in Section 5.2.3.

We define the model's parameters, to be optimized, as $\theta = (\Omega, \mu)$.

Modeling Assumptions and their Limitations In the previous section, we have formulated the induction of a bilingual lexicon as the search for an edge set E, which we treat as a latent variable that we marginalize out in Equation 5.6. Specifically, we assume

[2]Recall $\mathbf{u}_{trg} = V_{trg} \setminus \{i : (i, j) \in \mathbf{m}\}$

that E is a partial matching. Thus, for every $(i, j) \in \mathbf{m}$, we have $t_i \sim \mathcal{N}(\Omega\, s_j, I)$, that is, the embedding for $v_{trg}(i)$ is assumed to have been drawn from a Gaussian centered around the embedding for $v_{src}(j)$, after an orthogonal transformation. This gives rise to two modeling assumptions, which we make explicit: (i) There exists a single source for every word in the target lexicon *and* that source cannot be used more than once.[3] (ii) There exists an orthogonal transformation, after which the embedding spaces are more or less equivalent.

Assumption (i) may be true for related languages, but is likely false for morphologically rich languages that have a many-to-many relationship between the words in their respective lexicons. We propose to ameliorate this using a rank constraint that only considers the top n most frequent words in both lexicons for matching (§7.1.4). In addition, we experiment with priors that express different matchings (§7.1.5).

As for assumption (ii), previous work [Xing et al., 2015, Artetxe et al., 2017] has achieved some success using an orthogonal transformation; recently, however, Søgaard et al. [2018] demonstrated that monolingual embedding spaces are not approximately isomorphic and that there is a complex relationship between word form and meaning, which is only inadequately modeled by current approaches, which for example cannot model polysemy. Nevertheless, we will show that imbuing our model with these assumptions helps empirically (§7.1.4), giving them practical utility.

Why it Works: The Hubness Problem Why should we expect the bipartite matching prior to help, given that we know of cases when multiple source words *should match a target word*? One answer is because the bipartite prior helps us obviate the **hubness problem**, a common issue in word-embedding-based bilingual lexicon induction [Dinu et al., 2015]. The hubness problem is an intrinsic problem of high-dimensional vector spaces where certain vectors will be *universal* nearest neighbors, i.e. they will be the nearest neighbor to a disproportionate number of other vectors [Radovanović et al., 2010]. Thus, if we allow one-to-many alignments, we will find the embeddings of certain elements of V_{src} acting as hubs, i.e. the model will pick them to generate a disproportionate number of target embeddings, which reduces the quality of the embedding space.[4]

Another explanation for the positive effect of the one-to-one alignment prior is its connection to the Wasserstein distance and computational optimal transport [Villani, 2008]. Concurrent work [Grave et al., 2018] similarly has found the one-to-one alignment prior to be beneficial.

[3] This is true by the definition of a matching.

[4] In section 5.2.4, we discuss the one-to-many alignment used in several of our baseline systems.

Algorithm 4 Viterbi EM for our latent-variable model

repeat
  // Viterbi E-Step
  $\mathbf{m}^\star \leftarrow \operatorname{argmax}_{\mathbf{m}\in\mathcal{M}} \log p_\theta(\mathbf{m} \mid S, T)$
  $\mathbf{u}^\star_{trg} \leftarrow V_{trg} \setminus \{i : (i,j) \in \mathbf{m}^\star\}$
  // M-Step
  $U\Sigma V^\top \leftarrow \texttt{SVD}\left(T_{\mathbf{m}^\star} S^\top_{\mathbf{m}^\star}\right)$
  $\Omega^\star \leftarrow UV^\top$
  $\mu^\star \leftarrow 1/|\mathbf{u}^\star_{trg}| \cdot \sum_{i\in\mathbf{u}^\star_{trg}} t_i$
  $\theta \leftarrow (\Omega^\star, \mu^\star)$
until converged

5.2.3 Parameter estimation

We will conduct parameter estimation through Viterbi EM. We describe first the E-step, then the M-step. Viterbi EM estimates the parameters by alternating between the two steps until convergence. We give the complete pseudocode in Algorithm 4.

5.2.3.1 Viterbi E-Step

The E-step asks us to compute the posterior of latent bipartite matchings $p(\mathbf{m} \mid S, T)$. Computation of this distribution, however, is intractable as it would require a *sum* over all bipartite matchings, which is #P-hard [Valiant, 1979]. Tricks from combinatorial optimization make it possible to *maximize* over all bipartite matchings in polynomial time. Thus, we fall back on the Viterbi approximation for the E-step [Brown et al., 1993b, Samdani et al., 2012]. The derivation will follow Haghighi et al. [2008]. In order to compute

$$\mathbf{m}^\star = \operatorname*{argmax}_{\mathbf{m}\in\mathcal{M}} \log p_\theta(\mathbf{m} \mid S, T) \tag{5.11}$$

we construct a fully connected bipartite graph $G = (E, V_{src} \cup V_{trg})$, where $E = V_{src} \times V_{trg}$. We weight each arc $(i,j) \in E$ with the weight between the projected source word and target word embeddings: $w_{ij} = \log p(t_i \mid s_j) - \log p(t_i) = -1/2(||t_i - \Omega\, s_j||_2^2 - ||t_i - \mu||_2^2)$, where the normalizers of both Gaussians cancel as both have the same covariance matrix, i.e., I. Note that in the case where the t_i and the s_j are of length 1, that is, $||t_i||_2 = ||s_j||_2 = 1$, and $\mu = \mathbf{0}$, we recover cosine similarity between the vectors up to an additive constant as orthogonal matrices preserve length (the constant is always $-1/2$ as $||t_i||_2 = 1$).[5] We may ignore this constant during the E-step's combinatorial optimization.

[5] Proof of the equivalence of the difference between the two Gaussians and cosine similarity up to an additive constant:

$$\begin{aligned}\log p(t_i \mid s_j) - \log p(t_i) &= -1/2\left(||t_i - \Omega\, s_j||_2^2 - ||t_i||_2^2\right)\\ &= -1/2\left(2(1 - \cos(t_i, \Omega\, s_j)) - 1\right)\\ &= \cos(t_i, \Omega\, s_j) - 1/2\end{aligned}$$

Note the optimal partial matching will contain no edges with weight $w_{ij} < 0$. For this reason, we remove such edges from the bipartite graph. To find the maximal *partial* bipartite matching on G to compute $\mathbf{m}^\star$, we employ an efficient algorithm as detailed in the next section.

Finding a Maximal Bipartite Matching We frame finding an optimal one-to-one alignment between n_{src} source and n_{trg} words as a combinatorial optimization problem, specifically, a linear assignment problem [LAP; Bertsimas and Tsitsiklis, 1997]. In its original formulation, the LAP requires assigning a number of agents (source words) to a number of tasks (target words) at a cost that varies based on each assignment. An optimal solution assigns each source word to exactly one target word and vice versa at minimum cost. The Hungarian algorithm [Kuhn, 1955] is one of the most well-known approaches for solving the LAP, but runs in $O((n_{src} + n_{trg})^3)$. This works for smaller vocabulary sizes,[6] but is prohibitive for matching cross-lingual word embeddings with large vocabularies for real-world applications.[7]

For each source word, most target words, however, are unlikely candidates for alignment. We thus propose to consider only the top k most similar target words for alignment with every source word. We sparsify the graph by weighting the edges for all other words with $-\infty$. The remaining weights w_{ij} are chosen as discussed above. We employ a version of the Jonker-Volgenant algorithm [Jonker and Volgenant, 1987, Volgenant, 1996], which has been optimized for LAP on sparse graphs, to find the maximum-weight matching $\mathbf{m}^\star$ on G.[8]

5.2.3.2 M-Step

Next, we will describe the M-step. Given an optimal matching $\mathbf{m}^\star$ computed in section 5.2.3.1, we search for a matrix $\Omega \in \mathbb{R}^{d \times d}$. We additionally enforce the constraint that Ω is a real orthogonal matrix, i.e., $\Omega^\top \Omega = I$. Previous work [Xing et al., 2015, Artetxe et al., 2017] found that the orthogonality constraint leads to noticeable improvements.

[6] Haghighi et al. [2008] use the Hungarian algorithm to find a matching between 2000 source and target language words.

[7] For reference, in Section 7.1.4, we learn bilingual lexicons between embeddings of 200,000 source and target language words.

[8] After acceptance to EMNLP 2018, Edouard Grave pointed out that Sinkhorn propagation [Adams and Zemel, 2011, Mena et al., 2018] may have been a computationally more effective manner to deal with the latent matchings.

Our M-step optimizes two objectives independently. First, making use of the result in equation (5.10), we optimize the following:

$$\log p(T_{\mathbf{m}^\star} \mid S_{\mathbf{m}^\star}, \mathbf{m}^\star) = ||T_{\mathbf{m}^\star} - \Omega\, S_{\mathbf{m}^\star}||_F^2 + C \tag{5.12}$$

with respect to Ω subject to $\Omega^\top \Omega = I$.[9] Second, we optimize the objective

$$\log \prod_{i \in \mathbf{u}_{trg}} p(t_i) = \sum_{i \in \mathbf{u}_{trg}} ||t_i - \mu||_2^2 + D \tag{5.13}$$

with respect to the mean parameter μ, which is simply an average. Note, again, we may ignore the constant D during optimization.

Optimizing equation (5.12) with respect to Ω is known as the orthogonal Procrustes problem [Schönemann, 1966, Gower and Dijksterhuis, 2004] and has a closed form solution that exploits the singular value decomposition [Horn and Johnson, 2012]. Namely, we compute $U\Sigma V^\top = T_{\mathbf{m}}^\top S_{\mathbf{m}}$. Then, we directly arrive at the optimum: $\Omega^\star = UV^\top$. Optimizing equation (5.13) can also been done in closed form; the point which minimizes distance to the data points (thereby maximizing the log-probability) is the centroid: $\mu^\star = {}^1\!/\!_{|\mathbf{u}_{trg}|} \cdot \sum_{i \in \mathbf{u}_{trg}} t_i$.

5.2.4 Reinterpretation of Artetxe et al. [2017] as a latent-variable model

The self-training method of Artetxe et al. [2017], our strongest baseline in section 7.1.4, may also be interpreted as a latent-variable model in the spirit of our exposition in section 5.2.2. Indeed, we only need to change the edge-set prior $p(\mathbf{m})$ to allow for edge sets other than those that are matchings. Specifically, a matching enforces a one-to-one alignment between types in the respective lexicons. Artetxe et al. [2017], on the other hand, allow for one-to-many alignments. We show how this corresponds to an alignment distribution that is equivalent to IBM Model 1 [Brown et al., 1993b], and that Artetxe et al. [2017]'s self-training method is actually a form of Viterbi EM.

To formalize Artetxe et al. [2017]'s contribution as a latent-variable model, we lay down some more notation. Let $\mathcal{A} = \{1, \ldots, n_{src} + 1\}^{n_{trg}}$, where we define $(n_{src} + 1)$ to be `none`, a distinguished symbol indicating unalignment. The set $\mathcal{A}$ is to be interpreted as the set of all one-to-many alignments $\mathbf{a}$ on the bipartite vertex set $V = V_{trg} \cup V_{src}$ such that $a_i = j$ means the ith vertex in V_{trg} is aligned to the jth vertex in V_{src}. Note that

[9] We may ignore the constant C during the optimization.

$a_i = (n_{src} + 1) = \texttt{none}$ means that the ith element of V_{trg} is unaligned. Now, by analogy to our formulation in section 5.2.2, we define

$$p(T \mid S) \coloneqq \sum_{\mathbf{a} \in \mathcal{A}} p(T \mid S, \mathbf{a}) \cdot p(\mathbf{a}) \tag{5.14}$$

$$= \sum_{\mathbf{a} \in \mathcal{A}} \prod_{i=1}^{n_{trg}} p(t_i \mid s_{a_i}, a_i) \cdot p(a_i) \tag{5.15}$$

$$= \prod_{i=1}^{n_{trg}} \sum_{a_i=1}^{n_{src}+1} p(t_i \mid s_{a_i}, a_i) \cdot p(a_i) \tag{5.16}$$

The move from equation (5.15) to equation (5.16) is the dynamic-programming trick introduced in Brown et al. [1993b]. This reduces the number of terms in the expression from exponentially many to polynomially many. We take $p(\mathbf{a})$ to be a uniform distribution over all alignments with no parameters to be learned.

Artetxe et al. [2017]'s Viterbi E-Step In the context of Viterbi EM, it means the max over $\mathcal{A}$ will decompose additively s

$$\max_{\mathbf{a} \in \mathcal{A}} \log p(\mathbf{a} \mid S, T) = \sum_{i=1}^{n_{trg}} \max_{1 \leq a_i \leq (n_{src}+1)} \log p(a_i \mid S, T)$$

thus, we can simply find $\mathbf{a}^\star$ component-wise as follows:

$$a_i^\star = \operatorname*{argmax}_{1 \leq a_i \leq (n_{src}+1)} \log p(a_i \mid t_i, s_{a_i}) \tag{5.17}$$

Artetxe et al. [2017]'s M-step The M-step remains unchanged from the exposition in section 5.2.2 with the exception that we fit Ω given matrices $S_{\mathbf{a}}$ and $T_{\mathbf{a}}$ formed from a one-to-many alignment $\mathbf{a}$, rather than a matching $\mathbf{m}$.

Why a Reinterpretation? The reinterpretation of Artetxe et al. [2017] as a probabilistic model yields a clear analytical comparison between our method and theirs. The only difference between the two is the constraint on the bilingual lexicon that the model is allowed to induce.

5.2.5 Experiments

We first conduct experiments on bilingual dictionary induction and cross-lingual word similarity on three standard language pairs, English–Italian, English–German, and English–Finnish.

5.2.5.1 Experimental details

Datasets For bilingual dictionary induction, we use the English–Italian dataset by Dinu et al. [2015] and the English–German and English–Finnish datasets by Artetxe et al. [2017]. For cross-lingual word similarity, we use the RG-65 and WordSim-353 cross-lingual datasets for English–German and the WordSim-353 cross-lingual dataset for English–Italian by Camacho-Collados et al. [2015].

Monolingual Embeddings We follow Artetxe et al. [2017] and train monolingual embeddings with word2vec, CBOW, and negative sampling [Mikolov et al., 2013a] on a 2.8 billion word corpus for English (ukWaC + Wikipedia + BNC), a 1.6 billion word corpus for Italian (itWaC), a 0.9 billion word corpus for German (SdeWaC), and a 2.8 billion word corpus for Finnish (Common Crawl).

Seed dictionaries Following Artetxe et al. [2017], we use dictionaries of 5,000 words, 25 words, and a numeral dictionary consisting of words matching the `[0-9]+` regular expression in both vocabularies.[10] In line with Søgaard et al. [2018], we additionally use a dictionary of identically spelled strings in both vocabularies.

Implementation details Similar to Artetxe et al. [2017], we stop training when the improvement on the average cosine similarity for the induced dictionary is below 1×10^{-6} between succeeding iterations. Unless stated otherwise, we induce a dictionary of 200,000 source and 200,000 target words as in previous work [Mikolov et al., 2013c, Artetxe et al., 2016]. For optimal 1:1 alignment, we have observed the best results by keeping the top $k = 3$ most similar target words. If using a rank constraint, we restrict the matching in the E-step to the top 40,000 words in both languages.[11] Finding an optimal alignment on the 200,000 $\times$ 200,000 graph takes about 25 minutes on CPU;[12] with a rank constraint, matching takes around three minutes.

Baselines We compare our approach with and without the rank constraint to the original bilingual mapping approach by Mikolov et al. [2013c]. In addition, we compare with Zhang et al. [2016b] and Xing et al. [2015] who augment the former with an orthogonality constraint and normalization and an orthogonality constraint respectively.

[10] The resulting dictionaries contain 2772, 2148, and 2345 entries for English–{Italian, German, Finnish} respectively.

[11] We validated both values with identical strings using the 5,000 word lexicon as validation set on English–Italian.

[12] Training takes a similar amount of time as [Artetxe et al., 2017] due to faster convergence.

	English–Italian				English–German				English–Finnish			
	5,000	25	num	iden	5,000	25	num	iden	5,000	25	num	iden
Mikolov et al. [2013c]	34.93	00.00	0.00	1.87	35.00	0.00	0.07	19.20	25.91	0.00	0.00	7.02
Xing et al. [2015]	36.87	0.00	0.13	27.13	41.27	0.07	0.53	38.13	28.23	0.07	0.56	17.95
Zhang et al. [2016b]	36.73	0.07	0.27	28.07	40.80	0.13	0.87	38.27	28.16	0.14	0.42	17.56
Artetxe et al. [2016]	39.27	0.07	0.40	31.07	41.87	0.13	0.73	41.53	**30.62**	0.21	0.77	22.61
Artetxe et al. [2017]	39.67	37.27	39.40	39.97	40.87	39.60	40.27	40.67	28.72	**28.16**	26.47	27.88
Ours (1:1)	41.00	39.63	40.47	41.07	**42.60**	**42.40**	**42.60**	**43.20**	29.78	0.07	3.02	**29.76**
Ours (1:1, rank constr.)	**42.47**	**41.13**	**41.40**	**41.80**	41.93	**42.40**	41.93	41.47	28.23	27.04	**27.60**	27.81

TABLE 5.6: Precision at 1 (P@1) scores for bilingual lexicon induction of different models with different seed dictionaries and languages on the full vocabulary.

Finally, we compare with Artetxe et al. [2016] who add dimension-wise mean centering to Xing et al. [2015], and Artetxe et al. [2017].

Both Mikolov et al. [2013c] and Artetxe et al. [2017] are special cases of our famework and comparisons to these approaches thus act as an ablation study. Specifically, Mikolov et al. [2013c] does not employ orthogonal Procrustes, but rather allows the learned matrix Ω to range freely. Likewise, as discussed in section 5.2.4, Artetxe et al. [2017] make use of a Viterbi EM style algorithm with a different prior over edge sets.[13]

5.2.5.2 Results

We show results for bilingual dictionary induction in Table 5.6 and for cross-lingual word similarity in Table 5.7. Our method with a 1:1 prior outperforms all baselines on English–German and English–Italian.[14] Interestingly, the 1:1 prior by itself fails on English–Finnish with a 25 word and numerals seed lexicon. We hypothesize that the prior imposes too strong of a constraint to find a good solution for a distant language pair from a poor initialization. With a better—but still weakly supervised—starting point using identical strings, our approach finds a good solution. Alternatively, we can mitigate this deficiency effectively using a rank constraint, which allows our model to converge to good solutions even with a 25 word or numerals seed lexicon. The rank constraint generally performs similarly or boosts performance, while being about 8 times faster. All approaches do better with identical strings compared to numerals, indicating that the former may be generally suitable as a default weakly-supervised seed lexicon.

On cross-lingual word similarity, our approach yields the best performance on WordSim-353 and RG-65 for English–German and is only outperformed by Artetxe et al. [2017] on English–Italian Wordsim-353.

[13]Other recent improvements such as symmetric reweighting [Artetxe et al., 2018a] are orthogonal to our method, which is why we do not explicitly compare to them here.

[14]Note that results are not directly comparable to [Conneau et al., 2018a] due to the use of embeddings trained on different monolingual corpora (WaCKy vs. Wikipedia).

		en-it	en-de	
	Dict	WS	RG	WS
Mikolov et al. [2013c]	5k	.627	.643	.528
Xing et al. [2015]	5k	.614	.700	.595
Zhang et al. [2016b]	5k	.616	.704	.596
Artetxe et al. [2016]	5k	.617	.716	.597
Artetxe et al. [2017]	5k	.624	.742	.616
	25	.626	.749	.612
	num	**.628**	.739	.604
Ours (1:1)	5k	.621	.733	**.618**
	25	.621	.740	.617
	num	.624	.743	.617
Ours (1:1, rank constr.)	5k	.623	.741	.609
	25	.622	.753	.609
	num	.625	**.755**	.611

TABLE 5.7: Spearman correlations on English–Italian and English–German cross-lingual word similarity datasets.

5.2.6 Analysis

Vocabulary sizes The beneficial contribution of the rank constraint demonstrates that in similar languages, many frequent words will have one-to-one matchings, while it may be harder to find direct matches for infrequent words. We would thus expect the latent variable model to perform better if we only learn dictionaries for the top n most frequent words in both languages. We show results for our approach in comparison to the baselines in Figure 5.6 for English–Italian using a 5,000 word seed lexicon across vocabularies consisting of different numbers n of the most frequent words[15].

The comparison approaches mostly perform similar, while our approach performs particularly well for the most frequent words in a language.

Different priors An advantage of having an explicit prior as part of the model is that we can experiment with priors that satisfy different assumptions. Besides the 1:1 prior, we experiment with a 2:2 prior and a 1:2 prior. For the 2:2 prior, we create copies of the source and target words V'_{src} and V'_{trg} and add these to our existing set of vertices $V' = (V_{trg} + V'_{trg}, V_{src} + V'_{src})$. We run the Viterbi E-step on this new graph G' and merge matched pairs of words and their copies in the end. Similarly, for the 1:2 prior, which allows one source word to be matched to two target words, we augment the vertices with

[15]We only use the words in the 5,000 word seed lexicon that are contained in the n most frequent words. We do not show results for the 25 word seed lexicon and numerals as they are not contained in the smallest n of most frequent words.

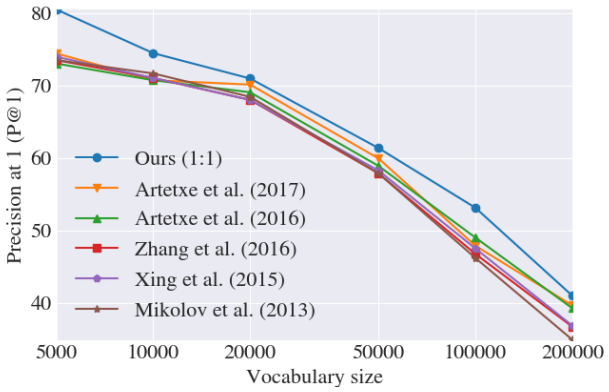
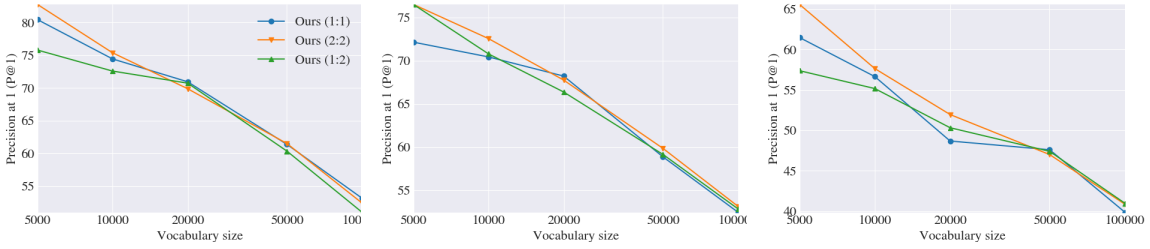


FIGURE 5.6: Bilingual dictionary induction results of our method and baselines for English–Italian with a 5,000 word seed lexicon across different vocabulary sizes.

a copy of the source words V'_{src} and proceed as above. We show results for bilingual dictionary induction with different priors across different vocabulary sizes in Figure 5.7.



(A) English–Italian (B) English–German (C) English–Finnish

FIGURE 5.7: Bilingual dictionary induction results of our method with different priors using a 5,000 word seed lexicon across different vocabulary sizes.

The 2:2 prior performs best for small vocabulary sizes. As solving the linear assignment problem for larger vocabularies becomes progressively more challenging, the differences between the priors become obscured and their performance converges.

Hubness problem We analyze empirically whether the prior helps with the hubness problem. Following Lazaridou et al. [2015], we define the *hubness* $N_k(y)$ at k of a target word y as follows:

$$N_k(y) = |\{x \in Q \mid y \in NN_k(x, G)\}| \tag{5.18}$$

where Q is a set of query source language words and $NN_k(x, G)$ denotes the k nearest neighbors of x in the graph G.[16] In accordance with Lazaridou et al. [2015], we set $k = 20$ and use the words in the evaluation dictionary as query terms. We show the target

[16]In other words, the hubness of a target word measures how often it occurs in the neighborhood of the query terms.

Artetxe et al. [2017]	Ours (1:1)
luis (20)	gleichgültigkeit - '*indifference*' (14)
ungarischen - '*Hungarian*' (18)	heuchelei - '*hypocrisy*' (13)
jorge (17)	ahmed (13)
mohammed (17)	ideologie - '*ideology*' (13)
gewiß - '*certainly*' (17)	eduardo (13)

TABLE 5.8: Hubs in English–German cross-lingual embedding space with degree of hubness. Non-name tokens are translated.

language words with the highest hubness using our method and Artetxe et al. [2017] for English–German with a 5,000 seed lexicon and the full vocabulary in Table 5.8.[17]

Hubs are fewer and occur less often with our method, demonstrating that the prior—to some extent—aids with resolving hubness. Interestingly, compared to Lazaridou et al. [2015], hubs seem to occur less often and are more meaningful in current cross-lingual word embedding models.[18] For instance, the neighbors of 'gleichgültigkeit' all relate to indifference and words appearing close to 'luis' or 'jorge' are Spanish names. This suggests that the prior might also be beneficial in other ways, e.g. by enforcing more reliable translation pairs for subsequent iterations.

Low-resource languages Cross-lingual embeddings are particularly promising for low-resource languages, where few labeled examples are typically available, but are not adequately reflected in current benchmarks (besides the English–Finnish language pair). We perform experiments with our method with and without a rank constraint and Artetxe et al. [2017] for three truly low-resource language pairs, English–{Turkish, Bengali, Hindi}. We additionally conduct an experiment for Estonian-Finnish, similarly to Søgaard et al. [2018]. For all languages, we use fastText embeddings [Bojanowski et al., 2017] trained on Wikipedia, the evaluation dictionaries provided by Conneau et al. [2018a], and a seed lexicon based on identical strings to reflect a realistic use case. We note that English does not share scripts with Bengali and Hindi, making this even more challenging. We show results in Table 5.9.

[17]We verified that hubs are mostly consistent across runs and similar across language pairs.

[18]Lazaridou et al. [2015] observed mostly rare words with N_{20} values of up to 50 and many with $N_{20} > 20$.

	en-tr	en-bn	en-hi	et-fi
Artetxe et al. [2017]	28.93	0.87	2.07	30.18
Ours (1:1)	38.73	2.33	10.47	33.79
Ours (1:1, rank constr.)	**42.40**	**11.93**	**31.80**	**34.78**

TABLE 5.9: Bilingual dictionary induction results for English-{Turkish, Bengali, Hindi} and Estonian-Finnish.

Surprisingly, the method by Artetxe et al. [2017] is unable to leverage the weak supervision and fails to converge to a good solution for English-Bengali and English-Hindi.[19] Our method without a rank constraint significantly outperforms Artetxe et al. [2017], while particularly for English-Hindi the rank constraint dramatically boosts performance.

Error analysis To illustrate the types of errors the model of Artetxe et al. [2017] and our method with a rank constraint make, we query both of them with words from the test dictionary of Artetxe et al. [2017] in German and seek their nearest neighbours in the English embedding space. P@1 over the German-English test set is 36.38 and 39.18 for Artetxe et al. [2017] and our method respectively. We show examples where nearest neighbours of the methods differ in Table 5.10.

Similar to Kementchedjhieva et al. [2018], we find that morphologically related words are often the source of mistakes. Other common sources of mistakes in this dataset are names that are translated to different names and nearly synonymous words being predicted. Both of these sources indicate that while the learned alignment is generally good, it is often not sufficiently precise.

5.2.7 Related work

Cross-lingual embedding priors Haghighi et al. [2008] first proposed an EM self-learning method for bilingual lexicon induction, representing words with orthographic and context features and using the Hungarian algorithm in the E-step to find an optimal 1:1 matching. Artetxe et al. [2017] proposed a similar self-learning method that uses word embeddings, with an implicit one-to-many alignment based on nearest neighbor queries. Vulić and Korhonen [2016] proposed a more strict one-to-many alignment based on symmetric translation pairs, which is also used by Conneau et al. [2018a]. Our method bridges the gap between early latent variable and word embedding-based approaches and explicitly allows us to reason over its prior.

[19] One possible explanation is that Artetxe et al. [2017] particularly rely on numerals, which are normalized in the fastText embeddings.

Query	Gold	Artetxe et al. [2017]	Ours
unregierbar	ungovernable	untenable	uninhabitable
nikolai	nikolaj	feodor	nikolai
memoranden	memorandums	communiqués	memos
argentinier	argentinians	brazilians	argentines
trostloser	bleaker	dreary	dark-coloured
umverteilungen	redistributions	inequities	reforms
modischen	modish	trend-setting	modish
tranquilizer	tranquillizers	clonidine	opiates
sammelsurium	hotchpotch	assortment	mishmash
demagogie	demagogy	opportunism	demagogy
andris	andris	rehn	viktor
dehnten	halmahera	overran	stretched
deregulieren	deregulate	deregulate	liberalise
eurokraten	eurocrats	bureaucrats	eurosceptics
holte	holte	threw	grabbed
reserviertheit	aloofness	disdain	antipathy
reaktiv	reactively	reacting	reactive
danuta	danuta	julie	monika
scharfblick	perspicacity	sagacity	astuteness

TABLE 5.10: Comparison of example translations for German-English by our model and Artetxe et al. [2017]

Hubness problem The hubness problem is an intrinsic problem in high-dimensional vector spaces [Radovanović et al., 2010]. Dinu et al. [2015] first observed it for cross-lingual embedding spaces and proposed to address it by re-ranking neighbor lists. Lazaridou et al. [2015] proposed a max-marging objective as a solution, while more recent approaches proposed to modify the nearest neighbor retrieval by inverting the softmax [Smith et al., 2017] or scaling the similarity values [Conneau et al., 2018a].

5.2.8 Summary

We have presented a novel latent-variable model for bilingual lexicon induction, building on the work of Artetxe et al. [2017]. Our model combines the prior over bipartite matchings inspired by Haghighi et al. [2008] and the discriminative, rather than generative, approach inspired by Irvine and Callison-Burch [2013]. We show empirical gains on six language pairs and theoretically and empirically demonstrate the application of the bipartite matching prior to solving the hubness problem.

5.3 Conclusions

In this chapter, we have analysed methods for learning cross-lingual word embeddings and proposed a new model inspired by classic approaches. We have emphasized the need to evaluate on low-resource languages, on which our latent variable model performed particularly well. Similar to sharing word embeddings between languages, the most common approach in multi-task learning is to share parameters between different tasks. In the next chapter, we will propose two novel models that enable more effective parameter sharing.

Chapter 6

Improved Sharing in Multi-task Learning

We have seen in the previous chapter that unsupervised methods fail if languages are too distant. Similarly, existing multi-task learning methods such as hard parameter sharing lead to negative transfer if tasks are dissimilar. In addition, they are restricted to sharing separate layers and cannot effectively exploit similarities between tasks. In this chapter, we propose two architectures for multi-task learning that enable more flexible sharing between tasks.

In Section 6.1, we propose sluice networks, a meta-architecture that automatically learns how tasks should share information. We achieve this by defining parameters that enable the model to control the flow of information between tasks. In addition, the model is able to exploit the intrinsic hierarchy of NLP tasks and learn which tasks require low-level syntactic information or high-level semantic information. Finally, we enable the model to learn shared and private representations by partitioning the weights into orthogonal subspaces. The model outperforms strong single-task learning and multi-task learning approaches on four tasks and multiple domains.

Section 6.2 presents a model that is able to leverage information from related label spaces more effectively. In particular, the model employs a label embedding layer to learn relationships between tasks. We also propose a label transfer network that allows us to leverage unlabelled data and auxiliary data from other tasks via semi-supervised learning methods similar to those used in Section 4.2. We evaluate the model on a range of pairwise sequence classification tasks. It outperforms the state-of-the-art for aspect- and topic-based sentiment analysis.

6.1 Latent Multi-task Architecture Learning

* Multi-task learning (MTL) allows deep neural networks to learn from related tasks by sharing parameters with other networks. In practice, however, MTL involves searching an enormous space of possible parameter sharing architectures to find (a) the layers or subspaces that benefit from sharing, (b) the appropriate amount of sharing, and (c) the appropriate relative weights of the different task losses. Recent work has addressed each of the above problems in isolation. In this work we present an approach that learns a *latent* multi-task architecture that jointly addresses (a)–(c). We present experiments on synthetic data and data from OntoNotes 5.0, including four different tasks and seven different domains. Our extension consistently outperforms previous approaches to learning latent architectures for multi-task problems and achieves up to 15% average error reductions over common approaches to MTL.

6.1.1 Introduction

Multi-task learning in deep neural networks is typically a result of parameter sharing between two networks (of usually the same dimensions) [Caruana, 1993]. If you have two three-layered, recurrent neural networks, both with an embedding inner layer and each recurrent layer feeding the task-specific classifier function through a feed-forward neural network, we have 19 pairs of layers that could share parameters. With the option of having private spaces, this gives us 5^{19} =19,073,486,328,125 possible MTL architectures. If we additionally consider soft sharing of parameters, the number of possible architectures grows infinite. It is obviously not feasible to search this space. Neural architecture search (NAS) [Zoph and Le, 2017] typically requires learning from a large pool of experiments with different architectures. Searching for multi-task architectures via reinforcement learning [Wong and Gesmundo, 2018] or evolutionary approaches [Liang et al., 2018] can therefore be quite expensive. In this paper, we *jointly* learn a *latent* multi-task architecture and task-specific models, paying a minimal computational cost over single task learning and standard multi-task learning (5-7% training time). We refer to this problem as *multi-task architecture learning*. In contrast to architecture *search*, the overall meta-architecture is fixed and the model learns the optimal latent connections and pathways for each task. Recently, a few authors have considered multi-task architecture learning [Misra et al., 2016, Meyerson and Miikkulainen, 2018], but these papers only address a subspace of the possible architectures typically considered in neural multi-task learning, while other approaches at most consider a couple of architectures for sharing [Søgaard and Goldberg, 2016, Peng and Dredze, 2016, Alonso and Plank, 2017]. In contrast,

*This section is adapted from: **Ruder, S.**, Bingel, J., Augenstein, I., and Søgaard, A. (2019). Latent Multi-task Architecture Learning. In *Proceedings of AAAI 2019*.

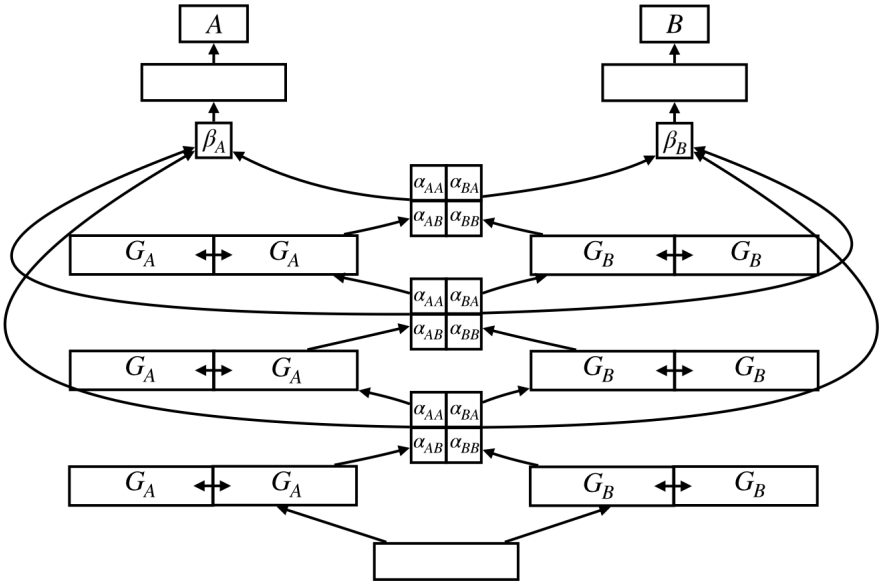


FIGURE 6.1: A sluice meta-network with one main task A and one auxiliary task B. It consists of a shared input layer (bottom), two task-specific output layers (top), and three hidden layers per task, each partitioned into two subspaces G that are enforced to be orthogonal. α parameters control which subspaces are shared between main and auxiliary task, while β parameters control which layer outputs are used for prediction. For simplicity, we do not index layers and subspaces. With two subspaces, each block $\alpha_{AA}, \alpha_{BA}, \ldots \in \mathbb{R}^{2\times 2}$. With three layers, $\beta_A, \beta_B \in \mathbb{R}^3$.

we introduce a framework that unifies previous approaches by introducing trainable parameters for all the components that differentiate multi-task learning approaches along the above dimensions.

Contributions We present a novel meta-architecture (shown in Figure 6.1) that generalizes several previous multi-task architectures, with an application to sequence tagging problems. Our meta-architecture enables multi-task architecture learning, i.e., learning (a) what layers to share between deep recurrent neural networks, but also (b) which parts of those layers to share, and with what strength, as well as (c) a mixture model of skip connections at the architecture's outer layer. We show that the architecture is a generalization of various multi-task [Caruana, 1998, Søgaard and Goldberg, 2016, Misra et al., 2016] and transfer learning algorithms [Daumé III, 2007]. We evaluate it on four tasks and across seven domains on OntoNotes 5.0 [Weischedel et al., 2013], where it consistently outperforms previous work on multi-task architecture learning, as well as common MTL approaches. Moreover, we study the task properties that predict gains and those that correlate with learning certain types of sharing.

6.1.2 Multi-task architecture learning

We introduce a meta-architecture for multi-task architecture learning, which we refer to as a sluice network, sketched in Figure 6.1 for the case of two tasks.

The network learns to share parameters between M neural networks—in our case, two deep recurrent neural networks (RNNs) [Hochreiter and Schmidhuber, 1997]. The network can be seen as an end-to-end differentiable union of a set of sharing architectures with parameters controlling the sharing. By learning the weights of those sharing parameters (*sluices*) jointly with the rest of the model, we arrive at a task-specific MTL architecture over the course of training.

The two networks A and B share an embedding layer associating the elements of an input sequence, in our case English words, with vector representations via word and character embeddings. The two sequences of vectors are then passed on to their respective inner recurrent layers. Each layer is divided into subspaces (by splitting the matrices in half), e.g., for network A into $G_{A,1}$ and $G_{A,2}$, which allow the sluice network to learn task-specific and shared representations, if beneficial. The subspaces have different weights.

The output of the inner layer of network A is then passed to its second layer, as well as to the second layer of network B. This traffic of information is mediated by a set of α and β parameters similar to the way a sluice controls the flow of water. Specifically, the second layer of each network receives a combination of the output of the two inner layers weighted by the α parameters. Importantly, these α parameters are trainable and allow the model to learn whether to share or to focus on task-specific features in a subspace. Finally, a weighted combination of the outputs of the outer recurrent layers $G_{\cdot,3,\cdot}$ as well as the weighted outputs of the inner layers are mediated through β parameters, which reflect a mixture over the representations at various depths of the multi-task architecture. In sum, sluice networks have the capacity to learn what layers and subspaces should be shared, and how much, as well as at what layers the meta-network has learned the best representations of the input sequences.

Matrix Regularization We cast learning what to share as a *matrix regularization* problem, following [Jacob et al., 2009, Yang and Hospedales, 2017]. Assume M different tasks that are loosely related, with M potentially non-overlapping datasets $\mathcal{D}_1, \ldots, \mathcal{D}_M$. Each task is associated with a deep neural network with K layers $L_1, \ldots L_K$. For simplicity, we assume that all the deep networks have the same hyper-parameters at the outset. Let $W \in \mathbb{R}^{M \times D}$ be a matrix in which each row i corresponds to a model θ_i with D parameters. The loss that sluice networks minimize, with a penalty term Ω, is then as

follows: $\lambda_1 \mathcal{L}_1(\mathbf{f}(x;\theta_1), y_1) + \ldots + \lambda_M \mathcal{L}_M(\mathbf{f}(x;\theta_M), y_M) + \Omega$. The loss functions $\mathcal{L}_i$ are cross-entropy functions of the form $-\sum_y p(y) \log q(y)$ where y_i are the labels of task i. Note that sluice networks are not restricted to tasks with the same loss functions, but could also be applied to jointly learn regression and classification tasks. The weights λ_i determine the importance of the different tasks during training. We *explicitly* add inductive bias to the model via the regularizer Ω below, but our model also *implicitly* learns regularization through multi-task learning [Caruana, 1993] mediated by the α parameters, while the β parameters are used to learn the mixture functions $\mathbf{f}(\cdot)$, as detailed in the following.

Learning Matrix Regularizers We now explain how updating α parameters can lead to different matrix regularizers. Each matrix W consists of M rows where M is the number of tasks. Each row is of length D with D the number of model parameters. Subvectors $L_{m,k}$ correspond to the parameters of network m at layer k. Each layer consists of two subspaces with parameters $G_{m,k,1}$ and $G_{m,k,2}$. Our meta-architecture is partly motivated by the observation that for loosely related tasks, it is often beneficial if only certain features in specific layers are shared, while many of the layers and subspaces remain more task-specific [Søgaard and Goldberg, 2016]. We want to learn what to share while inducing models for the different tasks. For simplicity, we ignore subspaces at first and assume only two tasks A and B. The outputs $h_{A,k,t}$ and $h_{B,k,t}$ of the k-th layer for time step t for task A and B respectively interact through the α parameters (see Figure 6.1). Omitting t for simplicity, the output of the α layers is:

$$\begin{bmatrix} \widetilde{h}_{A,k} \\ \widetilde{h}_{B,k} \end{bmatrix} = \begin{bmatrix} \alpha_{AA} & \alpha_{AB} \\ \alpha_{BA} & \alpha_{BB} \end{bmatrix} \begin{bmatrix} {h_{A,k}}^\top, & {h_{B,k}}^\top \end{bmatrix} \tag{6.1}$$

where $\widetilde{h}_{A,k}$ is a linear combination of the outputs that is fed to the $k+1$-th layer of task A, and $\left[a^\top, b^\top\right]$ designates the stacking of two vectors $a, b \in \mathbb{R}^D$ to a matrix $M \in \mathbb{R}^{2 \times D}$. Subspaces [Virtanen et al., 2011, Bousmalis et al., 2016] should allow the model to focus on task-specific and shared features in different parts of its parameter space. Extending the α-layers to include subspaces, for 2 tasks and 2 subspaces, we obtain an α matrix $\in \mathbb{R}^{4 \times 4}$ that not only controls the interaction between the layers of both tasks, but also between their subspaces:

$$\begin{bmatrix} \widetilde{h}_{A_1,k} \\ \vdots \\ \widetilde{h}_{B_2,k} \end{bmatrix} = \begin{bmatrix} \alpha_{A_1A_1} & \dots & \alpha_{B_2A_1} \\ \vdots & \ddots & \vdots \\ \alpha_{A_1B_2} & \dots & \alpha_{B_2B_2} \end{bmatrix} \begin{bmatrix} {h_{A_1,k}}^\top, & \dots, & {h_{B_2,k}}^\top \end{bmatrix} \tag{6.2}$$

where $h_{A_1,k}$ is the output of the first subspace of the k-th layer of task A and $\widetilde{h}_{A_1,k}$ is the linear combination for the first subspace of task A. The input to the $k+1$-th layer of task A is then the concatenation of both subspace outputs: $h_{A,k} = \left[\widetilde{h}_{A_1,k}\,,\ \ \widetilde{h}_{A_2,k}\right]$. Different α weights correspond to different matrix regularizers Ω, including several ones that have been proposed previously for multi-task learning. We review those in Section 3. For now just observe that if all α-values are set to 0.25 (or any other constant), we obtain hard parameter sharing [Caruana, 1993], which is equivalent to a heavy L_0 matrix regularizer.

Adding Inductive Bias Naturally, we can also add explicit inductive bias to sluice networks by partially constraining the regularizer or adding to the learned penalty. Inspired by work on shared-space component analysis [Salzmann et al., 2010], we add a penalty to enforce a division of labor and discourage redundancy between shared and task-specific subspaces. While the networks can theoretically learn such a separation, an explicit constraint empirically leads to better results and enables the sluice networks to take better advantage of subspace-specific α-values. We introduce an orthogonality constraint [Bousmalis et al., 2016] between the layer-wise subspaces of each model: $\Omega = \sum_{m=1}^{M} \sum_{k=1}^{K} \|{G_{m,k,1}}^\top G_{m,k,2}\|_F^2$, where M is the number of tasks, K is the number of layers, $\|\cdot\|_F^2$ is the squared Frobenius norm, and $G_{m,k,1}$ and $G_{m,k,2}$ are the first and second subspace respectively in the k-th layer of the m-th task model.

Learning Mixtures Many tasks form an implicit hierarchy of low-level to more complex tasks, with intuitive synergies between the low-level tasks and *parts of* the complex tasks. Rather than hard-coding this structure [Søgaard and Goldberg, 2016, Hashimoto et al., 2017], we enable our model to learn hierarchical relations by associating different tasks with different layers if this is beneficial for learning. Inspired by advances in residual learning [He et al., 2016], we employ skip-connections from each layer, controlled using β parameters. This layer acts as a mixture model, returning a mixture of expert predictions:

$$\widetilde{h}_A^\top = \begin{bmatrix} \beta_{A,1} \\ \cdots \\ \beta_{A,k} \end{bmatrix}^\top \begin{bmatrix} {h_{A,1}}^\top, & \ldots & {h_{A,k}}^\top \end{bmatrix} \tag{6.3}$$

where $h_{A,k}$ is the output of layer k of model A, while $\widetilde{h}_{A,t}$ is the linear combination of all layer outputs of model A that is fed into the final softmax layer.

Complexity Our model only adds a minimal number of additional parameters compared to single-task models of the same architecture. In our experiments, we add α parameters between all task networks. As such, they scale linearly with the number of layers and quadratically with the number of tasks and subspaces, while β parameters scale linearly with the number of tasks and the number of layers. For a sluice network with M tasks, K layers per task, and 2 subspaces per layer, we thus obtain $4KM^2$ additional α parameters and KM β parameters. Training sluice networks is not much slower than training hard parameter sharing networks, with only a 5–7% increase in training time.

6.1.3 Prior work as instances of sluice networks

Our meta-architecture is very flexible and can be seen as a generalization over several existing algorithms for transfer and multi-task learning, including [Caruana, 1998, Daumé III, 2007, Søgaard and Goldberg, 2016, Misra et al., 2016]. We show how to derive each of these below.

- **Hard parameter sharing** between the two networks appears if all α values are set to the same constant [Caruana, 1998, Collobert and Weston, 2008]. This is equivalent to a mean-constrained ℓ_0-regularizer $\Omega(\cdot) = |\cdot|_0^{\bar{w}_i}$ and $\sum_i \lambda_i \mathcal{L}_i < 1$. Since the penalty for not sharing a parameter is 1, it holds that if the sum of weighted losses is smaller than 1, the loss with penalty is always the highest when all parameters are shared.
- The ℓ_1/ℓ_2 **group lasso** regularizer is $\sum_{g=1}^{G} ||G_{1,i,g}||_2$, a weighted sum over the ℓ_2 norms of the groups, often used to enforce subspace sharing [Zhou et al., 2010, Świrszcz and Lozano, 2012]. Our architecture learns a ℓ_1/ℓ_2 group lasso over the two subspaces (with the same degrees of freedom), when all α_{AB} and α_{BA}-values are set to 0. When the outer layer α-values are not shared, we get block communication between the networks.
- The approach to domain adaptation in [Daumé III, 2007], commonly referred to as **frustratingly easy domain adaptation**, which relies on a shared and a private

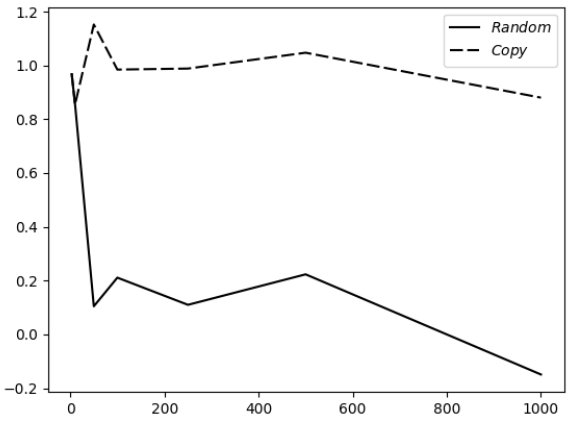


FIGURE 6.2: The relative importance of the auxiliary task ($\frac{\alpha_{\text{BA}}}{\alpha_{\text{AA}}}$) over number of training instances. With more data, the network learns *not* to share, when auxiliary task is randomly relabeled (*Random*).

space for each task or domain, can be encoded in sluice networks by setting all α_{AB}- and α_{BA}-weights associated with $G_{i,k,1}$ to 0, while setting all α_{AB}-weights associated with $G_{i,k,2}$ to α_{BB}, and α_{BA}-weights associated with $G_{i,k,2}$ to α_{AA}. Note that Daumé III [2007] Daumé III [2007] discusses three subspaces. We obtain this space if we only share one half of the second subspaces across the two networks.

- Søgaard and Goldberg [2016] Søgaard and Goldberg [2016] propose a **low supervision** model where only the inner layers of two deep recurrent works are shared. This is obtained using heavy mean-constrained L_0 regularization over the first layer $L_{i,1}$, e.g., $\Omega(W) = \sum_i^K ||L_{i,1}||_0$ with $\sum_i \lambda_i \mathcal{L}(i) < 1$, while for the auxiliary task, only the first layer β parameter is set to 1.
- Misra et al. [2016] Misra et al. [2016] introduce **cross-stitch networks** that have α values control the flow between layers of two convolutional neural networks. Their model corresponds to setting the α-values associated with $G_{i,j,1}$ to be identical to those for $G_{i,j,2}$, and by letting all but the β-value associated with the outer layer be 0.

In our experiments, we include hard parameter sharing, low supervision, and cross-stitch networks as baselines. We do not report results for group lasso and frustratingly easy domain adaptation, which were consistently inferior, by some margin, on development data.[1]

[1]Note that frustratingly easy domain adaptation was not designed for MTL.

6.1.4 Experiments

A synthetic experiment Our first experiment serves as a sanity check that our meta-architecture learns reasonable sharing architectures by learning α weights. We also want the α to adjust quickly in order not to slow down learning. We contrast two partially synthetic pairs of target and auxiliary data. In both cases, our target dataset is n instances (sentences) from our part-of-speech tagging dataset (see details below). In the first scenario (*Random*), the auxiliary dataset is a random relabeling of the same n instances. In the second scenario (*Copy*), the auxiliary dataset is a copy of the n instances.

For *Random*, we would like our α parameters to quickly learn that the auxiliary task at best contributes with noise injection. Initializing our α parameters to equal weights (0.25), we therefore hope to see a quick drop in the relative importance of the auxiliary task, given by $\frac{\alpha_{\text{BA}}}{\alpha_{\text{AA}}}$. Seeing n training instances, we expect this number to quickly drop, then stabilize to a slow decrease due to the reduced need for regularization with larger sample sizes.[2]

For *Copy*, in contrast, we expect no significant change in the relative importance of the auxiliary task over n training instances. We use the same hyperparameters as in our subsequent experiments (see below). The parameter settings are thus realistic, and not toy settings.

See Figure 6.2 for the results of our experiment. The two curves show the expected contrast between an auxiliary task with an all-noise signal (*Random*) and an auxiliary task with a perfect, albeit redundant, signal (*Copy*). This experiment shows that our meta-architecture quickly learns a good sharing architecture in clear cases such as *Random* and *Copy*. We now proceed to test whether multi-task architecture learning also leads to empirical gains over existing approaches to multi-task learning.

Data As testbed for our experiments, we choose the OntoNotes 5.0 dataset [Weischedel et al., 2013], not only due to its high inter-annotator agreement [Hovy et al., 2006], but also because it enables us to analyze the generalization ability of our models across different tasks and domains. The OntoNotes dataset provides data annotated for an array of tasks across different languages and domains. We present experiments with the English portions of datasets, for which we show statistics in Table 6.1.

[2]The quick drop is the meta-architecture learning that the auxiliary data is much less useful than the target data; the slight decrease after the first drop is the reduced need for regularization due to lower variance with more data.

	Domains						
	bc	**bn**	**mz**	**nw**	**pc**	**tc**	**wb**
Train	173289	206902	164217	878223	297049	90403	388851
Dev	29957	25271	15421	147955	25206	11200	49393
Test	35947	26424	17874	60756	25883	10916	52225

TABLE 6.1: Number of tokens for each domain in the OntoNotes 5.0 dataset.

WORDS	Abramov	had	a	car	accident
CHUNK	O	B-VP	B-NP	I-NP	I-NP
NER	B-PERSON	O	O	O	O
SRL	B-ARG0	B-V	B-ARG1	I-ARG1	I-ARG1
POS	NNP	VBD	DT	NN	NN

TABLE 6.2: Example annotations for CHUNK, NER, SRL, and POS.

Tasks In MTL, one task is usually considered the main task, while other tasks are used as auxiliary tasks to improve performance on the main task. As main tasks, we use chunking (CHUNK), named entity recognition (NER), and a simplified version of semantic role labeling (SRL) where we only identify headwords[3] , and pair them with part-of-speech tagging (POS) as an auxiliary task, following [Søgaard and Goldberg, 2016]. Example annotations for each task can be found in Table 6.2.

Model We use a state-of-the-art BiLSTM-based sequence labeling model [Plank et al., 2016] as the building block of our model. The BiLSTM consists of 3 layers with 100 dimensions that uses 64-dimensional word and 100-dimensional character embeddings, which are both randomly initialized. The output layer is an MLP with a dimensionality of 100. We initialize α parameters with a bias towards one source subspace for each direction and initialize β parameters with a bias towards the last layer[4]. We have found it most effective to apply the orthogonality constraint only to the weights associated with the LSTM inputs.

Training and Evaluation We train our models with stochastic gradient descent (SGD), an initial learning rate of 0.1, and learning rate decay[5]. During training, we uniformly sample from the data for each task. We perform early stopping with patience of 2 based on the main task and hyperparameter optimization on the in-domain development data of the newswire domain. We use the same hyperparameters for all comparison models across all domains. We train our models on each domain and evaluate them both

[3]We do this to keep pre-processing for SRL minimal.

[4]We experimented with different initializations for α and β parameters and found these to work best.

[5]We use SGD as Søgaard and Goldberg [2016] Søgaard and Goldberg [2016] also employed SGD. Adam yielded similar performance differences.

on the in-domain test set (Table 6.3, top) as well as on the test sets of all other domains (Table 6.3, bottom) to evaluate their out-of-domain generalization ability. Note that due to this set-up, our results are not directly comparable to the results reported in [Søgaard and Goldberg, 2016], who only train on the WSJ domain and use OntoNotes 4.0.

Baseline Models As baselines, we compare against i) a single-task model only trained on chunking; ii) the low supervision model [Søgaard and Goldberg, 2016], which predicts the auxiliary task at the first layer; iii) an MTL model based on hard parameter sharing [Caruana, 1993]; and iv) cross-stitch networks [Misra et al., 2016]. We compare these against our complete sluice network with subspace constraints and learned α and β parameters. We implement all models in DyNet [Neubig et al., 2017] and make our code available at `https://github.com/sebastianruder/sluice-networks`.

We first assess how well sluice networks perform on in-domain and out-of-domain data compared to the state-of-the-art. We evaluate all models on chunking with POS tagging as auxiliary task.

Chunking results We show results on in-domain and out-of-domain tests sets in Table 6.3. Out-of-domain results for each target domain are averages across the 6 remaining source domains. On average, sluice networks significantly outperform all other model architectures on both in-domain and out-of-domain data and perform best for all domains, except for the telephone conversation (tc) domain, where they are outperformed by cross-stitch networks. The average error reduction vs. single task learning is 12.8% on in-domain and 8.9% on out-of-domain data respectively. Compared to hard parameter sharing, the error reduction is 14.8% on in-domain data.

The performance boost is particularly significant for the out-of-domain setting, where sluice networks add more than 1 point in accuracy compared to hard parameter sharing and almost .5 compared to the strongest baseline on average, demonstrating that sluice networks are particularly useful to help a model generalize better. In contrast to previous studies on MTL [Alonso and Plank, 2017, Bingel and Søgaard, 2017, Augenstein et al., 2018], our model also consistently outperforms single-task learning. Overall, this demonstrates that our meta-architecture for learning which parts of multi-task models to share, with a small set of additional parameters to learn, can achieve significant and consistent improvements over strong baseline methods.

NER and SRL We now evaluate sluice nets on NER with POS tagging as auxiliary task and simplified semantic role labeling with POS tagging as auxiliary task. We show results in Table 6.4 (ID indicates in-domain and OOD means out-of-domain). Sluice

	In-domain results								
	System	bc	bn	mz	nw	pt	tc	wb	Avg
Baselines	Single task	90.80	92.20	91.97	92.76	97.13	89.84	92.95	92.52
	Hard sharing	90.31	91.73	92.33	92.22	96.40	90.59	92.84	92.35
	Low supervision	90.95	91.70	92.37	93.40	96.87	90.93	93.82	92.86
	Cross-stitch nets	91.40	92.49	92.59	93.52	96.99	**91.47**	94.00	93.21
Ours	Sluice network	**91.72**	**92.90**	**92.90**	**94.25**	**97.17**	90.99	**94.40**	**93.48**
	Out-of-domain results								
Baselines	Single task	85.95	87.73	86.81	84.29	90.91	84.55	73.36	84.80
	Hard sharing	86.31	87.73	86.96	84.99	90.76	84.48	73.56	84.97
	Low supervision	86.53	88.39	87.15	85.02	90.19	84.48	73.24	85.00
	Cross-stitch nets	87.13	88.40	87.67	85.37	91.65	**85.51**	73.97	85.67
Ours	Sluice network	**87.95**	**88.95**	**88.22**	**86.23**	**91.87**	85.32	**74.48**	**86.15**

TABLE 6.3: Accuracy scores for chunking on in-domain and out-of-domain test sets with POS as auxiliary task.

networks outperform the comparison models for both tasks on in-domain test data and successfully generalize to out-of-domain test data on average. They yield the best performance on 5 out of 7 domains and 4 out of 7 domains for NER and semantic role labeling.

	Named entity recognition								
	System	nw (ID)	bc	bn	mz	pt	tc	wb	OOD Avg
Baselines	Single task	95.04	93.42	93.81	93.25	94.29	94.27	92.52	93.59
	Hard sharing	94.16	91.36	93.18	93.37	**95.17**	93.23	**92.99**	93.22
	Low supervision	94.94	91.97	93.69	92.83	94.26	93.51	92.51	93.13
	Cross-stitch nets	95.09	92.39	93.79	93.05	94.14	93.60	92.59	93.26
Ours	Sluice network	**95.52**	**93.50**	**94.16**	**93.49**	93.61	**94.33**	92.48	**93.60**
	Simplified semantic role labeling								
Baselines	Single task	97.41	**95.67**	95.24	95.86	95.28	98.27	97.82	96.36
	Hard sharing	97.09	95.50	95.00	95.77	**95.57**	98.46	97.64	96.32
	Low supervision	97.26	95.57	95.09	95.89	95.50	98.68	97.79	96.42
	Cross-stitch nets	97.32	95.44	95.14	95.82	**95.57**	**98.69**	97.67	96.39
Ours	Sluice network	**97.67**	95.64	**95.30**	**96.12**	95.07	98.61	**98.01**	**96.49**

TABLE 6.4: Test accuracy scores for different target domains with nw as source domain for NER (main task) and SRL with POS as aux task.

Joint model Most work on MTL for NLP uses a single auxiliary task [Bingel and Søgaard, 2017, Alonso and Plank, 2017]. In this experiment, we use one sluice network to jointly learn our four tasks on the newswire domain and show results in Table 6.5.

System	CHUNK	NER	SRL	POS
Single task	**89.30**	94.18	96.64	88.62
Hard param.	88.30	94.12	**96.81**	89.07
Low super.	89.10	94.02	96.72	89.20
Sluice net	89.19	**94.32**	96.67	**89.46**

TABLE 6.5: All-tasks experiment: Test accuracy scores for each task with nw as source domain averaged across all target domains.

Here, the low-level POS tagging and simplified SRL tasks are the only ones that benefit from hard parameter sharing highlighting that hard parameter sharing by itself is not sufficient for doing effective multi-task learning with semantic tasks. We rather require task-specific layers that can be used to transform the shared, low-level representation into a form that is able to capture more fine-grained task-specific knowledge. Sluice networks outperform single task models for all tasks, except chunking and achieve the best performance on 2/4 tasks in this challenging setting.

6.1.5 Analysis

To better understand the properties and behavior of our meta-architecture, we conduct a series of analyses and ablations.

Task Properties and Performance Bingel and Søgaard [2017] correlate meta-characteristics of task pairs and gains compared to hard parameter sharing across a large set of NLP task pairs. Similarly, we correlate various meta-characteristics with error reductions and α, β values. Most importantly, we find that a) multi-task learning gains, also in sluice networks, are higher when there is less training data; and b) sluice networks learn to share more when there is more variance in the training data (cross-task αs are higher, intra-task αs lower). Generally, α values at the inner layers correlate more strongly with meta-characteristics than α values at the outer layers.

Ablation Analysis Different types of sharing may be more important than others. In order to analyze this, we perform an ablation analysis in Table 6.6. We investigate the impact of i) the α parameters; ii) the β parameters; and iii) the division into subspaces with an orthogonality penalty. We also evaluate whether concatenation of the outputs of each layer is a reasonable alternative to our mixture model. Overall, we find that learnable α parameters are preferable over constant α parameters. Learned β parameters marginally outperform skip-connections in the hard parameter sharing setting, while skip-connections are competitive with learned β values in the learned α setting.

Task sharing	Layer sharing	bc	bn	mz	nw	pt	tc	wb	Avg
constant α (hard)	Concatenation	86.70	88.24	87.20	85.19	90.64	85.33	73.75	85.29
	Skip-connections ($\beta = 1$)	86.65	88.10	86.82	84.91	90.92	84.89	73.62	85.13
	Mixture (learned β)	86.59	88.03	87.19	85.12	90.99	84.90	73.48	85.19
learned α (soft)	Concatenation	87.37	88.94	87.99	86.02	**91.96**	**85.83**	74.28	86.05
	Skip-connections	87.08	88.62	87.74	85.77	91.92	85.81	74.04	85.85
	Mixture	87.10	88.61	87.71	85.44	91.61	85.55	74.09	85.73
	Mixture + subspaces	**87.95**	**88.95**	**88.22**	**86.23**	91.87	85.32	**74.48**	**86.15**

TABLE 6.6: Ablation. Out-of-domain scores for Chunking with POS as auxiliary task on out-of-domain (OOD) test sets for Chunking (main task) with POS tagging as auxiliary task for different target domains for different configurations of sluice networks. OOD scores for each target domain are averaged across the 6 source domains.

In addition, modeling subspaces explicitly helps for almost all domains. To our knowledge, this is the first time that subspaces within individual LSTM layers have been shown to be beneficial.[6]. Being able to effectively partition LSTM weights opens the way to research in inducing more structured neural network representations that encode task-specific priors. Finally, concatenation of layer outputs is a viable form to share information across layers as has also been demonstrated by recent models such as DenseNet [Huang et al., 2017b].

Analysis of α values Figure 6.3 presents the final α weights in the inner and outer layers of sluice networks for Chunking, NER, and SRL, trained with newswire as training data. We see that a) for the low-level simplified SRL, there is more sharing at inner layers, which is in line with Søgaard and Goldberg [2016] Søgaard and Goldberg [2016], while Chunking and NER also rely on the outer layer, and b) more information is shared from the more complex target tasks than vice versa.

Analysis of β values Inspecting the β values for the all-tasks sluice net in Table 6.5, we find that all tasks place little emphasis on the first layer, but prefer to aggregate their representations in different later layers of the model: The more semantic NER and chunking tasks use the second and third layer to a similar extent, while for POS tagging and simplified SRL the representation of one of the two later layers dominates the prediction.

6.1.6 Related work

Hard parameter sharing [Caruana, 1993] is easy to implement, reduces overfitting, but is only guaranteed to work for (certain types of) closely related tasks [Baxter, 2000,

[6]Liu et al. [2017] induce subspaces between separate LSTM layers.

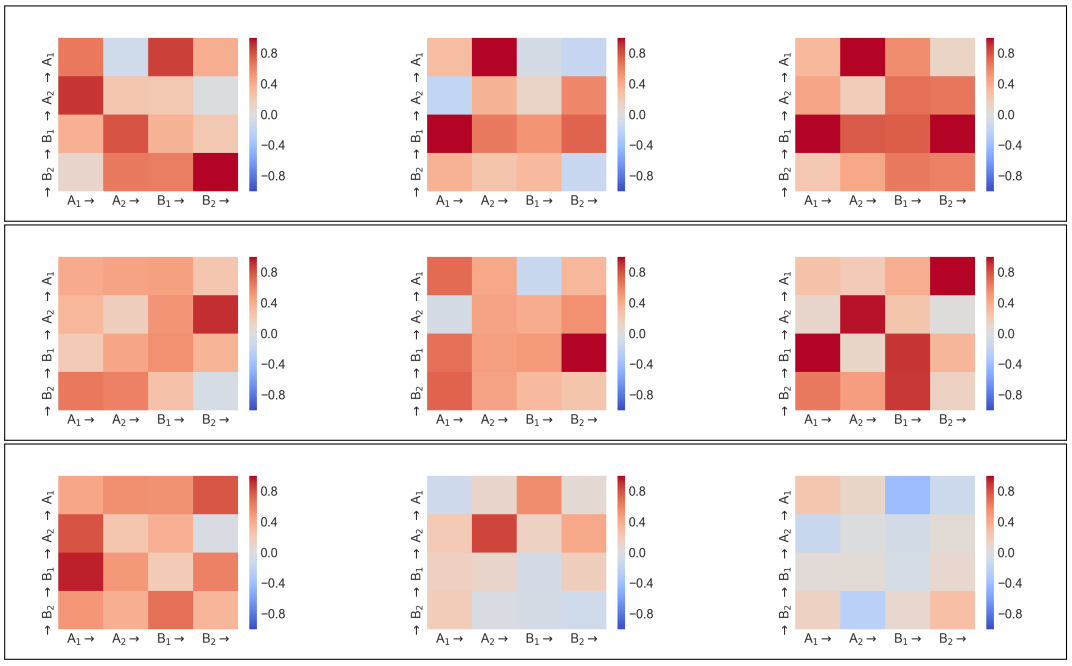


FIGURE 6.3: Heat maps of learned α parameters in sluice networks trained on newswire data for chunking, NER, and SRL (top to bottom). Layers are shown from bottom to top (left to right).

Maurer, 2007]. Peng and Dredze [2016] apply a variation of hard parameter sharing to multi-domain multi-task sequence tagging with a shared CRF layer and domain-specific projection layers. Yang et al. [2016] use hard parameter sharing to jointly learn different sequence-tagging tasks across languages. Alonso and Plank [2017] explore a similar set-up, but sharing is limited to the initial layer. In all three papers, the amount of sharing between the networks is fixed in advance.

In *soft parameter sharing* [Duong et al., 2015b], each task has separate parameters and separate hidden layers, as in our architecture, but the loss at the outer layer is regularized by the current distance between the models. Kumar and Daumé III [2012] and Maurer et al. [2013] enable *selective sharing* by allowing task predictors to select from sparse parameter bases for homogeneous tasks. Søgaard and Goldberg [2016] show that low-level tasks, i.e. syntactic tasks typically used for preprocessing such as POS tagging and NER, should be supervised at lower layers when used as auxiliary tasks.

Another line of work looks into separating the learned space into a private (i.e. task-specific) and shared space [Salzmann et al., 2010, Virtanen et al., 2011] to more explicitly capture the difference between task-specific and cross-task features. Constraints are enforced to prevent the models from duplicating information. Bousmalis et al. [2016] use shared and private encoders regularized with orthogonality and similarity constraints for domain adaptation for computer vision. Liu et al. [2017] use a similar technique for sentiment analysis. In contrast, we do not limit ourselves to a predefined way of sharing,

but let the model learn which parts of the network to share using latent variables, the weights of which are learned in an end-to-end fashion. Misra et al. [2016], focusing on applications in computer vision, consider a small subset of the sharing architectures that are learnable in sluice networks, i.e., *split architectures*, in which two n-layer networks share the innermost k layers with $0 \leq k \leq n$, but learn k with a mechanism very similar to α-values.

Our method is also related to the classic mixture-of-experts layer [Jacobs et al., 1991]. In contrast to this approach, our method is designed for multi-task learning and thus encourages a) the sharing of parameters between different task "experts" if this is beneficial as well as b) differentiating between low-level and high-level representations.

6.1.7 Summary

We introduced sluice networks, a meta-architecture for multi-task architecture search. In our experiments across four tasks and seven different domains, the meta-architecture consistently improved over strong single-task learning, architecture learning, and multi-task learning baselines. We also showed how our meta-architecture can learn previously proposed architectures for multi-task learning and domain adaptation.

While sluice networks seek to facilitate learning of tasks that require learning representations at different levels of a hierarchy, we will focus on the topmost level of the hierarchy, the labels, in the following section. We will propose an architecture that leverages information from related label spaces and apply this architecture to pairwise sequence classification tasks.

6.2 Multi-task Learning of Pairwise Sequence Classification Tasks Over Disparate Label Spaces

* We combine multi-task learning and semi-supervised learning by inducing a joint embedding space between disparate label spaces and learning transfer functions between label embeddings, enabling us to jointly leverage unlabelled data and auxiliary, annotated datasets. We evaluate our approach on a variety of sequence classification tasks with disparate label spaces. We outperform strong single and multi-task baselines and achieve a new state-of-the-art for aspect- and topic-based sentiment analysis.

This section is adapted from: Augenstein, I., **Ruder, S.***†, and Søgaard, A. (2018). Multi-task Learning of Pairwise Sequence Classification Tasks Over Disparate Label Spaces. In *Proceedings of NAACL-HLT 2018*. The model was implemented by Sebastian and Isabelle. Sebastian wrote most of the paper, while Isabelle ran the experiments.

6.2.1 Introduction

Multi-task learning (MTL) and semi-supervised learning are both successful paradigms for learning in scenarios with limited labelled data and have in recent years been applied to almost all areas of NLP. Applications of MTL in NLP, for example, include partial parsing [Søgaard and Goldberg, 2016], text normalisation [Bollman et al., 2017], neural machine translation [Luong et al., 2016], and keyphrase boundary classification [Augenstein and Søgaard, 2017].

Contemporary work in MTL for NLP typically focuses on learning representations that are useful across tasks, often through hard parameter sharing of hidden layers of neural networks [Collobert et al., 2011, Søgaard and Goldberg, 2016]. If tasks share optimal hypothesis classes at the level of these representations, MTL leads to improvements [Baxter, 2000]. However, while sharing hidden layers of neural networks is an effective regulariser [Søgaard and Goldberg, 2016], we potentially *loose synergies between the classification functions* trained to associate these representations with class labels. This paper sets out to build an architecture in which such synergies are exploited, with an application to pairwise sequence classification tasks. Doing so, we achieve a new state of the art on topic-based and aspect-based sentiment analysis tasks.

For many NLP tasks, disparate label sets are weakly correlated, e.g. part-of-speech tags correlate with dependencies [Hashimoto et al., 2017], sentiment correlates with emotion [Felbo et al., 2017, Eisner et al., 2016], etc. We thus propose to induce a joint label embedding space (visualised in Figure 6.5) using a Label Embedding Layer that allows us to model these relationships, which we show helps with learning.

In addition, for tasks where labels are closely related, we should be able to not only model their relationship, but also to directly estimate the corresponding label of the target task based on auxiliary predictions. To this end, we propose to train a Label Transfer Network (LTN) jointly with the model to produce pseudo-labels across tasks.

The LTN can be used to label unlabelled and auxiliary task data by utilising the 'dark knowledge' [Hinton et al., 2015] contained in auxiliary model predictions. This pseudo-labelled data is then incorporated into the model via semi-supervised learning, leading to a natural combination of multi-task learning and semi-supervised learning. We additionally augment the LTN with data-specific diversity features [Ruder and Plank, 2017] that aid in learning.

Contributions Our contributions are: a) We model the relationships between labels by inducing a joint label space for multi-task learning. b) We propose a Label Transfer

Network that learns to transfer labels between tasks and propose to use semi-supervised learning to leverage them for training. c) We evaluate MTL approaches on a variety of classification tasks and shed new light on settings where multi-task learning works. d) We perform an extensive ablation study of our model. e) We report state-of-the-art performance on topic-based and aspect-based sentiment analysis tasks.

6.2.2 Related work

Learning task similarities Existing approaches for learning similarities between tasks enforce a clustering of tasks [Evgeniou et al., 2005, Jacob et al., 2009], induce a shared prior [Yu et al., 2005, Xue et al., 2007, Daumé III, 2009], or learn a grouping [Kang et al., 2011, Kumar and Daumé III, 2012]. These approaches focus on homogeneous tasks and employ linear or Bayesian models. They can thus not be directly applied to our setting with tasks using disparate label sets.

Multi-task learning with neural networks Recent work in multi-task learning goes beyond hard parameter sharing [Caruana, 1993] and considers different sharing structures, e.g. only sharing at lower layers [Søgaard and Goldberg, 2016] and induces private and shared subspaces [Liu et al., 2017, Ruder et al., 2019a]. These approaches, however, are not able to take into account relationships between labels that may aid in learning. Another related direction is to train on disparate annotations of the same task [Chen et al., 2016, Peng et al., 2017]. In contrast, the different nature of our tasks requires a modelling of their label spaces.

Semi-supervised learning There exists a wide range of semi-supervised learning algorithms, e.g., self-training, co-training, tri-training, EM, and combinations thereof, several of which have also been used in NLP. Our approach is probably most closely related to an algorithm called *co-forest* [Li and Zhou, 2007]. In co-forest, like here, each learner is improved with unlabeled instances labeled by the ensemble consisting of all the other learners. Note also that several researchers have proposed using auxiliary tasks that are unsupervised [Plank et al., 2016, Rei, 2017], which also leads to a form of semi-supervised models.

Label transformations The idea of manually mapping between label sets or learning such a mapping to facilitate transfer is not new. Zhang et al. [2012] use distributional information to map from a language-specific tagset to a tagset used for other languages, in order to facilitate cross-lingual transfer. More related to this work, Kim et al. [2015] use

canonical correlation analysis to transfer between tasks with disparate label spaces. There has also been work on label transformations in the context of multi-label classification problems [Yeh et al., 2017].

6.2.3 Multi-task learning with disparate label spaces

Problem definition In our multi-task learning scenario, we have access to labelled datasets for T tasks $\mathcal{T}_1, \ldots, \mathcal{T}_T$ at training time with a target task $\mathcal{T}_T$ that we particularly care about. The training dataset for task $\mathcal{T}_i$ consists of N_k examples $X_{\mathcal{T}_i} = \{x_1^{\mathcal{T}_i}, \ldots, x_{N_k}^{\mathcal{T}_i}\}$ and their labels $Y_{\mathcal{T}_i} = \{\mathbf{y}_1^{\mathcal{T}_i}, \ldots, \mathbf{y}_{N_k}^{\mathcal{T}_i}\}$. Our base model is a deep neural network that performs classic hard parameter sharing [Caruana, 1993]: It shares its parameters across tasks and has task-specific softmax output layers, which output a probability distribution $\mathbf{p}^{\mathcal{T}_i}$ for task $\mathcal{T}_i$ according to the following equation:

$$\mathbf{p}^{\mathcal{T}_i} = \text{softmax}(\mathbf{W}^{\mathcal{T}_i}\mathbf{h} + \mathbf{b}^{\mathcal{T}_i}) \tag{6.4}$$

where $\text{softmax}(\mathbf{x}) = e^{\mathbf{x}} / \sum_{i=1}^{\|\mathbf{x}\|} e^{\mathbf{x}_i}$, $\mathbf{W}^{\mathcal{T}_i} \in \mathbb{R}^{L_i \times h}$, $\mathbf{b}^{\mathcal{T}_i} \in \mathbb{R}^{L_i}$ is the weight matrix and bias term of the output layer of task $\mathcal{T}_i$ respectively, $\mathbf{h} \in \mathbb{R}^h$ is the jointly learned hidden representation, L_i is the number of labels for task $\mathcal{T}_i$, and h is the dimensionality of $\mathbf{h}$.

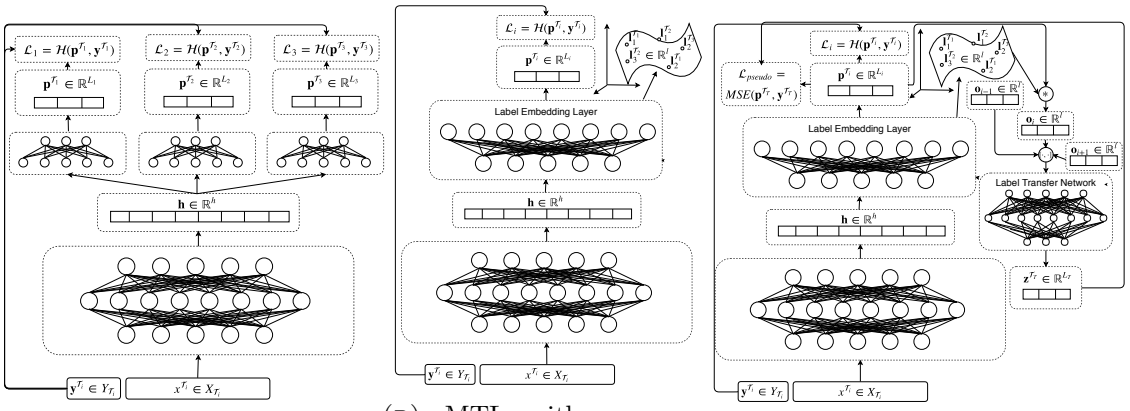
The MTL model is then trained to minimise the sum of the individual task losses:

$$\mathcal{L} = \lambda_1 \mathcal{L}_1 + \ldots + \lambda_T \mathcal{L}_T \tag{6.5}$$

where $\mathcal{L}_i$ is the negative log-likelihood objective $\mathcal{L}_i = \mathcal{H}(\mathbf{p}^{\mathcal{T}_i}, \mathbf{y}^{\mathcal{T}_i}) = -\frac{1}{N} \sum_n \sum_j \log \mathbf{p}_j^{\mathcal{T}_i} \mathbf{y}_j^{\mathcal{T}_i}$ and λ_i is a parameter that determines the weight of task $\mathcal{T}_i$. In practice, we apply the same weight to all tasks. We show the full set-up with hard parameter sharing and 3 tasks $\mathcal{T}_{1-3}$ and L_{1-3} labels per task in Figure 6.4a.

Label Embedding Layer In order to learn the relationships between labels, we propose a Label Embedding Layer (LEL) that embeds the labels of all tasks in a joint space. Instead of training separate softmax output layers as above, we introduce a label compatibility function $c(\cdot, \cdot)$ that measures how similar a label with embedding $\mathbf{l}$ is to the hidden representation $\mathbf{h}$:

$$c(\mathbf{l}, \mathbf{h}) = \mathbf{l} \cdot \mathbf{h} \tag{6.6}$$



(A) Multi-task learning (B) MTL with LEL (C) Semi-supervised MTL with LTN

FIGURE 6.4: a) Multi-task learning (MTL) with hard parameter sharing and 3 tasks $\mathcal{T}_{1-3}$ and L_{1-3} labels per task. A shared representation $\mathbf{h}$ is used as input to task-specific softmax layers, which optimise cross-entropy losses $\mathcal{L}_{1-3}$. b) MTL with the Label Embedding Layer (LEL) embeds task labels $\mathbf{l}_{1-L_i}^{\mathcal{T}_{1-3}}$ in a joint embedding space and uses these for prediction with a label compatibility function. c) Semi-supervised MTL with the Label Transfer Network (LTN) in addition optimises an unsupervised loss $\mathcal{L}_{pseudo}$ over pseudo-labels $\mathbf{z}^{\mathcal{T}_T}$ on auxiliary/unlabelled data. The pseudo-labels $\mathbf{z}^{\mathcal{T}_T}$ are produced by the LTN for the main task $\mathcal{T}_T$ using the concatenation of auxiliary task label output embeddings $[\mathbf{o}_{i-1}, \mathbf{o}_i, \mathbf{o}_{i+1}]$ as input.

where $\cdot$ is the dot product. This is similar to the Universal Schema Latent Feature Model introduced by [Riedel et al., 2013]. In contrast to other models that use the dot product in the objective function, we do not have to rely on negative sampling and a hinge loss [Collobert and Weston, 2008] as negative instances (labels) are known. For efficiency purposes, we use matrix multiplication instead of a single dot product and softmax instead of sigmoid activations:

$$\mathbf{p} = \text{softmax}(\mathbf{Lh}) \tag{6.7}$$

where $\mathbf{L} \in \mathbb{R}^{(\sum_i L_i) \times l}$ is the label embedding matrix for all tasks and l is the dimensionality of the label embeddings. In practice, we set l to the hidden dimensionality h. We use padding if $l < h$. We apply a task-specific mask to $\mathbf{L}$ in order to obtain a task-specific probability distribution $\mathbf{p}^{\mathcal{T}_i}$. The LEL is shared across all tasks, which allows us to learn the relationships between the labels in the joint embedding space. The LEL generalises the common setting of sharing hidden layers for tasks with the same label space to the setting of disparate label spaces. We show MTL with the LEL, which embeds task labels $\mathbf{l}_{1-L_i}^{\mathcal{T}_{1-3}}$ in a joint embedding space and uses these for prediction with a label compatibility function in Figure 6.4b.

Label Transfer Network The LEL allows us to learn the relationships between labels. In order to make use of these relationships, we would like to leverage the predictions of our auxiliary tasks to estimate a label for the target task. To this end, we introduce the Label Transfer Network (LTN). This network takes the auxiliary task outputs as input. In particular, we define the output label embedding $\mathbf{o}_i$ of task $\mathcal{T}_i$ as the sum of the task's label embeddings $\mathbf{l}_j$ weighted with their probability $\mathbf{p}_j^{\mathcal{T}_i}$:

$$\mathbf{o}_i = \sum_{j=1}^{L_i} \mathbf{p}_j^{\mathcal{T}_i} \mathbf{l}_j \tag{6.8}$$

The label embeddings $\mathbf{l}$ encode general relationship between labels, while the model's probability distribution $\mathbf{p}^{\mathcal{T}_i}$ over its predictions encodes fine-grained information useful for learning [Hinton et al., 2015]. The LTN is trained on labelled target task data. For each example, the corresponding label output embeddings of the auxiliary tasks are fed into a multi-layer perceptron (MLP), which is trained with a negative log-likelihood objective $\mathcal{L}_{\text{LTN}}$ to produce a pseudo-label $\mathbf{z}^{\mathcal{T}_T}$ for the target task $\mathcal{T}_T$:

$$\text{LTN}_T = \text{MLP}([\mathbf{o}_1, \ldots, \mathbf{o}_{T-1}]) \tag{6.9}$$

where $[\cdot, \cdot]$ designates concatenation. The mapping of the tasks in the LTN yields another signal that can be useful for optimisation and act as a regulariser. The LTN can also be seen as a mixture-of-experts layer [Jacobs et al., 1991] where the experts are the auxiliary task models. As the label embeddings are learned jointly with the main model, the LTN is more sensitive to the relationships between labels than a separately learned mixture-of-experts model that only relies on the experts' output distributions. As such, the LTN can be directly used to produce predictions on unseen data.

Semi-supervised MTL The downside of the LTN is that it requires additional parameters and relies on the predictions of the auxiliary models, which impacts the runtime during testing. Instead, of using the LTN for prediction directly, we can use it to provide pseudo-labels for unlabelled or auxiliary task data by utilising auxiliary predictions for semi-supervised learning.

We train the target task model on the pseudo-labelled data to minimise the squared error between the model predictions $\mathbf{p}^{\mathcal{T}_i}$ and the pseudo labels $\mathbf{z}^{\mathcal{T}_i}$ produced by the LTN:

$$\mathcal{L}_{pseudo} = MSE(\mathbf{p}^{\mathcal{T}_T}, \mathbf{z}^{\mathcal{T}_T}) = ||\mathbf{p}^{\mathcal{T}_T} - \mathbf{z}^{\mathcal{T}_T}||^2 \tag{6.10}$$

We add this loss term to the MTL loss in Equation 6.5. As the LTN is learned together with the MTL model, pseudo-labels produced early during training will likely not be helpful as they are based on unreliable auxiliary predictions. For this reason, we first train the base MTL model until convergence and then augment it with the LTN. We show the full semi-supervised learning procedure in Figure 6.4c. We additionally optimize an unsupervised loss $\mathcal{L}_{pseudo}$ over pseudo-labels $\mathbf{z}^{\mathcal{T}_T}$ on auxiliary/unlabelled data. The pseudo-labels $\mathbf{z}^{\mathcal{T}_T}$ are produced by the LTN for the main task $\mathcal{T}_T$ using the concatenation of auxiliary task label output embeddings $[\mathbf{o}_{i-1}, \mathbf{o}_i, \mathbf{o}_{i+1}]$ as input.

Data-specific features When there is a domain shift between the datasets of different tasks as is common for instance when learning NER models with different label sets, the output label embeddings might not contain sufficient information to bridge the domain gap.

To mitigate this discrepancy, we augment the LTN's input with features that have been found useful for transfer learning [Ruder and Plank, 2017]. In particular, we use the number of word types, type-token ratio, entropy, Simpson's index, and Rényi entropy as diversity features. We calculate each feature for each example.[1] The features are then concatenated with the input of the LTN.

Other multi-task improvements Hard parameter sharing can be overly restrictive and provide a regularisation that is too heavy when jointly learning many tasks. For this reason, we propose several additional improvements that seek to alleviate this burden: We use skip-connections, which have been shown to be useful for multi-task learning in recent work [Ruder et al., 2019a]. Furthermore, we add a task-specific layer before the output layer, which is useful for learning task-specific transformations of the shared representations [Søgaard and Goldberg, 2016, Ruder et al., 2019a].

6.2.4 Experiments

For our experiments, we evaluate on a wide range of text classification tasks. In particular, we choose pairwise classification tasks—i.e. those that condition the reading of one sequence on another sequence—as we are interested in understanding if knowledge can be transferred even for these more complex interactions. To the best of our knowledge, this is the first work on transfer learning between such pairwise sequence classification tasks. We implement all our models in Tensorflow [Abadi et al., 2016] and release the code at `https://github.com/coastalcph/mtl-disparate`.

[1] For more information regarding the feature calculation, refer to [Ruder and Plank, 2017].

Task	Domain	# of examples	# of labels	Metric
`Topic-2`	Twitter	4,346	2	ρ^{PN}
`Topic-5`	Twitter	6,000	5	MAE^{M}
`Target`	Twitter	6,248	3	F_1^M
`Stance`	Twitter	2,914	3	F_1^{FA}
`ABSA-L`	Reviews	2,909	3	*Acc*
`ABSA-R`	Reviews	2,507	3	*Acc*
`FNC-1`	News	39,741	4	*Acc*
`MultiNLI`	Diverse	392,702	3	*Acc*

TABLE 6.7: Training set statistics and evaluation metrics of every task.

Topic-based sentiment analysis	
Tweet	No power at home, sat in the dark listening to AC/DC in the hope it'll make the electricity come back again
Topic	AC/DC
Label	positive
Target-dependent sentiment analysis	
Text	how do you like settlers of catan for the wii?
Target	wii
Label	neutral
Aspect-based sentiment analysis	
Text	For the price, you cannot eat this well in Manhattan
Aspects	restaurant prices, food quality
Label	positive
Stance detection	
Tweet	Be prepared - if we continue the policies of the liberal left, we will be #Greece
Target	Donald Trump
Label	favor
Fake news detection	
Document	Dino Ferrari hooked the whopper wels catfish, (...), which could be the biggest in the world.
Headline	Fisherman lands 19 STONE catfish which could be the biggest in the world to be hooked
Label	agree
Natural language inference	
Premise	Fun for only children
Hypothesis	Fun for adults and children
Label	contradiction

TABLE 6.8: Example instances from the employed datasets.

Tasks and datasets We use the following tasks and datasets for our experiments, show task statistics in Table 6.7, and summarise examples in Table 6.8:

Topic-based sentiment analysis Topic-based sentiment analysis aims to estimate the sentiment of a tweet known to be about a given topic. We use the data from SemEval-2016 Task 4 Subtask B and C [Nakov et al., 2016] for predicting on a two-point scale of positive and negative (`Topic-2`) and five-point scale ranging from highly negative to highly positive (`Topic-5`) respectively. An example from this dataset would be to

classify the tweet "No power at home, sat in the dark listening to AC/DC in the hope it'll make the electricity come back again" known to be about the topic "AC/DC", which is labelled as a positive sentiment. The evaluation metrics for `Topic-2` and `Topic-5` are macro-averaged recall (ρ^{PN}) and macro-averaged mean absolute error (MAE^M) respectively, which are both averaged across topics.

Target-dependent sentiment analysis Target-dependent sentiment analysis (`Target`) seeks to classify the sentiment of a text's author towards an entity that occurs in the text as positive, negative, or neutral. We use the data from [Dong et al., 2014] [Dong et al., 2014]. An example instance is the expression "how do you like settlers of catan for the wii?" which is labelled as neutral towards the target "wii'.' The evaluation metric is macro-averaged F_1 (F_1^M).

Aspect-based sentiment analysis Aspect-based sentiment analysis is the task of identifying whether an aspect, i.e. a particular property of an item is associated with a positive, negative, or neutral sentiment [Ruder et al., 2016a]. We use the data of SemEval-2016 Task 5 Subtask 1 Slot 3 [Pontiki et al., 2016] for the laptops (`ABSA-L`) and restaurants (`ABSA-R`) domains. An example is the sentence "For the price, you cannot eat this well in Manhattan", labelled as positive towards both the aspects "restaurant prices" and "food quality". The evaluation metric for both domains is accuracy (*Acc*).

Stance detection Stance detection (`Stance`) requires a model, given a text and a target entity, which might not appear in the text, to predict whether the author of the text is in favour or against the target or whether neither inference is likely [Augenstein et al., 2016]. We use the data of SemEval-2016 Task 6 Subtask B [Mohammad et al., 2016]. An example from this dataset would be to predict the stance of the tweet "Be prepared - if we continue the policies of the liberal left, we will be #Greece" towards the topic "Donald Trump", labelled as "favor". The evaluation metric is the macro-averaged F_1 score of the "favour" and "against" classes (F_1^{FA}).

Fake news detection The goal of fake news detection in the context of the Fake News Challenge[2] is to estimate whether the body of a news article agrees, disagrees, discusses, or is unrelated towards a headline. We use the data from the first stage of the Fake News Challenge (`FNC-1`). An example for this dataset is the document "Dino Ferrari hooked the whopper wels catfish, (...), which could be the biggest in the world." with the

[2] `http://www.fakenewschallenge.org/`

headline "Fisherman lands 19 STONE catfish which could be the biggest in the world to be hooked" labelled as "agree". The evaluation metric is accuracy (*Acc*)[3].

Natural language inference Natural language inference is the task of predicting whether one sentences entails, contradicts, or is neutral towards another one. We use the Multi-Genre NLI corpus (`MultiNLI`) from the RepEval 2017 shared task [Nangia et al., 2017]. An example for an instance would be the sentence pair "Fun for only children", "Fun for adults and children", which are in a "contradiction" relationship. The evaluation metric is accuracy (*Acc*).

Base model Our base model is the Bidirectional Encoding model [Augenstein et al., 2016], a state-of-the-art model for stance detection that conditions a bidirectional LSTM (BiLSTM) encoding of a text on the BiLSTM encoding of the target. Unlike [Augenstein et al., 2016], we do not pre-train word embeddings on a larger set of unlabelled in-domain text for each task as we are mainly interested in exploring the benefit of multi-task learning for generalisation.

Training settings We use BiLSTMs with one hidden layer of 100 dimensions, 100-dimensional randomly initialised word embeddings, a label embedding size of 100. We train our models with RMSProp, a learning rate of 0.001, a batch size of 128, and early stopping on the validation set of the main task with a patience of 3.

	`Stance`	`FNC`	`MultiNLI`	`Topic-2`	`Topic-5*`	`ABSA-L`	`ABSA-R`	`Target`
Augenstein et al. [2016]	**49.01**	-	-	-	-	-	-	-
Riedel et al. [2017]	-	**88.46**	-	-	-	-	-	-
Chen et al. [2017b]	-	-	**74.90**	-	-	-	-	-
Palogiannidi et al. [2016]	-	-	-	79.90	-	-	-	-
Balikas and Amini [2016]	-	-	-	-	**0.719**	-	-	-
Brun et al. [2016]	-	-	-	-	-	**88.13**	-	-
Kumar et al. [2016b]	-	-	-	-	-	86.73	82.77	-
Vo and Zhang [2015]	-	-	-	-	-	-	-	**69.90**
STL	41.1	72.72	49.25	63.92	0.919	76.74	67.47	64.01
MTL + LEL	46.26	72.71	49.94	**80.52**	0.814	74.94	79.90	66.42
MTL + LEL + LTN, main model	43.16	72.73	48.75	73.90	0.810	75.06	**83.71**	66.10
MTL + LEL + LTN + semi, main model	43.56	72.72	48.00	72.35	0.821	75.42	83.26	63.00

TABLE 6.9: Comparison of our best performing models against the state-of-the-art.

Results Our main results are shown in Table 6.9, with a comparison against a single task (STL) baseline against the state-of-the-art with task-specific metrics. We present the results of our multi-task learning network with label embeddings (MTL + LEL), multi-task learning with label transfer (MTL + LEL + LTN), and the semi-supervised

[3]We use the same metric as [Riedel et al., 2017].

extension of this model. On 7/8 tasks, at least one of our architectures is better than single-task learning; and in 4/8, all our architectures are much better than single-task learning.

The state-of-the-art systems we compare against are often highly specialised, task-dependent architectures. Our architectures, in contrast, have not been optimised to compare favourably against the state of the art, as our main objective is to develop a novel approach to multi-task learning leveraging synergies between label sets and knowledge of marginal distributions from unlabeled data. For example, we do not use pre-trained word embeddings [Augenstein et al., 2016, Palogiannidi et al., 2016, Vo and Zhang, 2015], class weighting to deal with label imbalance [Balikas and Amini, 2016], or domain-specific sentiment lexicons [Brun et al., 2016, Kumar et al., 2016b]. Nevertheless, our approach outperforms the state-of-the-art on two-way topic-based sentiment analysis (`Topic-2`) and aspect-based sentiment analysis on restaurant reviews (`ABSA-R`). This is particularly surprising for the restaurants domain, as none of the datasets of other tasks are in the same domain.

The poor performance compared to the state-of-the-art on `FNC` and `MultiNLI` is expected; as we alternate among the tasks during training, our model only sees a comparatively small number of examples of both corpora, which are one and two orders of magnitude larger than the other datasets. For this reason, we do not achieve good performance on these tasks as main tasks, but they are still useful as auxiliary tasks as seen in Table 6.10.

6.2.5 Analysis

Label Embeddings Our results above show that, indeed, modelling the similarity between tasks using label embeddings sometimes leads to much better performance. Figure 6.5 shows why. In Figure 6.5, we visualise the label embeddings of an MTL+LEL model trained on all tasks, using PCA. As we can see, similar labels are clustered together across tasks, e.g. there are two positive clusters (middle-right and top-right), two negative clusters (middle-left and bottom-left), and two neutral clusters (middle-top and middle-bottom).

Our visualisation also provides us with a picture of what auxilary tasks are beneficial, and to what extent we can expect synergies from multi-task learning. For instance, the notion of positive sentiment appears to be very similar across the topic-based and aspect-based tasks, while the conceptions of negative and neutral sentiment differ. In addition, we can see that the model has failed to learn a relationship between `MultiNLI` labels and those of other tasks, possibly accounting for its poor performance on the

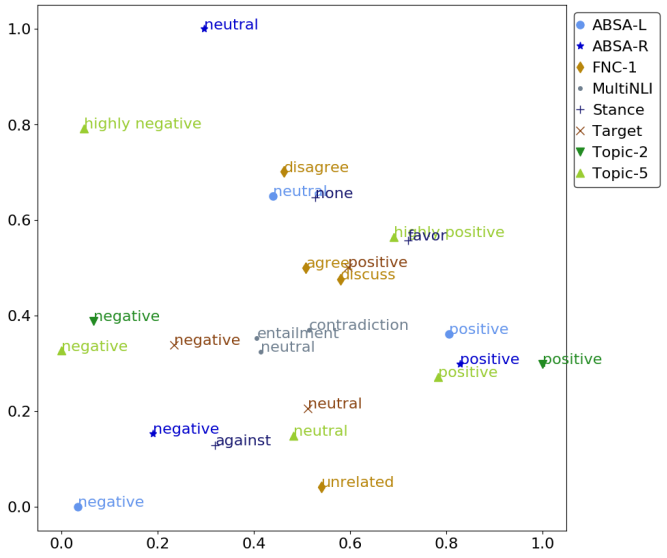


FIGURE 6.5: Label embeddings of all tasks.

inference task. We did not evaluate the correlation between label embeddings and task performance, but [Bjerva, 2017] recently suggested that mutual information of target and auxiliary task label sets is a good predictor of gains from multi-task learning.

Main task	Auxiliary tasks
`Topic-2`	`FNC-1`, `MultiNLI`, `Target`
`Topic-5`	`FNC-1`, `MultiNLI`, `ABSA-L`, `Target`
`Target`	`FNC-1`, `MultiNLI`, `Topic-5`
`Stance`	`FNC-1`, `MultiNLI`, `Target`
`ABSA-L`	`Topic-5`
`ABSA-R`	`Topic-5`, `ABSA-L`, `Target`
`FNC-1`	`Stance`, `MultiNLI`, `Topic-5`, `ABSA-R`, `Target`
`MultiNLI`	`Topic-5`

TABLE 6.10: Best-performing auxiliary tasks for different main tasks.

Auxilary Tasks For each task, we show the auxiliary tasks that achieved the best performance in Table 6.10. In contrast to most existing work, we did not restrict ourselves to performing multi-task learning with only one auxiliary task [Søgaard and Goldberg, 2016, Bingel and Søgaard, 2017]. Indeed we find that most often a combination of auxiliary tasks achieves the best performance. In-domain tasks are less used than we assumed; only `Target` is consistently used by all Twitter main tasks. In addition, tasks with a higher number of labels, e.g. `Topic-5` are used more often. Such tasks provide a more fine-grained reward signal, which may help in learning representations that generalise better. Finally, tasks with large amounts of training data such as `FNC-1` and

`MultiNLI` are also used more often. Even if not directly related, the larger amount of training data that can be indirectly leveraged via multi-task learning may help the model focus on relevant parts of the representation space [Caruana, 1993]. These observations shed additional light on when multi-task learning may be useful that go beyond existing studies [Bingel and Søgaard, 2017].

Ablation analysis We now perform a detailed ablation analysis of our model, testing different variants of our model with early stopping on the development set. We report task-specific evaluation metrics and show results in Table 6.11. Note that for `MultiNLI`, we down-sample the training data.

	Stance	FNC	MultiNLI	Topic-2	Topic-5*	ABSA-L	ABSA-R	Target
MTL	44.12	72.75	49.39	**80.74**	0.859	74.94	82.25	65.73
MTL + LEL	**46.26**	72.71	**49.94**	80.52	0.814	74.94	79.90	**66.42**
MTL + LTN	40.95	72.72	44.14	78.31	0.851	73.98	82.37	63.71
MTL + LTN, main model	41.60	72.72	47.62	79.98	0.814	75.54	81.70	65.61
MTL + LEL + LTN	44.48	**72.76**	43.72	74.07	0.821	**75.66**	81.92	65.00
MTL + LEL + LTN, main model	43.16	72.73	48.75	73.90	0.810	75.06	**83.71**	66.10
MTL + LEL + LTN + main preds feats	42.78	72.72	45.41	66.30	0.835	73.86	81.81	65.08
MTL + LEL + LTN + main preds feats, main model	42.65	72.73	48.81	67.53	**0.803**	75.18	82.59	63.95
MTL + LEL + LTN + main preds feats – diversity feats	42.78	72.72	43.13	66.3	0.835	73.5	81.7	63.95
MTL + LEL + LTN + main preds feats – diversity feats, main model	42.47	72.74	47.84	67.53	0.807	74.82	82.14	65.11
MTL + LEL + LTN + semi	42.65	72.75	44.28	77.81	0.841	74.10	81.36	64.45
MTL + LEL + LTN + semi, main model	43.56	72.72	48.00	72.35	0.821	75.42	83.26	63.00

TABLE 6.11: Ablation results with task-specific evaluation metrics. LTN - main preds feats means the output of the relabelling function is shown, which does not use the task predictions, only predictions from other tasks. MTL + EM, main model means that the main model predictions of the model that trains a relabelling function are used. *: lower is better. Bold: best. Underlined: second-best.

We ablate whether to use the LEL (*+ LEL*), whether to use the LTN (*+ LTN*), whether to use the LEL output or the main model output for prediction (main model output is indicated by *, main model*), and whether to use the LTN as a regulariser or for semi-supervised learning (semi-supervised learning is indicated by *+ semi*). We further test whether to use diversity features (*– diversity feats*) and whether to use main model predictions for the LTN (*+ main model feats*).

Overall, the addition of the Label Embedding Layer improves the performance over regular MTL in almost all cases.

Label transfer network To understand the performance of the LTN, we analyse learning curves of the relabelling function vs. the main model on the development set. The main model is pre-trained for 10 epochs, after which the relabelling function is trained. Examples for all tasks without semi-supervised learning are shown in Figure 6.6.

One can observe that the relabelling model does not take long to converge as it has fewer parameters than the main model. Once the relabelling model is learned alongside

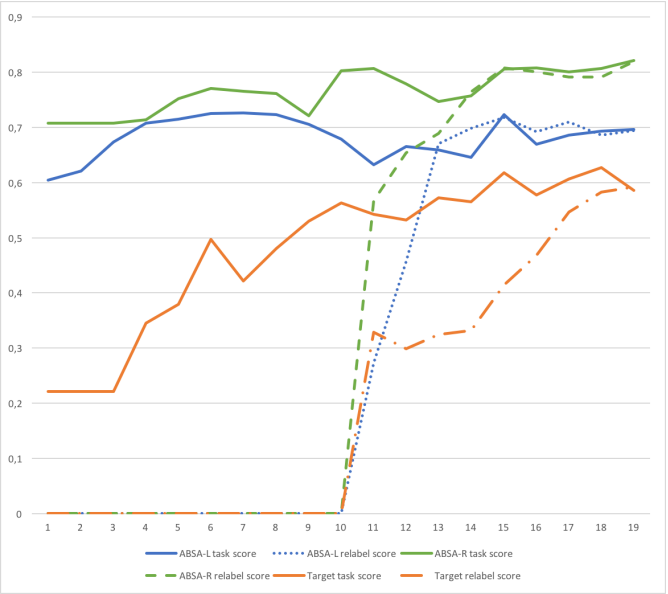


FIGURE 6.6: Development learning curves with Label Transfer Network for selected tasks.

the main model, the main model performance first stagnates, then starts to increase again. For some of the tasks, the main model ends up with a higher task score than the relabelling model. We hypothesise that this means the softmax predictions of other, even highly related tasks are less helpful for predicting main labels than the output layer of the main task model. At best, learning the relabelling model alongside the main model might act as a regulariser to the main model and thus improve the main model's performance over a baseline MTL model, as it is the case for `TOPIC-5` (see Table 6.11).

To further analyse the performance of the LTN, we look into to what degree predictions of the main model and the relabelling model for individual instances are complementary to one another. Or, said differently, we measure the percentage of correct predictions made only by the relabelling model or made only by the main model, relative to the number of correct predictions overall. Results of this for each task are shown in Table 6.12 for the LTN with and without semi-supervised learning.

Task	Main	LTN	Main (Semi)	LTN (Semi)
`Stance`	2.12	2.62	1.94	1.28
`FNC`	4.28	2.49	6.92	4.84
`MultiNLI`	1.5	1.95	1.94	1.28
`Topic-2`	6.45	4.44	5.87	5.59
`Topic-5*`	9.22	9.71	11.3	5.90
`ABSA-L`	3.79	2.52	9.06	6.63
`ABSA-R`	10.6	6.70	9.06	6.63
`Target`	26.3	14.6	20.1	15.7

TABLE 6.12: Error analysis of LTN with and without semi-supervised learning.

We use the percentage of correct predictions only made by either the relabelling function or the main model respectively, relative to the the number of all correct predictions as metric. One can observe that, even though the relabelling function overall contributes to the score to a lesser degree than the main model, a substantial number of correct predictions are made by the relabelling function that are missed by the main model. This is most prominently pronounced for `ABSA-R`, where the proportion is 14.6.

6.2.6 Summary

We have presented a multi-task learning architecture that (i) leverages potential synergies between classifier functions relating shared representations with disparate label spaces and (ii) enables learning from mixtures of labeled and unlabeled data. We have presented experiments with combinations of eight pairwise sequence classification tasks. Our results show that leveraging synergies between label spaces sometimes leads to big improvements, and we have presented a new state of the art for aspect-based and topic-based sentiment analysis. Our analysis further showed that (a) the learned label embeddings were indicative of gains from multi-task learning, (b) auxiliary tasks were often beneficial across domains, and (c) label embeddings almost always led to better performance. We also investigated the dynamics of the label transfer network we use for exploiting the synergies between disparate label spaces.

6.3 Conclusions

In this chapter, we have presented two novel multi-task learning approaches that learn to share different parts of the architecture. We have evaluated these architectures on a wide range of tasks and found that they outperform both single task and state-of-the-art multi-task learning approaches.

In the next chapter, we will finally tackle sequential transfer learning, the most ubiquitous transfer learning setting with neural networks, which transfers a trained model to a target task rather than learning both tasks jointly. We will first propose a novel sequential transfer learning framework based on fine-tuning language models and then extensively study the adaptation phase.

Chapter 7

Adapting Universal Pretrained Representations

Multi-task learning has become a common tool in many scenarios involving neural networks. However, in practice sequential transfer learning likely is the most common transfer learning setting, as pretrained representations provide a significant performance boost and can be transferred efficiently to a target task (§3.3). In this chapter, we tackle the most general setting of sequential transfer learning and aim to pretrain representations that can be adapted to any target task. As most prior work has focused on the pretraining stage (§3.3.2), we in particular analyze and propose novel techniques for the adaptation phase.

In Section 7.1, we propose Universal Language Model Fine-Tuning (ULMFiT), a novel framework for pretraining and adapting learned representations. We first pretrain a state-of-the-art language model on a large general-domain corpus and then fine-tune it on the target task. We propose several methods for adaptation, such as slanted triangular learning rates, discriminative fine-tuning, and gradual unfreezing that together facilitate the transfer of information to the target task. Our framework achieves significant improvements over the state-of-the-art across a wide range of text classification tasks.

In Section 7.2, we analyze the adaptation phase in more detail. We compare the two main adaptation paradigms, feature extraction and fine-tuning with two state-of-the-art pretrained language model representations across a diverse set of tasks. We also evaluate the usefulness of the methods we introduced in the previous section. We find that the relative performance of both adaptation approaches depends on the similarity of the pretraining and target tasks. Specifically, fine-tuning performs better on similar tasks. We finally provide practical guidelines to practitioners.

7.1 Universal Language Model Fine-tuning for Text Classification

* Inductive transfer learning has greatly impacted computer vision, but existing approaches in NLP still require task-specific modifications and training from scratch. We propose Universal Language Model Fine-tuning (ULMFiT), an effective transfer learning method that can be applied to any task in NLP, and introduce techniques that are key for fine-tuning a language model. Our method significantly outperforms the state-of-the-art on six text classification tasks, reducing the error by 18-24% on the majority of datasets. Furthermore, with only 100 labeled examples, it matches the performance of training from scratch on 100× more data. We open-source our pretrained models and code[†].

7.1.1 Introduction

Inductive transfer learning has had a large impact on computer vision (CV). Applied CV models (including object detection, classification, and segmentation) are rarely trained from scratch, but instead are fine-tuned from models that have been pretrained on ImageNet, MS-COCO, and other datasets [Sharif Razavian et al., 2014, Long et al., 2015a, He et al., 2016, Huang et al., 2017b].

Text classification is a category of Natural Language Processing (NLP) tasks with real-world applications such as spam, fraud, and bot detection [Jindal and Liu, 2007, Ngai et al., 2011, Chu et al., 2012], emergency response [Caragea et al., 2011], and commercial document classification, such as for legal discovery [Roitblat et al., 2010].

While Deep Learning models have achieved state-of-the-art on many NLP tasks, these models are trained from scratch, requiring large datasets, and days to converge. Research in NLP focused mostly on *transductive* transfer [Blitzer et al., 2007]. For *inductive* transfer, fine-tuning pretrained word embeddings [Mikolov et al., 2013a], a simple transfer technique that only targets a model's first layer, has had a large impact in practice and is used in most state-of-the-art models. Recent approaches that concatenate embeddings derived from other tasks with the input at different layers [Peters et al., 2017, McCann et al., 2017, Peters et al., 2018a] still train the main task model from scratch and treat pretrained embeddings as fixed parameters, limiting their usefulness.

In light of the benefits of pretraining [Erhan et al., 2010], we should be able to do better than *randomly initializing* the remaining parameters of our models. However, inductive

This section is adapted from: Howard, J. and **Ruder, S.*** (2018). Universal Language Model Fine-tuning for Text Classification. In *Proceedings of ACL 2018*. Jeremy focused on the implementation. Sebastian focused on the experiments and writing.

[†]`http://nlp.fast.ai/ulmfit`.

transfer via fine-tuning has been unsuccessful for NLP [Mou et al., 2016]. Dai and Le [2015] first proposed fine-tuning a language model (LM) but require millions of in-domain documents to achieve good performance, which severely limits its applicability.

We show that not the idea of LM fine-tuning but our lack of knowledge of how to train them effectively has been hindering wider adoption. LMs overfit to small datasets and suffered catastrophic forgetting when fine-tuned with a classifier. Compared to CV, NLP models are typically more shallow and thus require different fine-tuning methods.

We propose a new method, Universal Language Model Fine-tuning (ULMFiT) that addresses these issues and enables robust inductive transfer learning for any NLP task, akin to fine-tuning ImageNet models: The same 3-layer LSTM architecture—with the same hyperparameters and no additions other than tuned dropout hyperparameters—outperforms highly engineered models and transfer learning approaches on six widely studied text classification tasks. On IMDb, with 100 labeled examples, ULMFiT matches the performance of training from scratch with 10× and—given 50k unlabeled examples—with 100× more data.

Contributions Our contributions are the following: 1) We propose Universal Language Model Fine-tuning (ULMFiT), a method that can be used to achieve CV-like transfer learning for any task for NLP. 2) We propose *discriminative fine-tuning*, *slanted triangular learning rates*, and *gradual unfreezing*, novel techniques to retain previous knowledge and avoid catastrophic forgetting during fine-tuning. 3) We significantly outperform the state-of-the-art on six representative text classification datasets, with an error reduction of 18-24% on the majority of datasets. 4) We show that our method enables extremely sample-efficient transfer learning and perform an extensive ablation analysis. 5) We make the pretrained models and our code available to enable wider adoption.

7.1.2 Related work

Transfer learning in CV Features in deep neural networks in CV have been observed to transition from *general* to task-*specific* from the first to the last layer [Yosinski et al., 2014]. For this reason, most work in CV focuses on transferring the first layers of the model [Long et al., 2015b]. Sharif Razavian et al. [2014] achieve state-of-the-art results using features of an ImageNet model as input to a simple classifier. In recent years, this approach has been superseded by fine-tuning either the last [Donahue et al., 2014] or several of the last layers of a pretrained model and leaving the remaining layers frozen [Long et al., 2015a].

Hypercolumns In NLP, only recently have methods been proposed that go beyond transferring word embeddings. The prevailing approach is to pretrain embeddings that capture additional context via other tasks. Embeddings at different levels are then used as features, concatenated either with the word embeddings or with the inputs at intermediate layers. This method is known as hypercolumns [Hariharan et al., 2015] in CV[1] and is used by Peters et al. [2017], Peters et al. [2018a], Wieting and Gimpel [2017], Conneau et al. [2017], and McCann et al. [2017] who use language modeling, paraphrasing, entailment, and MT respectively for pretraining. Specifically, Peters et al. [2018a] require engineered custom architectures, while we show state-of-the-art performance with the same basic architecture across a range of tasks. In CV, hypercolumns have been nearly entirely superseded by end-to-end fine-tuning [Long et al., 2015a].

Multi-task learning A related direction is multi-task learning (§3.2). This is the approach taken by Rei [2017] and Liu et al. [2018a] who add a language modeling objective to the model that is trained jointly with the main task model. MTL requires the tasks to be trained from scratch every time, which makes it inefficient and often requires careful weighting of the task-specific objective functions [Chen et al., 2017c].

Fine-tuning Fine-tuning has been used successfully to transfer between similar tasks, e.g. in QA [Min et al., 2017], for distantly supervised sentiment analysis [Severyn and Moschitti, 2015], or MT domains [Sennrich et al., 2015] but has been shown to fail between unrelated ones [Mou et al., 2016]. Dai and Le [2015] also fine-tune a language model, but overfit with 10k labeled examples and require millions of in-domain documents for good performance. In contrast, ULMFiT leverages general-domain pretraining and novel fine-tuning techniques to prevent overfitting even with only 100 labeled examples and achieves state-of-the-art results also on small datasets.

7.1.3 Universal Language Model Fine-tuning

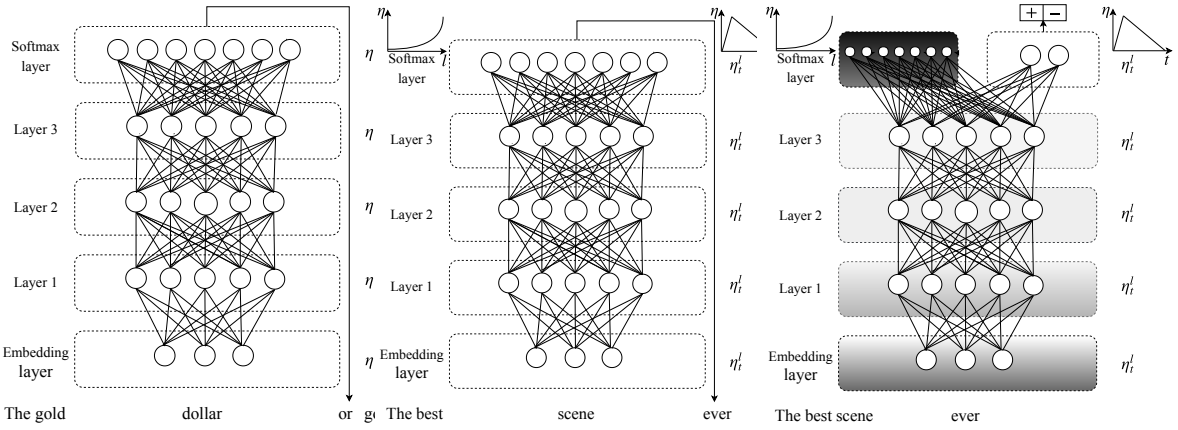
We are interested in the most general *inductive* transfer learning setting for NLP [Pan and Yang, 2010]: Given a static source task $\mathcal{T}_S$ and *any* target task $\mathcal{T}_T$ with $\mathcal{T}_S \neq \mathcal{T}_T$, we would like to improve performance on $\mathcal{T}_T$. Language modeling can be seen as the ideal source task and a counterpart of ImageNet for NLP: It captures many facets of language relevant for downstream tasks, such as long-term dependencies [Linzen et al., 2016], hierarchical relations [Gulordava et al., 2018], and sentiment [Radford et al., 2017].

[1] A hypercolumn at a pixel in CV is the vector of activations of all CNN units above that pixel. In analogy, a hypercolumn for a word or sentence in NLP is the concatenation of embeddings at different layers in a pretrained model.

In contrast to tasks like MT [McCann et al., 2017] and entailment [Conneau et al., 2017], it provides data in near-unlimited quantities for most domains and languages. Additionally, a pretrained LM can be easily adapted to the idiosyncrasies of a target task, which we show significantly improves performance (see Section 7.1.5). Moreover, language modeling already is a key component of existing tasks such as MT and dialogue modeling. Formally, language modeling induces a hypothesis space $\mathcal{H}$ that should be useful for many other NLP tasks [Vapnik and Kotz, 1982, Baxter, 2000].

We propose Universal Language Model Fine-tuning (ULMFiT), which pretrains a language model (LM) on a large general-domain corpus and fine-tunes it on the target task using novel techniques. The method is *universal* in the sense that it meets these practical criteria: 1) It works across tasks varying in document size, number, and label type; 2) it uses a single architecture and training process; 3) it requires no custom feature engineering or preprocessing; and 4) it does not require additional in-domain documents or labels.

In our experiments, we use the state-of-the-art language model AWD-LSTM [Merity et al., 2017a], a regular LSTM (with no attention, short-cut connections, or other sophisticated additions) with various tuned dropout hyperparameters. Analogous to CV, we expect that downstream performance can be improved by using higher-performance language models in the future.



(A) LM pre-training (B) LM fine-tuning (C) Classifier fine-tuning

FIGURE 7.1: ULMFiT consists of three stages: a) The LM is trained on a general-domain corpus to capture general features of the language in different layers. b) The full LM is fine-tuned on target task data using discriminative fine-tuning ('*Discr*') and slanted triangular learning rates (STLR) to learn task-specific features. c) The classifier is fine-tuned on the target task using gradual unfreezing, '*Discr*', and STLR to preserve low-level representations and adapt high-level ones (shaded: unfreezing stages; black: frozen).

ULMFiT consists of the following steps, which we show in Figure 7.1: a) General-domain LM pretraining (§7.1.3.1); b) target task LM fine-tuning (§7.1.3.2); and c) target task

classifier fine-tuning (§7.1.3.3). In a), the LM is trained on a general-domain corpus to capture general features of the language in different layers. In b), the full LM is fine-tuned on target task data using discriminative fine-tuning ('*Discr*') and slanted triangular learning rates (STLR) to learn task-specific features. In c), the classifier is fine-tuned on the target task using gradual unfreezing, '*Discr*', and STLR to preserve low-level representations and adapt high-level ones. Black layers in the figure remain frozen throughout training and shaded layers are gradually unfrozen. We discuss the stages in more detail in the following sections.

7.1.3.1 General-domain LM pretraining

An ImageNet-like corpus for language should be large and capture general properties of language. We pretrain the language model on Wikitext-103 [Merity et al., 2017b] consisting of 28,595 preprocessed Wikipedia articles and 103 million words. Pretraining is most beneficial for tasks with small datasets and enables generalization even with 100 labeled examples. We leave the exploration of more diverse pretraining corpora to future work, but expect that they would boost performance. While this stage is the most expensive, it only needs to be performed once and improves performance and convergence of downstream models.

7.1.3.2 Target task LM fine-tuning

No matter how diverse the general-domain data used for pretraining is, the data of the target task will likely come from a different distribution. We thus fine-tune the LM on data of the target task. Given a pretrained general-domain LM, this stage converges faster as it only needs to adapt to the idiosyncrasies of the target data, and it allows us to train a robust LM even for small datasets. We propose *discriminative fine-tuning* and *slanted triangular learning rates* for fine-tuning the LM, which we introduce in the following.

Discriminative fine-tuning As different layers capture *different types of information* [Yosinski et al., 2014], they should be fine-tuned to *different extents*. To this end, we propose a novel fine-tuning method, *discriminative fine-tuning*[2].

Instead of using the same learning rate for *all* layers of the model, discriminative fine-tuning allows us to tune *each* layer with different learning rates. For context, the regular

[2] An unrelated method of the same name exists for deep Boltzmann machines [Salakhutdinov and Hinton, 2009].

stochastic gradient descent (SGD) update of a model's parameters θ at time step t looks like the following [Ruder, 2016]:

$$\theta_t = \theta_{t-1} - \eta \cdot \nabla_\theta J(\theta) \tag{7.1}$$

where η is the learning rate and $\nabla_\theta J(\theta)$ is the gradient with regard to the model's objective function. For discriminative fine-tuning, we split the parameters θ into $\{\theta^1, \ldots, \theta^L\}$ where θ^l contains the parameters of the model at the l-th layer and L is the number of layers of the model. Similarly, we obtain $\{\eta^{(1)}, \ldots, \eta^{(L)}\}$ where $\eta^{(l)}$ is the learning rate of the l-th layer.

The SGD update with discriminative fine-tuning is then the following:

$$\theta_t^{(l)} = \theta_{t-1}^{(l)} - \eta^{(l)} \cdot \nabla_{\theta^{(l)}} J(\theta) \tag{7.2}$$

We empirically found it to work well to first choose the learning rate $\eta^{(L)}$ of the last layer by fine-tuning only the last layer and using $\eta^{(l-1)} = \eta^{(l)}/2.6$ as the learning rate for lower layers.

Slanted triangular learning rates For adapting its parameters to task-specific features, we would like the model to quickly converge to a suitable region of the parameter space in the beginning of training and then refine its parameters. Using the same learning rate (LR) or an annealed learning rate throughout training is not the best way to achieve this behaviour. Instead, we propose *slanted triangular learning rates* (STLR), which first linearly increases the learning rate and then linearly decays it according to the following update schedule, which can be seen in Figure 7.2:

$$\begin{aligned} cut &= \lfloor T \cdot cut_frac \rfloor \\ p &= \begin{cases} t/cut, & \text{if } t < cut \\ 1 - \frac{t - cut}{cut \cdot (1/cut_frac - 1)}, & \text{otherwise} \end{cases} \\ \eta_t &= \eta_{max} \cdot \frac{1 + p \cdot (ratio - 1)}{ratio} \end{aligned} \tag{7.3}$$

where T is the number of training iterations[3], cut_frac is the fraction of iterations we increase the LR, cut is the iteration when we switch from increasing to decreasing the LR, p is the fraction of the number of iterations we have increased or will decrease the LR respectively, $ratio$ specifies how much smaller the lowest LR is from the maximum

[3]In other words, the number of epochs times the number of updates per epoch.

LR η_{max}, and η_t is the learning rate at iteration t. We generally use $cut_frac = 0.1$, $ratio = 32$ and $\eta_{max} = 0.01$.

STLR modifies triangular learning rates [Smith, 2017] with a short increase and a long decay period, which we found key for good performance.[4] In Section 7.1.5, we compare against aggressive cosine annealing, a similar schedule that has recently been used to achieve state-of-the-art performance in CV [Loshchilov and Hutter, 2017b].[5]

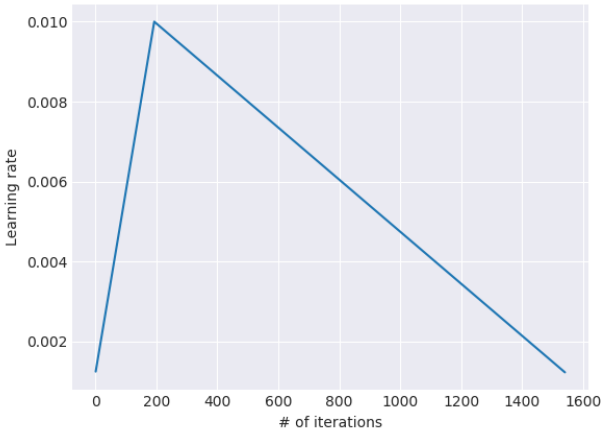


FIGURE 7.2: The slanted triangular learning rate schedule used for ULMFiT as a function of the number of training iterations.

7.1.3.3 Target task classifier fine-tuning

Finally, for fine-tuning the classifier, we augment the pretrained language model with two additional linear blocks. Following standard practice for CV classifiers, each block uses batch normalization [Ioffe and Szegedy, 2015] and dropout, with ReLU activations for the intermediate layer and a softmax activation that outputs a probability distribution over target classes at the last layer. Note that the parameters in these task-specific classifier layers are the only ones that are learned from scratch. The first linear layer takes as the input the pooled last hidden layer states.

Concat pooling The signal in text classification tasks is often contained in a few words, which may occur anywhere in the document. As input documents can consist of hundreds of words, information may get lost if we only consider the last hidden state of the model. For this reason, we concatenate the hidden state at the last time step $\mathbf{h}_T$

[4] We also credit personal communication with the author.

[5] While Loshchilov and Hutter [2017b] use multiple annealing cycles, we generally found one cycle to work best.

Dataset	Type	# classes	# examples
TREC-6	Question	6	5.5k
IMDb	Sentiment	2	25k
Yelp-bi	Sentiment	2	560k
Yelp-full	Sentiment	5	650k
AG	Topic	4	120k
DBpedia	Topic	14	560k

TABLE 7.1: Text classification datasets and tasks with number of classes and training examples.

of the document with both the max-pooled and the mean-pooled representation of the hidden states over as many time steps as fit in GPU memory $\mathbf{H} = \{\mathbf{h}_1, \ldots, \mathbf{h}_T\}$:

$$\mathbf{h}_c = [\mathbf{h}_T, \texttt{maxpool}(\mathbf{H}), \texttt{meanpool}(\mathbf{H})] \tag{7.4}$$

where [] is concatenation.

Fine-tuning the target classifier is the most critical part of the transfer learning method. Overly aggressive fine-tuning will cause catastrophic forgetting, eliminating the benefit of the information captured through language modeling; too cautious fine-tuning will lead to slow convergence (and resultant overfitting). Besides discriminative fine-tuning and triangular learning rates, we propose *gradual unfreezing* for fine-tuning the classifier.

Gradual unfreezing Rather than fine-tuning all layers at once, which risks catastrophic forgetting, we propose to gradually unfreeze the model starting from the last layer as this contains the *least general* knowledge [Yosinski et al., 2014]: We first unfreeze the last layer and fine-tune all unfrozen layers for one epoch. We then unfreeze the next lower frozen layer and repeat, until we fine-tune all layers until convergence at the last iteration. This is similar to '*chain-thaw*' [Felbo et al., 2017], except that we add a layer at a time to the set of 'thawed' layers, rather than only training a single layer at a time.

While discriminative fine-tuning, slanted triangular learning rates, and gradual unfreezing all are beneficial on their own, we show in Section 7.1.5 that they complement each other and enable our method to perform well across diverse datasets.

BPTT for Text Classification (BPT3C) Language models are trained with back-propagation through time (BPTT) to enable gradient propagation for large input sequences. In order to make fine-tuning a classifier for large documents feasible, we propose BPTT for Text Classification (BPT3C): We divide the document into fixed-length batches of size b. At the beginning of each batch, the model is initialized with the final state

of the previous batch; we keep track of the hidden states for mean and max-pooling; gradients are back-propagated to the batches whose hidden states contributed to the final prediction. In practice, we use variable length backpropagation sequences [Merity et al., 2017a].

Bidirectional language model Similar to existing work [Peters et al., 2017, 2018a], we are not limited to fine-tuning a unidirectional language model. For all our experiments, we pretrain both a forward and a backward LM. We fine-tune a classifier for each LM independently using BPT3C and average the classifier predictions.

7.1.4 Experiments

While our approach is equally applicable to sequence labeling tasks, we focus on text classification tasks in this work due to their important real-world applications.

7.1.4.1 Experimental setup

Datasets and tasks We evaluate our method on six widely-studied datasets, with varying numbers of documents and varying document length, used by state-of-the-art text classification and transfer learning approaches [Johnson and Zhang, 2017, McCann et al., 2017] as instances of three common text classification tasks: sentiment analysis, question classification, and topic classification. We show the statistics for each dataset and task in Table 7.1.

Sentiment Analysis For sentiment analysis, we evaluate our approach on the binary movie review IMDb dataset [Maas et al., 2011] and on the binary and five-class version of the Yelp review dataset compiled by Zhang et al. [2015].

Question Classification We use the six-class version of the small TREC dataset [Voorhees and Tice, 1999] dataset of open-domain, fact-based questions divided into broad semantic categories.

Topic classification For topic classification, we evaluate on the large-scale AG news and DBpedia ontology datasets created by Zhang et al. [2015].

Pre-processing We use the same pre-processing as in earlier work [Johnson and Zhang, 2017, McCann et al., 2017]. In addition, to allow the language model to capture aspects that might be relevant for classification, we add special tokens for upper-case words, elongation, and repetition.

Hyperparameters We are interested in a model that performs robustly across a diverse set of tasks. To this end, if not mentioned otherwise, we use the same set of hyperparameters across tasks, which we tune on the IMDb validation set. We use the AWD-LSTM language model [Merity et al., 2017a] with an embedding size of 400, 3 layers, 1150 hidden activations per layer, and a BPTT batch size of 70. We apply dropout of 0.4 to layers, 0.3 to RNN layers, 0.4 to input embedding layers, 0.05 to embedding layers, and weight dropout of 0.5 to the RNN hidden-to-hidden matrix. The classifier has a hidden layer of size 50. We use Adam with $\beta_1 = 0.7$ instead of the default $\beta_1 = 0.9$ and $\beta_2 = 0.99$, similar to [Dozat and Manning, 2017]. We use a batch size of 64, a base learning rate of 0.004 and 0.01 for fine-tuning the LM and the classifier respectively, and tune the number of epochs on the validation set of each task[6]. We otherwise use the same practices used in [Merity et al., 2017a].

Baselines and comparison models For each task, we compare against the current state-of-the-art. For the IMDb and TREC-6 datasets, we compare against CoVe [McCann et al., 2017], a state-of-the-art transfer learning method for NLP. For the AG, Yelp, and DBpedia datasets, we compare against the state-of-the-art text categorization method by Johnson and Zhang [2017].

7.1.4.2 Results

For consistency, we report all results as error rates (lower is better). We show the test error rates on the IMDb and TREC-6 datasets used by McCann et al. [2017] in Table 7.2. Our method outperforms both CoVe, a state-of-the-art transfer learning method based on hypercolumns, as well as the state-of-the-art on both datasets. On IMDb, we reduce the error dramatically by 43.9% and 22% with regard to CoVe and the state-of-the-art respectively. This is promising as the existing state-of-the-art requires complex architectures [Peters et al., 2018a], multiple forms of attention [McCann et al., 2017] and sophisticated embedding schemes [Johnson and Zhang, 2016], while our method employs a regular LSTM with dropout. We note that the language model fine-tuning

[6]On small datasets such as TREC-6, we fine-tune the LM only for 15 epochs without overfitting, while we can fine-tune longer on larger datasets. We found 50 epochs to be a good default for fine-tuning the classifier.

	Model	Test		Model	Test
IMDb	CoVe [McCann et al., 2017]	8.2	TREC-6	CoVe [McCann et al., 2017]	4.2
	oh-LSTM [Johnson and Zhang, 2016]	5.9		TBCNN [Mou et al., 2015]	4.0
	Virtual [Miyato et al., 2017]	5.9		LSTM-CNN [Zhou et al., 2016b]	3.9
	ULMFiT (ours)	**4.6**		ULMFiT (ours)	**3.6**

Table 7.2: Test error rates (%) on two text classification datasets used by McCann et al. [2017].

	AG	DBpedia	Yelp-bi	Yelp-full
Char-level CNN [Zhang et al., 2015]	9.51	1.55	4.88	37.95
CNN [Johnson and Zhang, 2016]	6.57	0.84	2.90	32.39
DPCNN [Johnson and Zhang, 2017]	6.87	0.88	2.64	30.58
ULMFiT (ours)	**5.01**	**0.80**	**2.16**	**29.98**

Table 7.3: Test error rates (%) on text classification datasets used by Johnson and Zhang [2017].

approach of Dai and Le [2015] only achieves an error of 7.64 vs. 4.6 for our method on IMDb, demonstrating the benefit of transferring knowledge from a large ImageNet-like corpus using our fine-tuning techniques. IMDb in particular is reflective of real-world datasets: Its documents are generally a few paragraphs long—similar to emails (e.g for legal discovery) and online comments (e.g for community management); and sentiment analysis is similar to many commercial applications, e.g. product response tracking and support email routing.

On TREC-6, our improvement—similar as the improvements of state-of-the-art approaches—is not statistically significant, due to the small size of the 500-examples test set. Nevertheless, the competitive performance on TREC-6 demonstrates that our model performs well across different dataset sizes and can deal with examples that range from single sentences—in the case of TREC-6—to several paragraphs for IMDb. Note that despite pretraining on more than two orders of magnitude less data than the 7 million sentence pairs used by McCann et al. [2017], we consistently outperform their approach on both datasets.

We show the test error rates on the larger AG, DBpedia, Yelp-bi, and Yelp-full datasets in Table 7.3. Our method again outperforms the state-of-the-art significantly. On AG, we observe a similarly dramatic error reduction by 23.7% compared to the state-of-the-art. On DBpedia, Yelp-bi, and Yelp-full, we reduce the error by 4.8%, 18.2%, 2.0% respectively.

7.1.5 Analysis

In order to assess the impact of each contribution, we perform a series of analyses and ablations. We run experiments on three corpora, IMDb, TREC-6, and AG that are representative of different tasks, genres, and sizes. For all experiments, we split off 10% of the training set and report error rates on this validation set with unidirectional LMs. We fine-tune the classifier for 50 epochs and train all methods but ULMFiT with early stopping.

Low-shot learning One of the main benefits of transfer learning is being able to train a model for a task with a small number of labels. We evaluate ULMFiT on different numbers of labeled examples in two settings: only labeled examples are used for LM fine-tuning ('*supervised*'); and all task data is available and can be used to fine-tune the LM ('*semi-supervised*'). We compare ULMFiT to training from scratch—which is necessary for hypercolumn-based approaches. We split off balanced fractions of the training data, keep the validation set fixed, and use the same hyperparameters as before. We show the results in Figure 7.3.

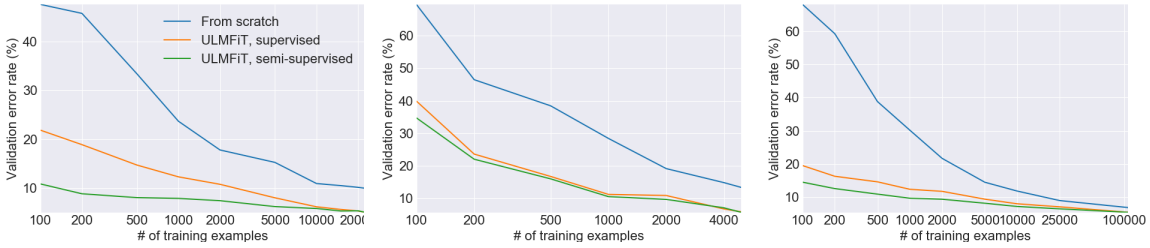


FIGURE 7.3: Validation error rates for few-shot learning with supervised and semi-supervised ULMFiT vs. training from scratch with different numbers of training examples on IMDb, TREC-6, and AG (from left to right).

On IMDb and AG, supervised ULMFiT with only 100 labeled examples matches the performance of training from scratch with 10× and 20× more data respectively, clearly demonstrating the benefit of general-domain LM pretraining. If we allow ULMFiT to also utilize unlabeled examples (50k for IMDb, 100k for AG), at 100 labeled examples, we match the performance of training from scratch with 50× and 100× more data on AG and IMDb respectively. On TREC-6, ULMFiT significantly improves upon training from scratch; as examples are shorter and fewer, supervised and semi-supervised ULMFiT achieve similar results.

Pretraining	IMDb	TREC-6	AG
Without pretraining	5.63	10.67	5.52
With pretraining	**5.00**	**5.69**	**5.38**

TABLE 7.4: Validation error rates for ULMFiT with and without pretraining.

LM	IMDb	TREC-6	AG
Vanilla LM	5.98	7.41	5.76
AWD-LSTM LM	**5.00**	**5.69**	**5.38**

TABLE 7.5: Validation error rates for ULMFiT with a vanilla LM and the AWD-LSTM LM.

LM fine-tuning	IMDb	TREC-6	AG
No LM fine-tuning	6.99	6.38	6.09
Full	5.86	6.54	5.61
Full + discr	5.55	6.36	5.47
Full + discr + stlr	**5.00**	**5.69**	**5.38**

TABLE 7.6: Validation error rates for ULMFiT with different variations of LM fine-tuning.

Impact of pretraining We compare using no pretraining with pretraining on WikiText-103 [Merity et al., 2017b] in Table 7.4. Pretraining is most useful for small and medium-sized datasets, which are most common in commercial applications. However, even for large datasets, pretraining improves performance.

Impact of LM quality In order to gauge the importance of choosing an appropriate LM, we compare a vanilla LM with the same hyperparameters without any dropout[7] with the AWD-LSTM LM with tuned dropout parameters in Table 7.5. Using our fine-tuning techniques, even a regular LM reaches surprisingly good performance on the larger datasets. On the smaller TREC-6, a vanilla LM without dropout runs the risk of overfitting, which decreases performance.

Impact of LM fine-tuning We compare no fine-tuning against fine-tuning the full model [Erhan et al., 2010] ('*Full*'), the most commonly used fine-tuning method, with and without discriminative fine-tuning ('*Discr*') and slanted triangular learning rates ('*Stlr*') in Table 7.6. Fine-tuning the LM is most beneficial for larger datasets. '*Discr*' and '*Stlr*' improve performance across all three datasets and are necessary on the smaller TREC-6, where regular fine-tuning is not beneficial.

[7] To avoid overfitting, we only train the vanilla LM classifier for 5 epochs and keep dropout of 0.4 in the classifier.

Classifier fine-tuning	IMDb	TREC-6	AG
From scratch	9.93	13.36	6.81
Full	6.87	6.86	5.81
Full + discr	5.57	6.21	5.62
Last	6.49	16.09	8.38
Chain-thaw	5.39	6.71	5.90
Freez	6.37	6.86	5.81
Freez + discr	5.39	5.86	6.04
Freez + stlr	5.04	6.02	5.35
Freez + cos	5.70	6.38	**5.29**
Freez + discr + stlr	**5.00**	**5.69**	5.38

TABLE 7.7: Validation error rates for ULMFiT with different methods to fine-tune the classifier.

Impact of classifier fine-tuning We compare training from scratch, fine-tuning the full model ('*Full*'), only fine-tuning the last layer ('*Last*') [Donahue et al., 2014], '*Chain-thaw*' [Felbo et al., 2017], and gradual unfreezing ('*Freez*'). We furthermore assess the importance of discriminative fine-tuning ('*Discr*') and slanted triangular learning rates ('*Stlr*'). We compare the latter to an alternative, aggressive cosine annealing schedule ('*Cos*') [Loshchilov and Hutter, 2017b]. We use a learning rate $\eta^L = 0.01$ for '*Discr*', learning rates of 0.001 and 0.0001 for the last and all other layers respectively for '*Chain-thaw*' as in [Felbo et al., 2017], and a learning rate of 0.001 otherwise. We show the results in Table 7.7.

Fine-tuning the classifier significantly improves over training from scratch, particularly on the small TREC-6. '*Last*', the standard fine-tuning method in CV, severely underfits and is never able to lower the training error to 0. '*Chain-thaw*' achieves competitive performance on the smaller datasets, but is outperformed significantly on the large AG. '*Freez*' provides similar performance as '*Full*'. '*Discr*' consistently boosts the performance of '*Full*' and '*Freez*', except for the large AG. Cosine annealing is competitive with slanted triangular learning rates on large data, but under-performs on smaller datasets. Finally, full ULMFiT classifier fine-tuning (bottom row) achieves the best performance on IMDB and TREC-6 and competitive performance on AG. Importantly, ULMFiT is the only method that shows excellent performance across the board—and is therefore the only *universal* method.

Classifier fine-tuning behavior While our results demonstrate that *how* we fine-tune the classifier makes a significant difference, fine-tuning for inductive transfer is currently under-explored in NLP as it mostly has been thought to be unhelpful [Mou et al., 2016]. To better understand the fine-tuning behavior of our model, we compare

the validation error of the classifier fine-tuned with ULMFiT and '*Full*' during training in Figure 7.4.

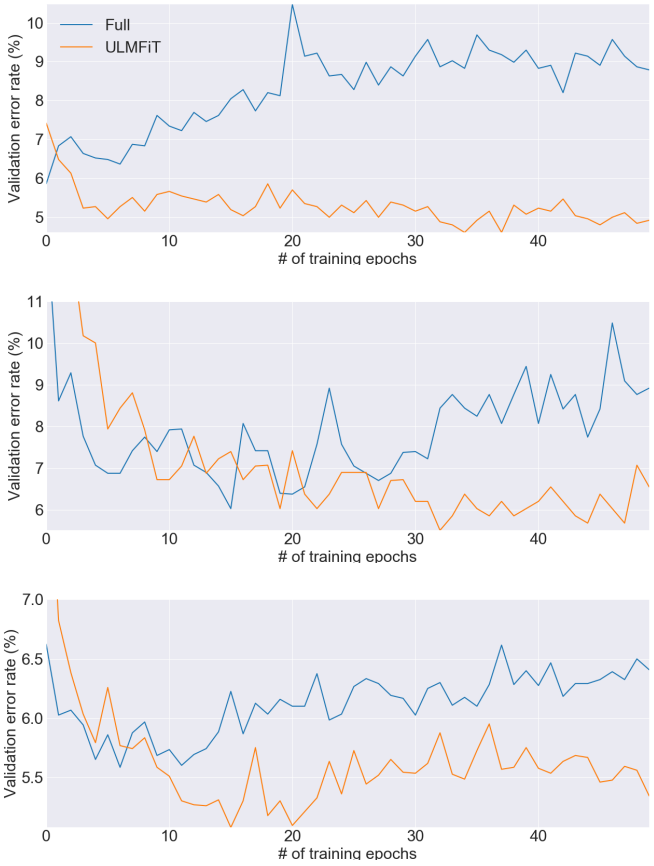


FIGURE 7.4: Validation error rate curves for fine-tuning the classifier with ULMFiT and '*Full*' on IMDb, TREC-6, and AG (top to bottom).

On all datasets, fine-tuning the full model leads to the lowest error comparatively early in training, e.g. already after the first epoch on IMDb. The error then increases as the model starts to overfit and knowledge captured through pretraining is lost. In contrast, ULMFiT is more stable and suffers from no such catastrophic forgetting; performance remains similar or improves until late epochs, which shows the positive effect of the learning rate schedule.

Impact of bidirectionality At the cost of training a second model, ensembling the predictions of a forward and backwards LM-classifier brings a performance boost of around 0.5–0.7. On IMDb we lower the test error from 5.30 of a single model to 4.58 for the bidirectional model.

7.1.6 Summary

We have proposed ULMFiT, an effective and extremely sample-efficient transfer learning method that can be applied to any NLP task. We have also proposed several novel fine-tuning techniques that in conjunction prevent catastrophic forgetting and enable robust learning across a diverse range of tasks. Our method significantly outperformed existing transfer learning techniques and the state-of-the-art on six representative text classification tasks.

Our proposed methods focused on the adaptation phase, which we will study in more detail in the following section. In particular, we will compare the two prevalent adaptation approaches on two state-of-the-art transfer learning methods.

7.2 To Tune or Not to Tune?

* While most previous work has focused on different pretraining objectives and architectures for transfer learning, we ask how to best adapt the pretrained model to a given target task. We focus on the two most common forms of adaptation, feature extraction (where the pretrained weights are frozen), and directly fine-tuning the pretrained model. Our empirical results across diverse NLP tasks with two state-of-the-art models show that the relative performance of fine-tuning vs. feature extraction depends on the similarity of the pretraining and target tasks. We explore possible explanations for this finding and provide a set of adaptation guidelines for the NLP practitioner.

7.2.1 Introduction

Sequential inductive transfer learning [Pan and Yang, 2010] consists of two stages: *pretraining*, in which the model learns a general-purpose representation of inputs, and *adaptation*, in which the representation is transferred to a new task. Most previous work in NLP has focused on different pretraining objectives for learning word or sentence representations [Mikolov et al., 2013a, Kiros et al., 2015]. Few works, however, have focused on the adaptation phase. There are two main paradigms for adaptation: *feature extraction* and *fine-tuning*. In feature extraction (EX) the model's weights are 'frozen' and the pretrained representations are used in a downstream model similar to classic feature-based approaches [Koehn et al., 2003]. Alternatively, a pretrained model's parameters

Peters, M., **Ruder, S.***, and Smith, N. A. (2019). To Tune or Not to Tune? Adapting Pretrained Representations to Diverse Tasks. Matthew ran experiments with ELMo and feature extraction and wrote most of the Experimental Setup section. Sebastian ran experiments with BERT, conducted analyses, and wrote most of the remainder.

Conditions			Guidelines
Pretrain	Adapt.	Task	
Any	EX	Any	Add many task parameters
Any	FT	Any	Add minimal task parameters Hyper-parameters!
Any	Any	Seq. / clas.	EX and FT have similar performance
ELMO	FT	Any	Use ULMFiT techniques
ELMo	Any	Sent. pair	use EX
BERT	Any	Sent. pair	use FT

TABLE 7.8: Guidelines for using feature extraction (EX) and fine-tuning (FT) with ELMo and BERT. Seq.: sequence labeling. Clas.: classification. Sent. pair: sentence pair tasks.

can be unfrozen and fine-tuned (FT) on a new task [Dai and Le, 2015]. Both have benefits: EX enables use of task-specific model architectures and may be computationally cheaper as features only need to be computed once. On the other hand, FT is convenient as it may allow us to adapt a general-purpose representation to many different tasks.

Gaining a better understanding of the adaptation phase is key in making the most use out of pretrained representations. To this end, we compare two state-of-the-art pretrained models, ELMo [Peters et al., 2018a] and BERT [Devlin et al., 2018] using both EX and FT across seven diverse tasks including named entity recognition, natural language inference (NLI), and paraphrase detection. We seek to characterize the conditions under which one approach substantially outperforms the other, and whether it is dependent on the pretraining objective or target task. We find that EX and FT have comparable performance in most cases, except when the source and target tasks are either highly similar or highly dissimilar. We furthermore shed light on the practical challenges of adaptation and provide a set of guidelines to the NLP practitioner, as summarized in Table 7.8.

7.2.2 Pretraining and adaptation

While pretraining tasks have been designed with particular downstream tasks in mind [Felbo et al., 2017], we focus on pretraining tasks that seek to induce *universal* representations suitable for any downstream task (§3.3.2.3).

Sentence embedding methods Such methods learn sentence representations via different pretraining objectives such as previous/next sentence prediction [Kiros et al., 2015, Logeswaran and Lee, 2018], NLI [Conneau et al., 2017], or a combination of objectives [Subramanian et al., 2018]. During the adaptation phase, the sentence

representation is typically provided as input to a linear classifier (EX). LM pretraining with FT has also been successfully applied to sentence level tasks. In the previous section, we propose ULMFiT, techniques for fine-tuning a LM, including triangular learning rate schedules and discriminative fine-tuning, which uses lower learning rates for lower layers. Radford et al. [2018] extend LM-FT to additional sentence and sentence-pair tasks.

Masked LM and next-sentence prediction BERT [Devlin et al., 2018] combines both word and sentence representations (via masked LM and next sentence prediction objectives) in a single very large pretrained transformer [Vaswani et al., 2017]. It is adapted to both word and sentence level tasks by FT with task-specific layers.

7.2.3 Experimental setup

We compare ELMo and BERT as representatives of the two best-performing pretraining settings. This section provides an overview of our methods. For fair comparison, all experiments include extensive hyper-parameter tuning. We tuned the learning rate, dropout ratio, weight decay and number of training epochs. In addition, the fine-tuning experiments also examined the impact of triangular learning rate schedules, gradual unfreezing, and discriminative learning rates presented in Section 7.1. Hyper-parameters were tuned on the development sets and the best setting evaluated on the test sets.

All models were optimized with the Adam optimizer [Kingma and Ba, 2015] with weight decay fix [Loshchilov and Hutter, 2017a]. We used the publicly available pretrained ELMo[1] and BERT[2] models in all experiments. For ELMo, we used the original two layer bidirectional LM. In the case of BERT, we used the BERT-base model, a 12 layer bidirectional transformer. We used the English uncased model for all tasks except for NER which used the English cased model.

7.2.3.1 Target tasks and datasets

We evaluate on a diverse set of target tasks: named entity recognition (NER), sentiment analysis (SA), and three sentence pair tasks, natural language inference (NLI), paraphrase detection (PD), and semantic textual similarity (STS).

NER We use the CoNLL 2003 dataset [Sang and Meulder, 2003], which provides token level annotations of newswire across four different entity types (`PER`, `LOC`, `ORG`, `MISC`).

[1] https://allennlp.org/elmo
[2] https://github.com/google-research/bert

SA We use the binary version of the Stanford Sentiment Treebank [SST-2; Socher et al., 2013], providing sentiment labels (`negative` or `positive`) for phrases and sentences of movie reviews.

NLI We use both the broad-domain MultiNLI dataset [Williams et al., 2018] and Sentences Involving Compositional Knowledge [SICK-E; Marelli et al., 2014].

PD For paraphrase detection (i.e., decide whether two sentences are semantically equivalent), we use the Microsoft Research Paraphrase Corpus [MRPC; Dolan and Brockett, 2005].

STS We employ the Semantic Textual Similarity Benchmark [STS-B; Cer et al., 2017] and SICK-R [Marelli et al., 2014]. Both datasets, provide a human judged similarity value from 1 to 5 for each sentence pair.

We now describe how we adapt ELMo and BERT to these tasks. For EX we require a task-specific architecture, while for FT we need a task-specific output layer.

7.2.3.2 Feature extraction

To isolate the effects of fine-tuning contextual word representations, all feature based models only include one type of word representation (ELMo or BERT) and do not include any other pretrained word representations. For all tasks, all layers of pretrained representations were weighted together with learned scalar parameters following [Peters et al., 2018a].

NER For the NER task, we use a two layer bidirectional LSTM in all experiments. For ELMo, the output layer is a CRF, similar to a state-of-the-art NER system [Lample et al., 2016]. Feature extraction for ELMo treated each sentence independently.

In the case of BERT, the output layer is a softmax to be consistent with the fine-tuned experiments presented in [Devlin et al., 2018]. In addition, as in [Devlin et al., 2018], we used document context to extract word piece representations. When composing multiple word pieces into a single word representation, we found it beneficial to run the BiLSTM layers over all word pieces before taking the LSTM states of the first word piece in each word. We experimented with other pooling operations to combine word pieces into a single word representation but they did not provide additional gains.

SA We used the implementation of the bi-attentive classification network in AllenNLP [Gardner et al., 2017] with default hyper-parameters, except for tuning those noted above. As in the fine-tuning experiments for SST-2, we used all available annotations during training, including those of sub-trees. Evaluation on the development and test sets used full sentences.

Sentence pair tasks When extracting features from ELMo, each sentence was handled separately. For BERT, we extracted features for both sentences jointly to be consistent with the pretraining procedure. As reported in Section 5 this improved performance over extracting features for each sentence separately.

Our model is the ESIM model [Chen et al., 2017a], modified as needed to support regression tasks in addition to classification. We used default hyper-parameters except for those described above.

7.2.3.3 Fine-tuning

When fine-tuning ELMo, we found it beneficial to use discriminative learning rates (§7.1) where the learning rate decreased by 0.4× in each layer. In addition, for SST-2 and NER, we also found it beneficial to gradually unfreeze the weights starting with the top layer. In this setting, in each epoch one additional layer of weights is unfrozen until all weights are training. These settings were chosen by tuning development set performance.

For fine-tuning BERT, we used the default learning rate schedule [Devlin et al., 2018] that is similar to our slanted triangular learning rate schedule (§7.1).

SA We considered several pooling operations for composing the ELMo LSTM states into a vector for prediction including max pooling, average pooling and taking the first/last states. Max pooling performed slightly better than average pooling on the development set.

Sentence pair tasks Our bi-attentive fine-tuning mechanism is similar to the attention mechanism in the feature based ESIM model. To apply it, we first computed the bi-attention between all words in both sentences, then applied the same "enhanced" pooling operation as in [Chen et al., 2017a] before predicting with a softmax. Note that this attention mechanism and pooling operation does not add any additional parameters to the network.

Pretraining	Adaptation	NER CoNLL 2003	SA SST-2	Nat. lang. inference MNLI	SICK-E	Semantic textual similarity SICK-R	MRPC	STS-B
Skip-thoughts	EX	-	81.8	62.9	-	86.6	75.8	71.8
ELMo	EX	91.7	**91.8**	**79.6**	**86.3**	**86.1**	**76.0**	**75.9**
	FT	**91.9**	91.2	76.4	83.3	83.3	74.7	75.5
	Δ=FT-EX	0.2	-0.6	-3.2	-3.3	-2.8	-1.3	-0.4
BERT-base	EX	92.2	93.0	**84.6**	84.8	86.4	78.1	82.9
	FT	**92.4**	**93.5**	**84.6**	**85.8**	**88.7**	**84.8**	**87.1**
	Δ=FT-EX	0.2	0.5	0.0	1.0	2.3	6.7	4.2

TABLE 7.9: Test set performance of feature extraction (EX) and fine-tuning (FT) approaches for ELMo and BERT-base compared to two sentence embedding methods. Settings that are good for FT are colored in red (Δ=FT-EX > 1.0); settings good for EX are colored in blue (Δ=FT-EX < -1.0). Numbers for baseline methods are from respective papers, except for SST-2, MNLI, and STS-B results, which are from Wang et al. [2018a]. BERT fine-tuning results (except on SICK) are from Devlin et al. [2018]. The metric varies across tasks (higher is always better): accuracy for SST-2, SICK-E, and MRPC; matched accuracy for MultiNLI; Pearson correlation for STS-B and SICK-R; and span F_1 for CoNLL 2003. For CoNLL 2003, we report the mean with five seeds; standard deviation is about 0.2%.

7.2.4 Results

We show results in Table 7.9 comparing ELMo and BERT for both EX and FT approaches across the seven tasks with one sentence embedding method, Skip-thoughts [Kiros et al., 2015], that employs a next-sentence prediction objective similar to BERT.

Both ELMo and BERT outperform the sentence embedding method significantly, except on the semantic textual similarity tasks (STS) where Skip-thoughts is similar to ELMo. The overall performance of EX and FT varies from task to task, with small differences except for a few notable cases. For ELMo, we find the largest differences for sentence pair tasks where EX consistently outperforms FT. For BERT, we obtain nearly the opposite result: FT significantly outperforms EX on all STS tasks, with much smaller differences for the others.

Discussion Past work in NLP [Mou et al., 2016] showed that similar pretraining tasks transfer better.[3] In computer vision (CV), Yosinski et al. [2014] similarly found that the transferability of features decreases as the distance between the pretraining and target task increases. In this vein, Skip-thoughts—and Quick-thoughts [Logeswaran and Lee, 2018], which has similar performance—which use a next-sentence prediction objective similar to BERT, perform particularly well on STS tasks, indicating a close alignment between the pretraining and target task. This strong alignment also seems to be the reason for BERT's strong relative performance on these tasks.

[3]Mou et al. [2016], however, only investigate transfer between classification tasks (NLI → SICK-E/MRPC).

	SICK-E	SICK-R	STS-B	MRPC
ELMo-FT +bi-attn.	**83.8**	**84.0**	**80.2**	**77.0**
w/o bi-attn.	70.9	51.8	38.5	72.3

TABLE 7.10: Comparison of ELMo-FT cross-sentence embedding methods on dev. sets of sentence pair tasks.

In CV, FT generally outperforms EX when transferring from ImageNet supervised classification pretraining to other classification tasks [Kornblith et al., 2018]. Recent results suggest FT is less useful for more distant target tasks such as semantic segmentation [He et al., 2018a]. This is in line with our results, which show strong performance with FT between closely aligned tasks (next-sentence prediction in BERT and STS tasks) and poor performance for more distant tasks (LM in ELMo and sentence pair tasks). A confounding factor may be the suitability of the inductive bias of the model architecture for sentence pair tasks, which we will analyze next.

7.2.5 Analyses

Modelling pairwise interactions LSTMs consider each token sequentially, while Transformers can relate each token to every other in each layer [Vaswani et al., 2017]. This might facilitate FT with Transformers on sentence pair tasks, on which ELMo-FT performs comparatively poorly. To analyze this further, we compare different ways of encoding the sentence pair with ELMo and BERT. For ELMo, we compare encoding with and without cross-sentence bi-attention in Table 7.10. When adapting the ELMo LSTM to a sentence pair task, modeling the sentence interactions by fine-tuning through the bi-attention mechanism provides the best performance.[4] This provides further evidence that the LSTM has difficulty modeling the pairwise interactions during sequential processing. This is in contrast to a Transformer LM that can be fine-tuned in this manner [Radford et al., 2018].

For BERT-EX, we compare joint encoding of the sentence pair with encoding the sentences separately in Table 7.11. The latter leads to a drop in performance, which shows that the BERT representations encode cross-sentence relationships and are therefore particularly well-suited for sentence pair tasks.

Impact of additional parameters We evaluate whether adding parameters is useful for both adaptation settings on NER. We add a CRF layer (as used in FT) and a BiLSTM with a CRF layer (as used in EX) to both and show results in Table 7.12. We find

[4]This is similar to text classification tasks, where we find max-pooling to outperform using the final hidden state, similar to [Howard and Ruder, 2018].

	SICK-E	SICK-R	STS-B	MRPC
BERT-EX, joint enc.	**85.5**	86.4	**88.1**	**83.3**
separate encoding	81.2	**86.8**	86.8	81.4

TABLE 7.11: Comparison of BERT-EX cross-sentence embedding methods on dev. sets of sentence pair tasks.

Model configuration	$\mathbf{F}_1$
EX + BiLSTM + CRF	**95.5**
EX + CRF	91.9
FT + CRF + gradual unfreeze	**95.5**
FT + BiLSTM + CRF + gradual unfreeze	95.2
FT + CRF	95.1

TABLE 7.12: Comparison of CoNLL 2003 NER development set performance (F_1) for ELMo for both feature extraction and fine-tuning. All results averaged over five random seeds.

that additional parameters are key for EX, but hurt performance with FT. In addition, FT requires gradual unfreezing [Howard and Ruder, 2018] to match performance of feature extraction.

Extreme gradual unfreezing One hypothesis why additional parameters for FT hurt is that training too many randomly initialized parameters together with fully trained ones complicates the optimization problem. To investigate this hypothesis, we run another experiment with an extreme form of gradual unfreezing: We first use a learning rate of 0 for all layers of the pretrained LM and only train the target task-specific parameters on the downstream. This is the same as EX. The additional parameters should have adapted to the pretrained parameters at this point. We now gradually unfreeze all the layers of the model, in every epoch setting the learning rate of the next lower layer to a non-zero value. We find that the performance of the model decays as soon as more than the top layers are trained jointly, even if learning rates are tuned carefully. The negative impact of additional parameters thus cannot be explained by the discrepancy in initialization alone.

ELMo fine-tuning We found fine-tuning the ELMo LSTM to be initially difficult and required careful hyper-parameter tuning. Once tuned for one task, other tasks have similar hyper-parameters. Our best models used slanted triangular learning rates and discriminative fine-tuning [Howard and Ruder, 2018] and in some cases gradual unfreezing.

	TE	GO	TR	FI	SL
BERT-EX	84.4	86.7	86.1	84.5	80.9
Δ=FT-EX	-1.1	-0.2	-0.6	0.4	-0.6
JS div	0.21	0.18	0.14	0.09	0.09

TABLE 7.13: Accuracy of feature extraction (EX) and fine-tuning (FT) with BERT-base trained on training data of different MNLI domains and evaluated on corresponding dev sets. TE: telephone. FI: fiction. TR: travel. GO: government. SL: slate.

Impact of target domain Pretrained language model representations are intended to be universal. However, the target domain might still impact the adaptation performance. We calculate the Jensen-Shannon divergence based on term distributions (§4.1) between the domains used to train BERT (books and Wikipedia) and each MNLI domain. We show results in Table 7.13. We find no significant correlation. At least for this task, the distance of the source and target domains does not seem to have a major impact on the adaptation performance.

Representations at different layers In addition, we are interested how the information in the different layers of the models develops over the course of fine-tuning. We measure this information in two ways: a) with diagnostic classifiers [Adi et al., 2017]; and b) with mutual information [MI; Noshad et al., 2018]. Both methods allow us to associate the hidden activations of our model with a linguistic property. In both cases, we use the mean of the hidden activations of BERT-base[5] of each token / word piece of the sequence(s) as the representation.[6]

With diagnostic classifiers, for each example, we extract the pretrained and fine-tuned representation at each layer as features. We use these features as input to train a logistic regression model (linear regression for STS-B, which has real-valued outputs) on the training data of two single sentence (CoLA[7] and SST-2) and two pair sentence tasks (MRPC and STS-B). We show its performance on the corresponding dev sets in Figure 7.5.

For all tasks, diagnostic classifier performance generally is higher in higher layers of the model. Fine-tuning improves the performance of the diagnostic classifier at every layer. For the single sentence classification tasks CoLA and SST-2, pretrained performance increases gradually until the last layers. In contrast, for the sentence pair tasks MRPC and STS-B performance is mostly flat after the fourth layer. Relevant information for

[5]We show results for BERT as they are more inspectable due to the model having more layers. Trends for ELMo are similar.

[6]We observed similar results when using max-pooling or the representation of the first token.

[7]The Corpus of Linguistic Acceptability (CoLA) consists of examples of expert English sentence acceptability judgments drawn from 22 books and journal articles on linguistic theory. It uses the Matthews correlation coefficient [Matthews, 1975] for evaluation and is available at: `nyu-mll.github.io/CoLA`

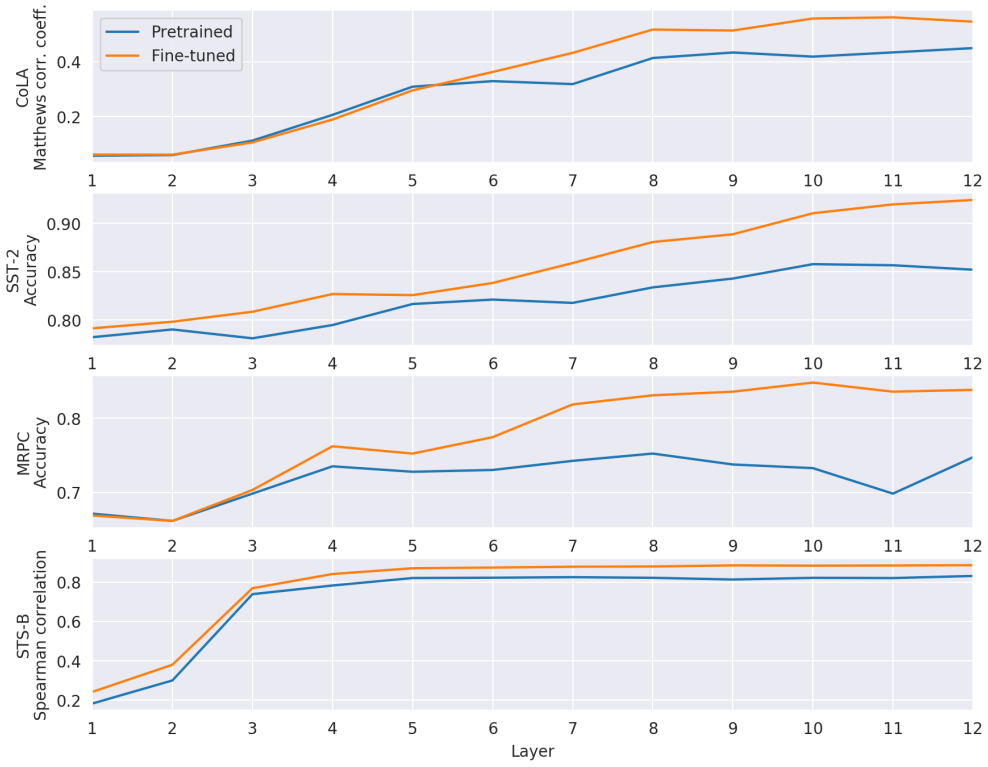


FIGURE 7.5: Performance of diagnostic classifiers trained on pretrained and fine-tuned BERT representations at different layers on the dev sets of the corresponding tasks.

sentence pair tasks thus does not seem to be concentrated primarily in the upper layers of pretrained representations, which could explain why fine-tuning is particularly useful in these scenarios.

Computing the mutual information with regard to representations of deep neural networks has only become feasible recently with the development of more sophisticated MI estimators. In our experiments, we use the state-of-the-art ensemble dependency graph estimator [EDGE; Noshad et al., 2018] with default hyper-parameter values. As a sanity check, we compute the MI between hidden activations and random labels and random representations and random labels, which yields 0 in every case as we would expect.[8]

We show the mutual information $I(H;Y)$ between the pretrained and fine-tuned mean hidden activations H at each layer of BERT and the output labels Y on the dev sets of CoLA, SST-2, and MRPC in Figure 7.6.

The MI between pretrained representations and labels is close to 0 across all tasks and layers, except for SST where the last layer shows a small non-zero value. In contrast, fine-tuned representations display much higher MI values. The MI for fine-tuned representations rises gradually through the intermediate and last layers for the

[8]For the same settings, we obtain non-zero values with earlier estimators [Saxe et al., 2018], which seem to be less reliable for higher numbers of dimensions.

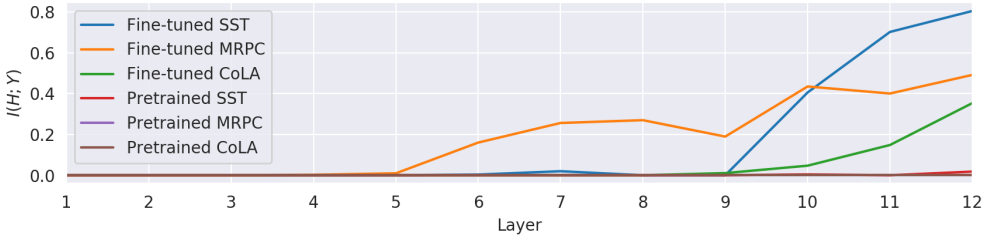


FIGURE 7.6: The mutual information between fine-tuned and pretrained mean BERT representations and the labels on the dev set of the corresponding tasks.

sentence pair task MRPC, while for the single sentence classification tasks, the MI rises sharply in the last layers. Similar to our findings with diagnostic classifiers, knowledge for single sentence classification tasks thus seems mostly concentrated in the last layers, while pair sentence classification tasks gradually build up information in the intermediate and last layers of the model.

7.2.6 Summary

We have empirically analyzed fine-tuning and feature extraction approaches across diverse datasets, finding that the relative performance depends on the similarity of the pretraining and target tasks. We have explored possible explanations and provided practical recommendations for adapting pretrained representations to NLP practitioners. We have additionally analyzed the representations at different layers using diagnostic classifiers and mutual information.

7.3 Conclusions

In this chapter, we have analyzed and proposed a novel framework for sequential transfer learning, focusing on the adaptation phase. Our ULMFiT framework outperformed the state-of-the-art across six diverse text classification tasks, while we have empirically studied adaptation capabilities of two state-of-the-art models across a range of classification and sequence labelling tasks.

In the following and final chapter of this thesis, we will look back on the methods proposed in this dissertation, highlight insights and commonalities, and provide an outlook into the future.

Chapter 8

Conclusion

Throughout this thesis, we have made contributions to the four areas of transfer learning in natural language processing: domain adaptation (§4), cross-lingual learning (§5), multi-task learning (§6), and sequential transfer learning (§7). In this final chapter, we will recapitulate the proposed methods (§8.1), summarize our findings (§8.2), and provide an outlook into the future (§8.3).

8.1 Synopsis

In this dissertation, we have studied the problem of automatically learning transferable representations for a variety of NLP tasks. In Chapter 3, we have presented a taxonomy for transfer learning for NLP and provided a comprehensive overview of each area.

Chapter 4 presented approaches for domain adaptation that overcome the discrepancy between domains by automatically selecting relevant and informative examples. In particular, it presented a novel automatic data selection method and a novel semi-supervised learning approach that combines the strengths of both classic and neural techniques. We evaluated our methods across three diverse NLP tasks and multiple domains.

Chapter 5 focused on cross-lingual learning, particularly unsupervised and weakly supervised approaches. We first extensively analyzed the limitations of unsupervised approaches and proposed a weakly supervised method that mitigates them. We then proposed a different latent-variable view of existing neural methods and a novel cross-lingual word embedding model that combines both traditional and current approaches. We evaluated our methods on bilingual dictionary induction where they outperform the state of the art.

In Chapter 6, we studied the problem of multi-task learning across heterogeneous tasks. We proposed two novel models that enable more flexible sharing than existing architectures. We evaluated our methods on diverse sets of tasks and domains where they outperformed hard parameter sharing and strong single-task models.

Chapter 7 tackled the most impactful sequential transfer learning scenario involving transfer to an arbitrary target task. We proposed ULMFiT, a novel framework that pretrains language model representations and then transfers them to a target task using novel techniques. Our method achieved significant error reductions over the state-of-the-art across a range of text classification datasets. We additionally compared feature extraction and fine-tuning of two state-of-the-art pretrained representations across a diverse set of tasks. We found that the similarity between pretraining and target task is indicative of the relative performance between feature extraction and fine-tuning. We finally provided practical guidelines to practitioners.

8.2 Summary of findings

This dissertation investigated the hypothesis that deep neural networks that leverage relevant information using transfer learning techniques outperform their supervised counterparts across a wide range of tasks. Over the course of this thesis, we have presented multiple novel methods for different transfer learning scenarios and evaluated them across a diversity of settings where they outperformed single-task learning as well as competing transfer learning methods. We now recapitulate how our methods addressed the desiderata we laid out initially and summarize our contributions and findings.

Research objectives

Overcoming a discrepancy between source and target setting Transferring between dissimilar or distant settings is one of the biggest challenges in transfer learning and a key reason for the failure of current systems. Domain adaptation methods deteriorate when domains are dissimilar; cross-lingual embedding methods fail when languages are distant; and multi-task learning approaches break down when tasks are unrelated. We have proposed methods that seek to address this challenge in every setting: For domain adaptation, we have proposed a method that takes into account the relevancy and informativeness of examples using Bayesian Optimization (§4.1). When learning cross-lingual word embeddings between distant languages such as English and Finnish, we characterized this distance with a novel eigenvector similarity metric and found weak supervision in the form of identically spelled strings to provide an effective bridge (§5).

Our proposed architectures for multi-task learning aim to be as flexible as possible so that beneficial information can be shared, but unrelated tasks do not result in worse performance (§6). In particular, using subspaces and allowing for sharing across the hierarchy proved beneficial. Our new framework for sequential transfer learning learns general-purpose representations that can be transferred to any target task (§7.1). In this framework, we proposed several techniques that facilitate transfer to a different target task. We found these techniques key in adapting LSTM-based representations and observed that task similarity plays a role in adaptation. When tasks are dissimilar, feature extraction is preferred, while fine-tuning works better for related tasks (§7.2).

Inducing an inductive bias Much of the work in this thesis has focused on leveraging an inductive bias that narrows a model's hypothesis space and improves its ability to generalize. To this end, we employed a wide array of inductive biases. We used a model's or auxiliary models' predictions on unlabelled data as additional learning signal via semi-supervised learning (§4.2, §6.2). Multi-task learning similarly was a pervasive inductive bias that we employed across many models (§4.2, §6). Another common bias that we found helpful is an orthogonality constraint, which allowed us to encourage multi-task learning models to focus on learning different features (§4.2, §6.1). For cross-lingual learning, we found weak supervision in the form of identically spelled strings to be a useful inductive bias (§5). As another inductive bias for cross-lingual learning, we employed a 1:1 matching prior that enabled us to learn a better matching between words and their translations and mitigate the hubness problem (§5.2). In our multi-task learning model, we found a hierarchical bias in the form of a latent mixture to be helpful (§6.1). Finally, pretrained language model representations allowed us to achieve state-of-the-art performance across a wide range of tasks (§7).

Combining traditional and current approaches In this thesis, we have extensively reviewed the existing literature (§3). At the same time, many current approaches reinvent the wheel, replacing classic feature-based approaches with a different toolkit rather than seeking inspiration and drawing on lessons of the past. While classic approaches were an inspiration for this thesis in general, in two cases we explicitly combine them with current methods. For semi-supervised learning, we adapt classic bootstrapping approaches to neural networks and propose a novel method that combines both the agreement-based learning of tri-training and the efficiency of neural multi-task learning (§4.2). For cross-lingual learning, we combine the theoretically motivated bipartite matching dictionary prior of Haghighi et al. [2008] with a state-of-the-art embedding-based approach (§5.2).

Transfer across the hierarchy of NLP tasks Tasks in natural language processing are naturally hierarchical and range from tasks that require morphological information, to syntactic and semantic tasks. This hierarchy offers challenges, as it complicates sharing of layers, but also brings opportunities, if information related to different levels of meaning can be organized and accessed reliably. Our multi-task meta-architecture was designed with this hierarchy in mind. It enables sharing across all levels of the hierarchy, from low-level to high-level tasks (§6.1). Our second multi-task architecture focuses on the highest layer, the label embedding. It leverages new mechanisms to better integrate information across related label spaces to transfer between coarse-grained and fine-grained sentiment tasks (§6.2). Finally, ULMFiT introduces new techniques that seek to minimize catastrophic forgetting during adaptation. It employs gradual unfreezing and discriminative fine-tuning to retain the general information in the lowest layers of the model and transfer these to the target task (§7.1).

Generalization across many settings We were interested in methods that do not depend on any particular task and in models that generalize across a wide range of settings. We thus evaluated our models on a wide range of tasks including POS tagging, dependency parsing, and sentiment analysis (§4); chunking, NER, semantic role labelling, and POS tagging (§6.1); pairwise classification tasks (6.2); text classification tasks (§7.1); and seven diverse NLP tasks (§7.2). In general, we were interested in models that not only perform well on in-domain data but also achieve good performance on data outside of the training distribution. To this end, we evaluated our models on product review, movie review, and Twitter domains for sentiment analysis (§4, §6.2, §7.1), web domains for sequence labelling tasks (§4, §6.1), and multiple domains for topic classification (§7.1). We finally evaluated our approaches on a diverse set of languages, including Estonian, Finnish, Greek, Hungarian, Polish, and Turkish (§5.1) as well as English, German, Italian, Finnish, Turkish, Bengali, and Hindi (§5.2).

Contributions

We now summarize this thesis' contributions with regard to transfer learning overall and each transfer learning area in particular:

Transfer learning We presented a taxonomy for transfer learning for NLP (§3.1.3) by adapting the widely used taxonomy of Pan and Yang [2010]. The taxonomy reflects the most common transfer learning scenarios encountered in current natural language processing. We extensively reviewed each of the scenarios (§3.2, §3.3, §3.4, §3.5) and highlighted connections and similarities between the approaches of different areas. For

instance, we made it clear where sequential transfer learning and domain adaptation approaches build on multi-task learning techniques.

Domain adaptation We have presented a model that automatically learns to select training examples that are relevant for a particular target domain (§4.1). We have evaluated this model across three tasks where it outperformed existing domain similarity metrics and demonstrated the transferability of the learned policy across different settings. We additionally re-evaluated a range of classic self-labelling approaches to neural networks in the context of a domain shift (§4.2). We found that tri-training works best and even outperforms a recent state-of-the-art method. We addressed the drawback of tri-training, its time and space complexity, with a more efficient multi-task tri-training method. Our novel method outperformed both traditional tri-training and recent alternatives on sentiment analysis. For POS tagging, classic tri-training still was superior, particularly on OOV and low frequency tokens, which suggests it is less affected by error propagation.

Cross-lingual learning We presented a taxonomy for cross-lingual word embedding models that organizes models based on the main sources of variation, the type of alignment and the comparability of the training data (§3.5.3). We additionally showed that cross-lingual word embedding models that learn on the word level optimize similar objectives by reducing each model to a combination of monolingual losses and regularization terms (§3.5.4). We theoretically and empirically analyzed the limitations of unsupervised cross-lingual embedding models and showed that unsupervised methods perform much worse on distant language pairs, with monolingual corpora from different domains, and different embedding algorithms (§5.1). We proposed a novel eigenvector-based metric to characterize the distance between monolingual embedding spaces and gauge the potential of unsupervised methods. We furthermore presented a novel latent-variable model for bilingual lexicon induction that combines a bipartite matching prior with a state-of-the-art embedding-based approach, which we evaluated on three standard and three low-resource language pairs where it outperformed the state-of-the-art (§5.2). We finally showed how existing approaches can be viewed as a similar latent variable model with a different prior.

Multi-task learning We presented sluice networks, a novel multi-task learning framework that automatically learns which layers to share between different tasks and detailed how this model generalizes existing transfer learning and multi-task learning architectures (§6.1). We evaluated the model on synthetic data and data from OntoNotes 5.0 covering multiple tasks and domains where it consistently outperformed single-task and multi-task baselines. We additionally proposed a novel multi-task learning model that integrates

information from disparate label spaces and propose methods to leverage unlabelled and auxiliary data (§6.2). We evaluated the model on a variety of pairwise sequence classification tasks where it outperformed single and multi-task baselines and achieved a new state of the art for aspect- and topic-based sentiment analysis.

Sequential transfer learning We proposed a novel framework for sequential transfer learning using pretrained language models and novel adaptation techniques including novel ways to fine-tune a language model (§7.1). Our method convincingly outperformed the state-of-the-art on six text classification tasks with an error reduction of 18–24% on most tasks and demonstrated significant few-shot learning capabilities. In addition, we analyzed the adaptation phase of sequential transfer learning comparing the two prevalent adaptation methods with state-of-the-art pretrained representations on a diverse range of tasks (§7.2). We found that the relative performance of fine-tuning vs. feature extraction depends on the similarity of the pretraining and target tasks. In addition, we found that information in pretrained BERT [Devlin et al., 2018] representations on sentence pair tasks is mostly flat across intermediate and higher layers, while information for single sentence tasks gradually increases throughout the network, which may explain why fine-tuning is particularly useful for the former set of tasks. We finally provided guidelines to the NLP practitioner, e.g. ELMo [Peters et al., 2018a] should extract features by default, while BERT should be fine-tuned.

8.3 Future directions

In this section, we will give an outlook into the future for each of the respective transfer learning areas in particular and for transfer learning in general.

Domain adaptation

Robustness to out-of-distribution data As we become more cognizant of the brittleness of our current methods, we hope that more work will focus on making them more robust. To test robustness, evaluation on out-of-domain data will become more common. The recent CoQA [Reddy et al., 2018] dataset already includes out-of-domain evaluation. We hope that in the future when new datasets are created, in addition to a random test set, dedicated out-of-domain test sets will become common practice. This will enable us to test whether our models truly generalize.

Challenge sets A related direction is to evaluate models on challenge sets, which have recently become more popular [Linzen et al., 2016, Gulordava et al., 2018]; see [Belinkov and Glass, 2019] for an overview. Such challenge sets probe particular aspects such as certain linguistic phenomena on which current models fail. Challenge sets have been mainly created for tasks such as question answering and machine translation and English. In the future, we expect the creation of challenge sets for other tasks and non-English languages.

Adversarial examples One way to induce robustness is via adversarial examples [Goodfellow et al., 2015, Wang and Bansal, 2018]. Adversarial examples are an ongoing research topic and are closely related to understanding the generalization properties of current models. Adversarial examples have become more common in NLP, but generating useful adversarial examples that work across tasks and domains is still a challenge.

Semi-supervised learning We also expect more effective approaches that combine explicit semi-supervised learning such as self-labelling (§4.2) with transfer learning. While transfer learning allows us to leverage unlabelled data via an unsupervised pretraining or auxiliary task, semi-supervised learning uses unlabelled data directly for our target task, for instance by making the model's predictions more consistent with itself. Current semi-supervised learning methods, however, still struggle with a domain shift (§3.4.4.2) and need to be made more robust to be used at their full potential and as a complement to transfer learning methods.

Cross-lingual learning

Cross-lingual resources In order to bridge the digital language divide, NLP models must be applied to non-English languages. To this end, resources will need to be created not just for English but for the world's other 6,000 languages. This can be done comparatively inexpensively in many cases by releasing small test datasets in multiple languages that can be used to evaluate cross-lingual models. A recent example of such a data set is XNLI [Conneau et al., 2018b], which contains NLI test sets for 15 languages.

Unsupervised cross-lingual methods As training data is not available for most of these languages, unsupervised and weakly supervised cross-lingual methods will be required to deal with such low-resource scenarios (§5). As we have shown, current unsupervised methods are still unstable in many settings, such as in the context of different domains, dissimilar languages, or different algorithms. Developing more robust unsupervised methods is thus an important research direction.

Deep cross-lingual embedding models Similar to the progression from shallow word embeddings to deep language models, we expect training of deep cross-lingual models to be a fruitful research direction. The main challenge here is to enable models to go beyond learning from sentence-level parallel data and incorporate non-parallel monolingual corpora. To this end, unsupervised and weakly-supervised techniques will also be useful. A shared representation at the input layers such as cross-lingual word embeddings or a shared word piece vocabulary [Lample et al., 2018b] may often be sufficient to kick-start the training of a deep model.

Multi-task learning

Massively multi-task models Hard parameter sharing will likely still be used as the default when learning between two tasks as the method is simple and efficient. However, models that learn from more tasks will likely move away from this paradigm as it will be too restrictive. In particular, we expect the development of 'massively multi-task models' that are heavily modularized and update only a comparatively small number of parameters for each task. Such architectures will be more flexible and allow for variable degrees of sharing (§6). A challenge will be to allow for such flexibility without the number of parameters blowing up.

Convergence of multi-task learning and sequential transfer learning Given a larger number of tasks, sequential transfer learning and multi-task learning will likely converge, as it will often be prohibitive to train all tasks jointly. Instead, many tasks will be pretrained and optionally equipped with regularizers to encourage positive backwards transfer. A related direction is the development of scalable lifelong learning approaches that are not memory-based and scale sublinearly with the number of tasks.

Understanding task relationships Finally, we hope for more efforts focusing on understanding the relationships between tasks. So far, most of the insight in NLP has come from anecdotal evidence and small-scale empirical studies that investigated the pairwise relationships between a few tasks (§3.2.6). A large-scale study such as the one done by Zamir et al. [2018] in computer vision is necessary to obtain more generally useful insights. A confounding factor is the role of the domain, which is harder to isolate in NLP. As tasks are diverse, ranging from natural language generation, to sequence tagging, natural language understanding, and text classification, it is harder to obtain annotations for each instance across many tasks.

Sequential transfer learning

Specialized pretraining tasks We expect deep pretrained representations to become a common tool for the NLP practitioner, in their utility similar to pretrained word embeddings in the past years. As transfer learning becomes more pervasive, we expect the emergence of dedicated pretraining tasks that encode specialized knowledge. This is already evidenced in approaches that report gains by combining different pretrained word embeddings [Kiela et al., 2018b]. Eventually, downstream models will be able to choose from a large range of pretrained representations. Some of these will be specialized to particular domains such as biomedical or legal documents, while others will be specialized to specific tasks such as particular forms of reasoning and natural language understanding, e.g. anaphora resolution, arithmetic reasoning, etc. By sparsely selecting among a set of relevant modules, approaches will both be more computationally efficient and use richer representations compared to classic NLP pipelines.

Adaptation We similarly expect more work focusing on effective adaptation strategies that allow us to go from the pretrained initialization to the final trained model more effectively. When adapting to a few number of examples, meta-learning might be a viable method, though it still requires access to labelled data from a diverse distribution of tasks for training the meta-learner. As pretrained models get deeper, we will see best practices emerge regarding the adaptation of many layers that leverage different learning rates or freeze part of the model, such as the techniques we proposed (§7). In addition, methods that flexibly augment pretrained parameters with task-specific ones—similar to residual adapters [Rebuffi et al., 2017]—and that extend these connections to multiple tasks will become more commonplace.

Transfer learning

On a broader note, in the long-term we expect transfer learning to be an integral part of NLP systems. It will become the exception that models are trained from scratch; particularly, as we tackle more challenging problems such as applications that require common sense reasoning, we will need to develop components that can acquire and integrate world knowledge from disparate sources into our models. Transfer learning will play an important role to bridge the digital language divide and to bring NLP capabilities to the rest of the world's languages. As NLP systems become more mature and are applied to many more real-world—and often low-resource—scenarios, leveraging information from related domains, tasks, and languages will be key.

Language is a reflection of our context and experience. Training from a blank slate deprives our models of experience and the ability to interpret context. Ultimately, in order to come closer to the elusive goal of true natural language understanding, we need to equip our models with as much relevant knowledge and experience as possible.

Bibliography

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., et al. (2016). Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467*.

Abney, S. (2007). *Semisupervised learning for computational linguistics*. CRC Press.

Abu-Mostafa, Y. S. (1990). Learning from hints in neural networks. *Journal of Complexity*, 6(2):192–198.

Adams, O., Makarucha, A., Neubig, G., Bird, S., and Cohn, T. (2017). Cross-lingual word embeddings for low-resource language modeling. In *Proceedings of EACL*, pages 937–947.

Adams, R. P. and Zemel, R. S. (2011). Ranking via Sinkhorn propagation. *arXiv preprint arXiv:1106.1925*.

Adi, Y., Kermany, E., Belinkov, Y., Lavi, O., and Goldberg, Y. (2017). Fine-grained Analysis of Sentence Embeddings Using Auxiliary Prediction Tasks. In *Proceedings of ICLR 2017*.

Al-Rfou, R., Perozzi, B., and Skiena, S. (2013). Polyglot: Distributed word representations for multilingual NLP. In *Proceedings of CoNLL*, pages 183–192.

Alam, F., Joty, S., and Imran, M. (2018). Domain Adaptation with Adversarial Training and Graph Embeddings. In *Proceedings of ACL 2018*.

Ali, S. M. and Silvey, S. D. (1966). A general class of coefficients of divergence of one distribution from another. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 131–142.

Allenby, G. M. and Rossi, P. E. (1998). Marketing models of consumer heterogeneity. *Journal of econometrics*, 89(1-2):57–78.

Alonso, H. M. and Plank, B. (2017). When is multitask learning effective? Multitask learning for semantic sequence prediction under varying data conditions. In *EACL*.

Ammar, H. B., Eaton, E., Cis, E., Edu, U., Ruvolo, P., College, O., Ruvolo, P., Edu, O., Taylor, M. E., Eecs, T., and Edu, W. S. U. (2014). Online Multi-Task Learning for Policy Gradient Methods. In *Proceedings of ICML 2014*.

Ammar, W., Mulcaire, G., Ballesteros, M., Dyer, C., and Smith, N. A. (2016a). One Parser, Many Languages. *Transactions of the Association for Computational Linguistics*, 4:431–444.

Ammar, W., Mulcaire, G., Tsvetkov, Y., Lample, G., Dyer, C., and Smith, N. A. (2016b). Massively Multilingual Word Embeddings. *CoRR*, abs/1602.01925.

Ando, R. K. (2004). Exploiting unannotated corpora for tagging and chunking. In *Proceedings of the ACL 2004 on Interactive poster and demonstration sessions*, page 13. Association for Computational Linguistics.

Ando, R. K. and Zhang, T. (2005a). A Framework for Learning Predictive Structures from Multiple Tasks and Unlabeled Data. *Journal of Machine Learning Research*, 6:1817–1853.

Ando, R. K. and Zhang, T. (2005b). A high-performance semi-supervised learning method for text chunking. In *Proceedings of the 43rd annual meeting on association for computational linguistics*, pages 1–9. Association for Computational Linguistics.

Andrychowicz, M., Denil, M., Gomez, S., Hoffman, M. W., Pfau, D., Schaul, T., and de Freitas, N. (2016). Learning To Learn by Gradient Descent by Gradient Descent. In *Advances in Neural Information Processing Systems 29 (NIPS 2016)*, number Nips, pages 1–16.

Argyriou, A. and Pontil, M. (2007). Multi-Task Feature Learning. In *Advances in Neural Information Processing Systems*.

Arik, S. O., Chrzanowski, M., Coates, A., Diamos, G., Gibiansky, A., Kang, Y., Li, X., Miller, J., Raiman, J., Sengupta, S., et al. (2017). Deep voice: Real-time neural text-to-speech. *arXiv preprint arXiv:1702.07825*.

Arjovsky, M., Chintala, S., and Léon B (2017). Wasserstein Generative Adversarial Networks. In *Proceedings of ICML 2017*.

Arnold, A., Nallapati, R., and Cohen, W. W. (2007). A comparative study of methods for transductive transfer learning. In *ICDM Workshops*, pages 77–82.

Arora, S., Liang, Y., and Tengyu Ma (2017). A Simple But Tough-to-Beat Baseline for Sentence Embeddings. In *ICLR 2017*.

Artetxe, M., Labaka, G., and Agirre, E. (2016). Learning principled bilingual mappings of word embeddings while preserving monolingual invariance. *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP-16)*, pages 2289–2294.

Artetxe, M., Labaka, G., and Agirre, E. (2017). Learning bilingual word embeddings with (almost) no bilingual data. In *Proceedings of ACL*, pages 451–462.

Artetxe, M., Labaka, G., and Agirre, E. (2018a). Generalizing and Improving Bilingual Word Embedding Mappings with a Multi-Step Framework of Linear Transformations. In *Proceedings of AAAI 2018*.

Artetxe, M., Labaka, G., Agirre, E., and Cho, K. (2018b). Unsupervised neural machine translation. In *Proceedings of ICLR (Conference Track)*.

Aue, A. and Gamon, M. (2005). Customizing sentiment classifiers to new domains: A case study. In *Proceedings of recent advances in natural language processing (RANLP)*, volume 1, pages 2–1. Citeseer.

Augenstein, I., Rocktäschel, T., Vlachos, A., and Bontcheva, K. (2016). Twitter Stance Detection with Bidirectional Conditional Encoding. In *Proceedings of EMNLP*.

Augenstein, I., Ruder, S., and Søgaard, A. (2018). Multi-task Learning of Pairwise Sequence Classification Tasks Over Disparate Label Spaces. In *Proceedings of NAACL-HLT 2018*.

Augenstein, I. and Søgaard, A. (2017). Multi-task learning of keyphrase boundary detection. In *Proceedings of ACL*.

Axelrod, A., He, X., and Gao, J. (2011). Domain adaptation via pseudo in-domain data selection. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*. ACM.

Ba, J., Hinton, G., Mnih, V., Leibo, J. Z., and Ionescu, C. (2016). Using Fast Weights to Attend to the Recent Past. In *Advances in Neural Information Processing Systems 29 (NIPS 2016)*, pages 1–9.

Bahdanau, D., Cho, K., and Bengio, Y. (2015). Neural Machine Translation by Jointly Learning to Align and Translate. In *ICLR 2015*.

Baker, C. F., Fillmore, C. J., and Lowe, J. B. (1998). The berkeley framenet project. In *Proceedings of the 17th international conference on Computational linguistics-Volume 1*, pages 86–90. Association for Computational Linguistics.

Bakker, B. and Heskes, T. (2003). Task Clustering and Gating for BayesianMultitask Learning. *Journal of Machine Learning Research*, 1(1):83–99.

Balikas, G. and Amini, M.-R. (2016). TwiSE at SemEval-2016 Task 4: Twitter Sentiment Classification. In *Proceedings of SemEval*.

Balikas, G. and Moura, S. (2017). Multitask Learning for Fine-Grained Twitter Sentiment Analysis. In *International ACM SIGIR Conference on Research and Development in Information Retrieval 2017*.

Banko, M. and Etzioni, O. (2007). Strategies for Lifelong Knowledge Extraction from the Web. In *K-CAP '07*.

Barone, A. V. M. (2016). Towards cross-lingual distributed representations without parallel text trained with adversarial autoencoders. *Proceedings of the 1st Workshop on Representation Learning for NLP*, pages 121–126.

Baroni, M., Dinu, G., and Kruszewski, G. (2014). Don't count, predict! A systematic comparison of context-counting vs. context-predicting semantic vectors. In *Proceedings of ACL 2014*, pages 238–247.

Baxter, J. (1997). A Bayesian/information theoretic model of learning to learn via multiple task sampling. *Machine Learning*, 28:7–39.

Baxter, J. (2000). A Model of Inductive Bias Learning. *Journal of Artificial Intelligence Research*, 12:149–198.

Belinkov, Y. and Bisk, Y. (2018). Synthetic and Natural Noise Both Break Neural Machine Translation. In *Proceedings of ICLR 2018*.

Belinkov, Y. and Glass, J. (2019). Analysis Methods in Neural Language Processing: A Survey. *Transactions of the ACL*.

Ben-David, S., Blitzer, J., Crammer, K., and Pereira, F. (2007). Analysis of representations for domain adaptation. *Advances in Neural Information Processing Systems*, 19:137–144.

Ben-David, S. and Schuller, R. (2003). Exploiting task relatedness for multiple task learning. *Learning Theory and Kernel Machines*, pages 567–580.

Bengio, Y., Louradour, J., Collobert, R., and Weston, J. (2009). Curriculum learning. In *International Conference on Machine Learning, ICML*.

Benton, A., Mitchell, M., and Hovy, D. (2017). Multi-Task Learning for Mental Health using Social Media Text. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*.

Bergsma, S. and Van Durme, B. (2011). Learning bilingual lexicons using the visual similarity of labeled Web images. In *Proceedings of IJCAI*, pages 1764–1769.

Bertsimas, D. and Tsitsiklis, J. (1997). *Introduction to linear optimization*. Athena Scientific.

Bhattacharya, A. (1943). On a measure of divergence between two statistical population defined by their population distributions. *Bulletin Calcutta Mathematical Society 35.99-109 (1943): 28.*, 35(99-109):28.

Bickel, S., Brückner, M., and Scheffer, T. (2007). Discriminative learning for differing training and test distributions. In *Proceedings of the 24th international conference on Machine learning*, pages 81–88. ACM.

Bingel, J. and Søgaard, A. (2017). Identifying beneficial task relations for multi-task learning in deep neural networks. In *EACL*.

Binkowski, M., Sutherland, D. J., Arbel, M., and Gretton, A. (2018). Demystifying MMD GANs. In *Proceedings of ICLR 2018*.

Bjerva, J. (2017). Will my auxiliary tagging task help? Estimating Auxiliary Tasks Effectivity in Multi-Task Learning. *Proceedings of the 21st Nordic Conference of Computational Linguistics*, (May):216–220.

Blei, D. M., Ng, A. Y., and Jordan, M. I. (2003). Latent Dirichlet Allocation. *Journal of Machine Learning Research*, 3(4-5):993–1022.

Blitzer, J., Crammer, K., Kulesza, A., Pereira, F., and Wortman, J. (2008). Learning bounds for domain adaptation. In *Advances in neural information processing systems*, pages 129–136.

Blitzer, J., Dredze, M., and Pereira, F. (2007). Biographies, bollywood, boom-boxes and blenders: Domain adaptation for sentiment classification. *Annual Meeting-Association for Computational Linguistics*, 45(1):440.

Blitzer, J., Foster, D. P., and Kakade, S. M. (2009). Zero-shot domain adaptation: A multi-view approach. *Tech. Rep. TTI-TR-2009-1*.

Blitzer, J., McDonald, R., and Pereira, F. (2006). Domain Adaptation with Structural Correspondence Learning. *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing (EMNLP '06)*, pages 120–128.

Blum, A. and Mitchell, T. (1998). Combining labeled and unlabeled data with co-training. *Proceedings of the eleventh annual conference on Computational learning theory - COLT' 98*, pages 92–100.

Bojanowski, P., Grave, E., Joulin, A., and Mikolov, T. (2017). Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:125–136.

Bollegala, D., Weir, D., and Carroll, J. (2011). Using Multiple Sources to Construct a Sentiment Sensitive Thesaurus for Cross-Domain Sentiment Classification. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 132–141.

Bollman, M., Bingel, J., and Søgaard, A. (2017). Learning attention for historical text normalization by learning to pronounce. In *Proceedings of ACL*.

Borgwardt, K. M., Gretton, A., Rasch, M. J., Kriegel, H.-P., Schölkopf, B., and Smola, A. J. (2006). Integrating structured biological data by kernel maximum mean discrepancy. *Bioinformatics*, 22(14):e49–e57.

Bousmalis, K., Trigeorgis, G., Silberman, N., Krishnan, D., and Erhan, D. (2016). Domain Separation Networks. *NIPS*.

Braud, C., Lacroix, O., and Søgaard, A. (2017). Cross-lingual and cross-domain discourse segmentation of entire documents. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*.

Brochu, E., Cora, V. M., and De Freitas, N. (2010). A tutorial on Bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning. In *CoRR*.

Brown, H. D. and Lee, H. (1994). *Teaching by principles: An interactive approach to language pedagogy*, volume 1. Prentice Hall Regents Englewood Cliffs, NJ.

Brown, P. F., DeSouza, P. V., Mercer, R. L., Della Pietra, V. J., and Lai, J. C. (1992). Class-Based n-gram Models of Natural Language. *Association for Computational Linguistics*, (1950):467–479.

Brown, P. F., Pietra, S. D., Pietra, V. J. D., and Mercer, R. L. (1993a). The mathematics of statistical machine translation: Parameter estimation. *Computational Linguistics*, 19(2):263–311.

Brown, P. F., Pietra, V. J. D., Pietra, S. A. D., and Mercer, R. L. (1993b). The mathematics of statistical machine translation: Parameter estimation. *Computational Linguistics*, 19(2):263–311.

Brown, P. F., Pietra, V. J. D., Pietra, S. A. D., and Mercer, R. L. (1993c). The mathematics of statistical machine translation: Parameter estimation. *Computational linguistics*, 19(2):263–311.

Brun, C., Perez, J., and Roux, C. (2016). XRCE at SemEval-2016 Task 5: Feedbacked Ensemble Modelling on Syntactico-Semantic Knowledge for Aspect Based Sentiment Analysis. *Proceedings of SemEval*.

Bruner, J. (1985). Child's talk: Learning to use language. *Child Language Teaching and Therapy*, 1(1):111–114.

Calixto, I., Liu, Q., and Campbell, N. (2017). Multilingual Multi-modal Embeddings for Natural Language Processing. *CoRR*, abs/1702.01101.

Camacho-Collados, J., Pilehvar, M. T., Collier, N., and Navigli, R. (2017). SemEval-2017 Task 2: Multilingual and cross-lingual semantic word similarity. In *Proceedings of SEMEVAL*, pages 15–26.

Camacho-Collados, J., Pilehvar, M. T., and Navigli, R. (2015). A Framework for the Construction of Monolingual and Cross-lingual Word Similarity Datasets. *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Short Papers)*, (April 2016).

Caragea, C., McNeese, N., Jaiswal, A., Traylor, G., Kim, H.-W., Mitra, P., Wu, D., Tapia, A. H., Giles, L., Jansen, B. J., et al. (2011). Classifying text messages for the haiti earthquake. In *Proceedings of the 8th international conference on information systems for crisis response and management (ISCRAM2011)*. Citeseer.

Carey, S. and Bartlett, E. (1978). Acquiring a single new word.

Carlson, A., Betteridge, J., Kisiel, B., Settles, B., Jr., E. R. H., and Mitchell, T. M. (2010). Toward an Architecture for Never-Ending Language Learning. In *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence (AAAI-10) Toward*.

Carreira, J. and Zisserman, A. (2017). Quo vadis, action recognition? a new model and the kinetics dataset. In *Computer Vision and Pattern Recognition (CVPR), 2017 IEEE Conference on*, pages 4724–4733. IEEE.

Caruana, R. (1993). Multitask learning: A knowledge-based source of inductive bias. In *Proceedings of the Tenth International Conference on Machine Learning*.

Caruana, R. (1998). Multitask Learning. *Autonomous Agents and Multi-Agent Systems*, 27(1):95–133.

Caruana, R. and de Sa, V. R. (1997). Promoting poor features to supervisors: Some inputs work better as outputs. *Advances in Neural Information Processing Systems 9: Proceedings of The 1996 Conference*, 9:389.

Cavallanti, G., Cesa-Bianchi, N., and Gentile, C. (2010). Linear Algorithms for Online Multitask Classification. *Journal of Machine Learning Research*, 11:2901–2934.

Cer, D., Yang, Y., Kong, S.-y., Hua, N., and Limtiaco, N. (2018). Universal Sentence Encoder.

Cer, D. M., Diab, M. T., Agirre, E., Lopez-Gazpio, I., and Specia, L. (2017). Semeval-2017 task 1: Semantic textual similarity multilingual and crosslingual focused evaluation. In *SemEval@ACL*.

Chan, Y. S. and Ng, H. T. (2006). Estimating class priors in domain adaptation for word sense disambiguation. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, pages 89–96. Association for Computational Linguistics.

Chandar, S., Lauly, S., Larochelle, H., Khapra, M. M., Ravindran, B., Raykar, V., and Saha, A. (2014). An autoencoder approach to learning bilingual word representations. In *Proceedings of NIPS*, pages 1853–1861.

Chapelle, O., Schölkopf, B., and Zien, A. (2006). *Semi-Supervised Learning*. MIT Press.

Chattopadhyay, R., Sun, Q., Ye, J., Panchanathan, S., Fan, W., and Davidson, I. (2012). Multi-Source Domain Adaptation and Its Application to Early Detection of Fatigue. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 6(4).

Chawla, N. V., Bowyer, K. W., Hall, L. O., and Kegelmeyer, W. P. (2002). SMOTE : Synthetic Minority Over-sampling Technique. *Journal of Artificial Intelligence Research*, 16:321–357.

Chelba, C. and Acero, A. (2006). Adaptation of maximum entropy capitalizer: Little data can help a lot. *Computer Speech & Language*, 20(4):382–399.

Chen, B., Lam, W., Tsang, I., and Wong, T.-L. (2009). Extracting discriminative concepts for domain adaptation in text mining. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 179–188. ACM.

Chen, H., Zhang, Y., and Liu, Q. (2016). Neural Network for Heterogeneous Annotations. In *Proceedings of EMNLP*.

Chen, J., Qiu, X., Liu, P., and Huang, X. (2018). Meta Multi-Task Learning for Sequence Modeling. In *Proceedings of AAAI 2018*.

Chen, M. (2017). Efficient Vector Representation for Documents Through Corruption. In *ICLR 2017*.

Chen, M., Weinberger, K. Q., and Blitzer, J. C. (2011a). Co-Training for Domain Adaptation. In *Advances in Neural Information Processing Systems*.

Chen, M., Weinberger, K. Q., and Chen, Y. (2011b). Automatic Feature Decomposition for Single View Co-training. *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 953–960.

Chen, M., Xu, Z., Weinberger, K. Q., and Sha, F. (2012). Marginalized Denoising Autoencoders for Domain Adaptation. *Proceedings of the 29th International Conference on Machine Learning (ICML-12)*, pages 767—-774.

Chen, Q., Zhu, X., Ling, Z., Wei, S., Jiang, H., and Inkpen, D. (2017a). Enhanced LSTM for Natural Language Inference. In *Proceedings of ACL 2017*.

Chen, Q., Zhu, X., Ling, Z.-H., Wei, S., Jiang, H., and Inkpen, D. (2017b). Recurrent neural network-based sentence encoder with gated attention for natural language inference. *arXiv preprint arXiv:1708.01353*.

Chen, X. and Cardie, C. (2018). Multinomial Adversarial Networks for Multi-Domain Text Classification. *Proceedings of NAACL-HLT 2018*.

Chen, X., Kim, S., Lin, Q., Carbonell, J. G., and Xing, E. P. (2010). Graph-Structured Multi-task Regression and an Efficient Optimization Method for General Fused Lasso. pages 1–21.

Chen, Z., Badrinarayanan, V., Lee, C.-Y., and Rabinovich, A. (2017c). GradNorm: Gradient Normalization for Adaptive Loss Balancing in Deep Multitask Networks. pages 1–10.

Chen, Z. and Liu, B. (2014). Topic Modeling using Topics from Many Domains, Lifelong Learning and Big Data. In *Proceedings of ICML*, volume 32, pages 703–711.

Chen, Z., Ma, N., and Liu, B. (2015). Lifelong Learning for Sentiment Classification. In *Proceedings of ACL*, pages 750–756.

Cheng, H., Fang, H., and Ostendorf, M. (2015). Open-Domain Name Error Detection using a Multi-Task RNN. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 737–746.

Choi, E., Hewlett, D., Uszkoreit, J., Lacoste, A., and Berant, J. (2017). Coarse-to-Fine Question Answering for Long Documents. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, pages 209–220.

Chu, Z., Gianvecchio, S., Wang, H., and Jajodia, S. (2012). Detecting automation of twitter accounts: Are you a human, bot, or cyborg? *IEEE Transactions on Dependable and Secure Computing*, 9(6):811–824.

Church, K. W. and Hanks, P. (1990). Word association norms, mutual information, and lexicography. *Computational Linguistics*, 16(1):22–29.

Ciaramita, M. and Chapelle, O. (2010). Adaptive parameters for entity recognition with perceptron hmms. In *Proceedings of the 2010 Workshop on Domain Adaptation for Natural Language Processing*, pages 1–7. Association for Computational Linguistics.

Clark, K., Luong, T., and Le., Q. V. (2018). Semi-Supervised Sequence Modeling with Cross-View Training. In *Proceedings of EMNLP 2018*.

Collobert, R. and Weston, J. (2008). A unified architecture for natural language processing. *Proceedings of the 25th international conference on Machine learning - ICML '08*, 20(1):160–167.

Collobert, R., Weston, J., Bottou, L., Karlen, M., Kavukcuoglu, K., and Kuksa, P. (2011). Natural language processing (almost) from scratch. *The Journal of Machine Learning Research*, 12:2493–2537.

Conneau, A., Kiela, D., Schwenk, H., Barrault, L., and Bordes, A. (2017). Supervised Learning of Universal Sentence Representations from Natural Language Inference Data. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*.

Conneau, A., Lample, G., Ranzato, M., Denoyer, L., and Jégou, H. (2018a). Word Translation Without Parallel Data. In *Proceedings of ICLR 2018*.

Conneau, A., Lample, G., Rinott, R., Williams, A., Bowman, S. R., Schwenk, H., and Stoyanov, V. (2018b). XNLI: Evaluating Cross-lingual Sentence Representations. In *Proceedings of EMNLP 2018*.

Cordella, L. P., Foggia, P., Sansone, C., and Vento, M. (2001). An improved algorithm for matching large graphs. *Proceedings of the 3rd IAPR TC-15 Workshop on Graphbased Representations in Pattern Recognition*, 17:1–35.

Coulmance, J., Marty, J.-M., Wenzek, G., and Benhalloum, A. (2015). Trans-gram, fast cross-lingual word-embeddings. In *Proceedings of EMNLP*, pages 1109–1113.

Crammer, K. and Mansour, Y. (2012). Learning Multiple Tasks Using Shared Hypotheses. *Neural Information Processing Systems (NIPS)*, pages 1484–1492.

Csiszár, I. (1967). Information-type measures of difference of probability distributions and indirect observation. *studia scientiarum Mathematicarum Hungarica*, 2:229–318.

Csurka, G. (2017). Domain Adaptation for Visual Applications: A Comprehensive Survey. *arXiv preprint arXiv:1702.05374*.

Csurka, G. and Chidlovskii, B. (2016). A Domain Adaptation Regularization for Denoising Autoencoders. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pages 26–31.

Dai, A. M. and Le, Q. V. (2015). Semi-supervised Sequence Learning. *Advances in Neural Information Processing Systems (NIPS '15)*.

Dai, W., Jin, O., Xue, G.-R., Yang, Q., and Yu, Y. (2009). Eigentransfer: a unified framework for transfer learning. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 193–200. ACM.

Dai, W., Xue, G.-R., Yang, Q., and Yu, Y. (2007). Transferring naive bayes classifiers for text classification. In *AAAI*, volume 7, pages 540–545.

Dai, W., Yang, Q., Xue, G.-R., and Yu, Y. (2008). Self-taught clustering. In *Proceedings of the 25th international conference on Machine learning*, pages 200–207. ACM.

Daumé III, H. (2007). Frustratingly Easy Domain Adaptation. *Association for Computational Linguistic (ACL)*, (June):256–263.

Daumé III, H. (2009). Bayesian multitask learning with latent hierarchies. pages 135–142.

Daumé III, H., Kumar, A., and Saha, A. (2010). Frustratingly Easy Semi-Supervised Domain Adaptation. *Proceedings of the 2010 Workshop on Domain Adaptation for Natural Language Processing, ACL 2010*, (July):53–59.

Daumé III, H. and Marcu, D. (2006). Domain Adaptation for Statistical Classifiers. *Journal of Artificial Intelligence Research*, 26:101–126.

Deerwester, S., Dumais, S. T., Furnas, G. W., Landauer, T. K., and Harshman, R. (1990). Indexing by latent semantic analysis. *Journal of the American society for information science*, 41(6):391–407.

Deng, J., Dong, W., Socher, R., Li, L.-j., Li, K., and Fei-fei, L. (2009). ImageNet: A Large-Scale Hierarchical Image Database. In *IEEE Conference on Computer Vision and Pattern Recognition*.

Deng, L., Hinton, G. E., and Kingsbury, B. (2013). New types of deep neural network learning for speech recognition and related applications: An overview. *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 8599–8603.

Denkowski, M. and Neubig, G. (2017). Stronger Baselines for Trustable Results in Neural Machine Translation. In *Workshop on Neural Machine Translation (WNMT)*.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2018). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding.

Dhillon, P. S., Foster, D., and Ungar, L. (2011). Multi-View Learning of Word Embeddings via CCA. In *Proceedings of NIPS 2011*.

Dhingra, B., Liu, H., Salakhutdinov, R., and Cohen, W. W. (2017). A Comparative Study of Word Embeddings for Reading Comprehension. *arXiv preprint arXiv:1703.00993*.

Dinu, G., Lazaridou, A., and Baroni, M. (2015). Improving Zero-Shot Learning by Mitigating the Hubness Problem. *ICLR 2015 Workshop track*, pages 1–10.

Doersch, C., Gupta, A., and Efros, A. A. (2015). Unsupervised visual representation learning by context prediction. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1422–1430.

Dolan, W. B. and Brockett, C. (2005). Automatically constructing a corpus of sentential paraphrases. In *Proceedings of the Third International Workshop on Paraphrasing (IWP2005)*.

Donahue, J., Jia, Y., Vinyals, O., Hoffman, J., Zhang, N., Tzeng, E., and Darrell, T. (2014). DeCAF: A Deep Convolutional Activation Feature for Generic Visual Recognition. In *Proceedings of ICML 2014*.

Dong, D., Wu, H., He, W., Yu, D., and Wang, H. (2015). Multi-Task Learning for Multiple Language Translation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, pages 1723–1732.

Dong, L., Wei, F., Tan, C., Tang, D., Zhou, M., and Xu, K. (2014). Adaptive Recursive Neural Network for Target-dependent Twitter Sentiment Classification. In *Proceedings of ACL*, pages 49–54.

Dowty, D. (1991). Thematic proto-roles and argument selection. *language*, 67(3):547–619.

Dozat, T. and Manning, C. D. (2017). Deep Biaffine Attention for Neural Dependency Parsing. In *Proceedings of ICLR 2017*.

Dredze, M., Blitzer, J., Talukdar, P. P., Ganchev, K., Graca, J., and Pereira, F. (2007). Frustratingly hard domain adaptation for dependency parsing. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*.

Duan, L., Tsang, I. W., Xu, D., and Chua, T.-S. (2009). Domain Adaptation from Multiple Sources via Auxiliary Classifiers. In *Proceedings of the 26th Annual International Conference on Machine Learning*.

Duh, K., Neubig, G., Sudoh, K., and Tsukada, H. (2013). Adaptation Data Selection using Neural Language Models: Experiments in Machine Translation. *ACL-2013*, (1):678–683.

Duong, L., Cohn, T., Bird, S., and Cook, P. (2015a). Cross-lingual transfer for unsupervised dependency parsing without parallel data. In *Proceedings of CoNLL*, pages 113–122.

Duong, L., Cohn, T., Bird, S., and Cook, P. (2015b). Low Resource Dependency Parsing: Cross-lingual Parameter Sharing in a Neural Network Parser. *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Short Papers)*, pages 845–850.

Duong, L., Kanayama, H., Ma, T., Bird, S., and Cohn, T. (2016). Learning crosslingual word embeddings without bilingual corpora. In *Proceedings of EMNLP*, pages 1285–1295.

Dyer, C., Chahuneau, V., and Smith, N. (2013). A simple, fast, and effective parameterization of IBM Model 2. In *Proceedings of NAACL-HLT*, pages 644–649.

Džeroski, S. and Ženko, B. (2004). Is combining classifiers with stacking better than selecting the best one? *Machine learning*, 54(3):255–273.

Efron, B. and Tibshirani, R. J. (1994). *An introduction to the bootstrap*. CRC press.

Eisner, B., Rocktäschel, T., Augenstein, I., Bosnjak, M., and Riedel, S. (2016). emoji2vec: Learning Emoji Representations from their Description. In *Proceedings of SocialNLP*.

El-Gamal, M. A. (1991). The role of priors in active Bayesian learning in the sequential statistical decision framework. In *Maximum Entropy and Bayesian Methods*, pages 33–38. Springer Netherlands.

Elazar, Y. and Goldberg, Y. (2018). Adversarial Removal of Demographic Attributes from Text Data. In *Proceedings of EMNLP 2018*.

Elliott, D., Frank, S., Sima'an, K., and Specia, L. (2016). Multi30k: Multilingual English-German image descriptions. In *Proceedings of the 5th Workshop on Vision and Language*, pages 70–74.

Elliott, D. and Kádár, Á. (2017). Imagination improves Multimodal Translation.

Elman, J. L. (1990). Finding structure in time. *Cognitive science*, 14(2):179–211.

Erhan, D., Bengio, Y., Courville, A., Manzagol, P.-A., Vincent, P., and Bengio, S. (2010). Why does unsupervised pre-training help deep learning? *Journal of Machine Learning Research*, 11(Feb):625–660.

Evgeniou, T., Micchelli, C. A., and Pontil, M. (2005). Learning multiple tasks with kernel methods. *Journal of Machine Learning Research*, 6:615–637.

Evgeniou, T. and Pontil, M. (2004). Regularized multi-task learning. *International Conference on Knowledge Discovery and Data Mining*, page 109.

Fan, X., Monti, E., Mathias, L., and Dreyer, M. (2017a). Transfer Learning for Neural Semantic Parsing. *ACL Repl4NLP 2017*.

Fan, Y., Tian, F., Qin, T., Bian, J., and Liu, T.-Y. (2017b). Learning What Data to Learn. In *Workshop track - ICLR 2017*.

Fang, M. and Cohn, T. (2017). Model Transfer for Tagging Low-resource Languages using a Bilingual Dictionary. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL 2017)*.

Faruqui, M., Dodge, J., Jauhar, S. K., Dyer, C., Hovy, E., and Smith, N. A. (2015). Retrofitting word vectors to semantic lexicons. In *Proceedings of NAACL-HLT*, pages 1606–1615.

Faruqui, M. and Dyer, C. (2014). Improving Vector Space Word Representations Using Multilingual Correlation. *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pages 462 – 471.

Faruqui, M., Tsvetkov, Y., Rastogi, P., and Dyer, C. (2016). Problems with evaluation of word embeddings using word similarity tasks. In *Proceedings of REPEVAL*, pages 30–35.

Felbo, B., Mislove, A., Søgaard, A., Rahwan, I., and Lehmann, S. (2017). Using millions of emoji occurrences to learn any-domain representations for detecting sentiment, emotion and sarcasm. In *Proceedings of EMNLP*.

Finkel, J. R. and Manning, C. D. (2009). Hierarchical bayesian domain adaptation. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 602–610. Association for Computational Linguistics.

Finn, C., Abbeel, P., and Levine, S. (2017). Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks. In *Proceedings of ICML*.

Foster, G., Goutte, C., and Kuhn, R. (2010). Discriminative instance weighting for domain adaptation in statistical machine translation. In *Proceedings of the 2010 conference on empirical methods in natural language processing*, pages 451–459. Association for Computational Linguistics.

Fu, L., Nguyen, T. H., Min, B., and Grishman, R. (2017). Domain Adaptation for Relation Extraction with Domain Adversarial Neural Network. In *Proceedings of the The 8th International Joint Conference on Natural Language Processing*, pages 425–429.

Ganin, Y. and Lempitsky, V. (2015). Unsupervised Domain Adaptation by Backpropagation. In *Proceedings of the 32nd International Conference on Machine Learning.*, volume 37.

Ganin, Y., Ustinova, E., Ajakan, H., Germain, P., Larochelle, H., Laviolette, F., Marchand, M., and Lempitsky, V. (2016). Domain-Adversarial Training of Neural Networks. *Journal of Machine Learning Research*, 17:1–35.

Gardner, M., Grus, J., Neumann, M., Tafjord, O., Dasigi, P., Liu, N. F., Peters, M., Schmitz, M., and Zettlemoyer, L. S. (2017). Allennlp: A deep semantic natural language processing platform.

Gardner, M., Huang, K., Paplexakis, E., Fu, X., Talukdar, P., Faloutsos, C., Sidiropoulos, N., Mitchell, T., and Sidiropoulos, N. (2015). Translation invariant word embeddings. In *Proceedings of EMNLP*, pages 1084–1088.

Gella, S., Sennrich, R., Keller, F., and Lapata, M. (2017). Image pivoting for learning multilingual multimodal representations. In *Proceedings of EMNLP*, pages 2829–2835.

Gers, F. A., Schmidhuber, J., and Cummins, F. (1999). Learning to forget: Continual prediction with lstm.

Gerz, D., Vulić, I., Hill, F., Reichart, R., and Korhonen, A. (2016). SimVerb-3500: A large-scale evaluation set of verb similarity. In *Proceedings of EMNLP*, pages 2173–2182.

Gillick, D., Brunk, C., Vinyals, O., and Subramanya, A. (2016). Multilingual Language Processing From Bytes. *NAACL*, pages 1296–1306.

Girshick, R. (2015). Fast R-CNN. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1440–1448.

Glorot, X., Bordes, A., and Bengio, Y. (2011). Domain Adaptation for Large-Scale Sentiment Classification: A Deep Learning Approach. *Proceedings of the 28th International Conference on Machine Learning*, pages 513–520.

Go, A., Bhayani, R., and Huang, L. (2009). Twitter sentiment classification using distant supervision. *CS224N Project Report, Stanford*, 1(12).

Goldberg, Y. and Levy, O. (2014). word2vec Explained: Deriving Mikolov et al.'s Negative-Sampling Word-Embedding Method. *arXiv preprint arXiv:1402.3722*.

Golub, D., Huang, P.-s., He, X., and Deng, L. (2017). Two-Stage Synthesis Networks for Transfer Learning in Machine Comprehension. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*.

Gong, C., He, D., Tan, X., Qin, T., Wang, L., and Liu, T.-Y. (2018). FRAGE: Frequency-Agnostic Word Representation. In *Proceedings of NIPS 2018*.

Goodfellow, I., Bengio, Y., Courville, A., and Bengio, Y. (2016). *Deep learning*, volume 1. MIT press Cambridge.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014). Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680.

Goodfellow, I. J., Shlens, J., and Szegedy, C. (2015). Explaining and Harnessing Adversarial Examples. In *Proceedings of ICLR 2015*.

Gordon, C., Webb, D. L., and Wolpert, S. (1992). One cannot hear the shape of a drum. *Bulletin of the American Mathematical Society*.

Gouws, S., Bengio, Y., and Corrado, G. (2015). BilBOWA: Fast Bilingual Distributed Representations without Word Alignments. *Proceedings of The 32nd International Conference on Machine Learning*, pages 748–756.

Gouws, S. and Søgaard, A. (2015). Simple task-specific bilingual word embeddings. In *Proceedings of NAACL-HLT*, pages 1302–1306.

Gower, J. C. (1975). Generalized procrustes analysis. *Psychometrika*, 40(1):33–51.

Gower, J. C. and Dijksterhuis, G. B. (2004). *Procrustes problems*. Oxford University Press.

Grave, E., Joulin, A., and Berthet, Q. (2018). Unsupervised Alignment of Embeddings with Wasserstein Procrustes. *arXiv preprint arXiv:1805.11222*.

Graves, A., Jaitly, N., and Mohamed, A.-r. (2013). Hybrid speech recognition with deep bidirectional lstm. In *Automatic Speech Recognition and Understanding (ASRU), 2013 IEEE Workshop on*, pages 273–278. IEEE.

Gretton, A., Borgwardt, K. M., Rasch, M. J., Schölkopf, B., and Smola, A. (2012). A kernel two-sample test. *Journal of Machine Learning Research*, 13(Mar):723–773.

Gu, J., Wang, Y., Chen, Y., Cho, K., and Li, V. O. K. (2018). Meta-Learning for Low-Resource Neural Machine Translation. In *Proceedings of EMNLP 2018*.

Gulordava, K., Bojanowski, P., Grave, E., Linzen, T., and Baroni, M. (2018). Colorless green recurrent networks dream hierarchically. In *Proceedings of NAACL-HLT 2018*.

Gulrajani, I., Ahmed, F., Arjovsky, M., Dumoulin, V., and Courville, A. C. (2017). Improved training of wasserstein gans. In *Advances in Neural Information Processing Systems*, pages 5767–5777.

Guo, C., Pleiss, G., Sun, Y., and Weinberger, K. Q. (2017). On Calibration of Modern Neural Networks. *Proceedings of ICML 2017*.

Guo, H., Zhu, H., Guo, Z., Zhang, X., Wu, X., and Su, Z. (2009). Domain adaptation with latent semantic association for named entity recognition. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 281–289. Association for Computational Linguistics.

Guo, J., Che, W., Wang, H., and Liu, T. (2016). Exploiting Multi-typed Treebanks for Parsing with Deep Multi-task Learning.

Guo, J., Che, W., Yarowsky, D., Wang, H., and Liu, T. (2015). Cross-lingual dependency parsing based on distributed representations. In *Proceedings of ACL*, pages 1234–1244.

Guo, J., Shah, D. J., and Barzilay, R. (2018). Multi-Source Domain Adaptation with Mixture of Experts. In *Proceedings of EMNLP 2018*.

Gutmann, M. U. and Hyvärinen, A. (2012). Noise-contrastive estimation of unnormalized statistical models, with applications to natural image statistics. *Journal of Machine Learning Research*, 13(1):307–361.

Ha, D., Dai, A. M., and Le, Q. V. (2017). HyperNetworks. In *Proceedings of ICLR 2017*.

Haghighi, A., Liang, P., Berg-Kirkpatrick, T., and Klein, D. (2008). Learning bilingual lexicons from monolingual corpora. In *Proceedings of ACL*, pages 771–779.

Hariharan, B., Arbeláez, P., Girshick, R., and Malik, J. (2015). Hypercolumns for object segmentation and fine-grained localization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 447–456.

Hashimoto, K., Xiong, C., Tsuruoka, Y., and Socher, R. (2017). A Joint Many-Task Model: Growing a Neural Network for Multiple NLP Tasks. In *Proceedings of EMNLP*.

Hauer, B., Nicolai, G., and Kondrak, G. (2017). Bootstrapping unsupervised bilingual lexicon induction. In *Proceedings of EACL*, pages 619–624.

Hayes, S. C., Barnes-Holmes, D., and Roche, B. (2002). Relational frame theory: A précis. In *Relational frame theory*, pages 141–154. Springer.

He, K., Girshick, R., and Dollár, P. (2018a). Rethinking ImageNet Pre-training. *arXiv preprint arXiv:1811.08883*.

He, K., Gkioxari, G., Dollár, P., and Girshick, R. (2017). Mask R-CNN. In *ICCV 2017*.

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep Residual Learning for Image Recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

He, R., Lee, W. S., Ng, H. T., and Dahlmeier, D. (2018b). Adaptive Semi-supervised Learning for Cross-domain Sentiment Classification. In *Proceedings of EMNLP 2018*, pages 3467–3476.

He, Y. and Zhou, D. (2011). Self-Training from Labeled Features for Sentiment Analysis. *Information Processing & Management*, 47(4):606–616.

Hermann, K., Hill, F., Green, S., Wang, F., Faulkner, R., Soyer, H., Szepesvari, D., Czarnecki, W., Jaderberg, M., Teplyashin, D., Wainwright, M., Apps, C., Hassabis, D., and Blunsom, P. (2017). Grounded Language Learning in a Simulated 3D World.

Hermann, K. M. and Blunsom, P. (2013). Multilingual distributed representations without word alignment. In *Proceedings of ICLR (Conference Track)*.

Hermann, K. M. and Blunsom, P. (2014). Multilingual models for compositional distributed semantics. In *Proceedings of ACL*, pages 58–68.

Hershcovich, D., Abend, O., and Rappoport, A. (2018). Multitask Parsing Across Semantic Representations. In *Proceedings of ACL 2018*.

Heskes, T. (2000). Empirical Bayes for Learning to Learn. *Proceedings of the Seventeenth International Conference on Machine Learning*, pages 367–364.

Hill, F., Cho, K., and Korhonen, A. (2016a). Learning Distributed Representations of Sentences from Unlabelled Data. In *NAACL-HLT*.

Hill, F., Cho, K., Korhonen, A., and Bengio, Y. (2016b). Learning to Understand Phrases by Embedding the Dictionary. *Transactions of the Association for Computational Linguistics*, 4:17–30.

Hinton, G., Vinyals, O., and Dean, J. (2015). Distilling the Knowledge in a Neural Network. *arXiv preprint arXiv:1503.02531*.

Hinton, G. E., Osindero, S., and Teh, Y. W. (2006). A Fast Learning Algorithm for Deep Belief Nets. *Neural Computation*, 18:1527–1554.

Hochreiter, S. and Schmidhuber, J. (1997). Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780.

Horn, R. A. and Johnson, C. R. (2012). *Matrix Analysis*. Cambridge University Press.

Hotelling, H. (1992). The generalization of student's ratio. In *Breakthroughs in statistics*, pages 54–65. Springer.

Hovy, E., Marcus, M., Palmer, M., Ramshaw, L., and Weischedel, R. (2006). OntoNotes: The 90 % Solution. In *Proceedings of NAACL-HLT*.

Howard, J. and Ruder, S. (2018). Universal Language Model Fine-tuning for Text Classification. In *Proceedings of ACL 2018*.

Huang, F. and Yates, A. (2009). Distributional representations for handling sparsity in supervised sequence-labeling. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 1-Volume 1*, pages 495–503. Association for Computational Linguistics.

Huang, F. and Yates, A. (2010a). Exploring representation-learning approaches to domain adaptation. In *Proceedings of the 2010 Workshop on Domain Adaptation for Natural Language Processing*, pages 23–30. Association for Computational Linguistics.

Huang, F. and Yates, A. (2010b). Open-domain semantic role labeling by modeling word spans. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 968–978. Association for Computational Linguistics.

Huang, G., Li, Y., Pleiss, G., Liu, Z., Hopcroft, J. E., and Weinberger, K. Q. (2017a). Snapshot Ensembles: Train 1, get M for free. In *Proceedings of ICLR 2017*.

Huang, G., Liu, Z., Weinberger, K. Q., and van der Maaten, L. (2017b). Densely Connected Convolutional Networks. In *Proceedings of CVPR 2017*.

Huang, J., Gretton, A., Borgwardt, K. M., Schölkopf, B., and Smola, A. J. (2007). Correcting sample selection bias by unlabeled data. In *Advances in neural information processing systems*, pages 601–608.

Huang, P.-s., Wang, C., Singh, R., Yih, W.-t., and He, X. (2018). Natural Language to Structured Query Generation via Meta-Learning. In *Proceedings of NAACL-HLT 2018*.

Huang, S., Li, K., Dai, X., and Chen, J. (2010). Improving word alignment by semi-supervised ensemble. In *Proceedings of the Fourteenth Conference on Computational Natural Language Learning*, pages 135–143. Association for Computational Linguistics.

Inan, H., Khosravi, K., and Socher, R. (2016). Tying Word Vectors and Word Classifiers: A Loss Framework for Language Modeling. *arXiv preprint arXiv:1611.01462*.

Ioffe, S. and Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning*, pages 448–456.

Irvine, A. and Callison-Burch, C. (2013). Supervised bilingual lexicon induction with multiple monolingual signals. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 518–523, Atlanta, Georgia. Association for Computational Linguistics.

Iyyer, M., Manjunatha, V., Boyd-Graber, J., and Daumé III, H. (2015). Deep Unordered Composition Rivals Syntactic Methods for Text Classification. *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1681–1691.

Jacob, L., Vert, J.-P., Bach, F. R., and Vert, J.-p. (2009). Clustered Multi-Task Learning: A Convex Formulation. In *Proceedings of NIPS*, pages 745–752.

Jacobs, R. a., Jordan, M. I., Nowlan, S. J., and Hinton, G. E. (1991). Adaptive Mixtures of Local Experts. *Neural Computation*, 3(1):79–87.

Jalali, A., Ravikumar, P., Sanghavi, S., and Ruan, C. (2010). A Dirty Model for Multi-task Learning. *Advances in Neural Information Processing Systems*.

Jernite, Y., Bowman, S. R., and Sontag, D. (2017). Discourse-Based Objectives for Fast Unsupervised Sentence Representation Learning.

Ji, Y.-S., Chen, J.-J., Niu, G., Shang, L., and Dai, X.-Y. (2011). Transfer learning via multi-view principal component analysis. *Journal of Computer Science and Technology*, 26(1):81–98.

Jia, R. and Liang, P. (2017). Adversarial Examples for Evaluating Reading Comprehension Systems. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*.

Jiang, J. (2008). A literature survey on domain adaptation of statistical classifiers. (March).

Jiang, J. (2009). Multi-task transfer learning for weakly-supervised relation extraction. *Proceedings of the 47th Annual Meeting of the ACL and the 4th IJCNLP of the AFNLP*, (August):1012–1020.

Jiang, J. and Zhai, C. (2007). Instance Weighting for Domain Adaptation in NLP. *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, (October):264–271.

Jindal, N. and Liu, B. (2007). Review spam detection. In *Proceedings of the 16th international conference on World Wide Web*, pages 1189–1190. ACM.

Johnson, M., Schuster, M., Le, Q. V., Krikun, M., Wu, Y., Chen, Z., Thorat, N., Viégas, F., Wattenberg, M., Corrado, G., Hughes, M., and Dean, J. (2016). Google's Multilingual Neural Machine Translation System: Enabling Zero-Shot Translation.

Johnson, R. and Zhang, T. (2016). Supervised and semi-supervised text categorization using lstm for region embeddings. In *International Conference on Machine Learning*, pages 526–534.

Johnson, R. and Zhang, T. (2017). Deep pyramid convolutional neural networks for text categorization. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 562–570.

Jonker, R. and Volgenant, A. (1987). A shortest augmenting path algorithm for dense and sparse linear assignment problems. *Computing*, 38(4):325–340.

Joshi, V., Peters, M., and Hopkins, M. (2018). Extending a Parser to Distant Domains Using a Few Dozen Partially Annotated Examples. In *Proceedings of ACL 2018*.

Kaiser, Ł., Nachum, O., Roy, A., and Bengio, S. (2017). Learning to Remember Rare Events. In *ICLR 2017*.

Kalchbrenner, N., Grefenstette, E., and Blunsom, P. (2014). A Convolutional Neural Network for Modelling Sentences. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, pages 655–665.

Kang, Z., Grauman, K., and Sha, F. (2011). Learning with whom to share in multi-task feature learning. *Proceedings of the 28th International Conference on Machine Learning*, (4):4–5.

Katiyar, A. and Cardie, C. (2017). Going out on a limb : Joint Extraction of Entity Mentions and Relations without Dependency Trees. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, pages 917–928.

Kementchedjhieva, Y., Ruder, S., Cotterell, R., and Søgaard, A. (2018). Generalizing Procrustes Analysis for Better Bilingual Dictionary Induction. In *Proceedings of CoNLL 2018*.

Kendall, A., Gal, Y., and Cipolla, R. (2018). Multi-Task Learning Using Uncertainty to Weigh Losses for Scene Geometry and Semantics. In *Proceedings of CVPR 2018*.

Kiela, D. and Clark, S. (2015). Multi- and cross-modal semantics beyond vision: Grounding in auditory perception. In *Proceedings of EMNLP*, pages 2461–2470.

Kiela, D., Conneau, A., Jabri, A., and Nickel, M. (2018a). Learning Visually Grounded Sentence Representations. In *Proceedings of NAACL-HLT 2018*.

Kiela, D., Vulić, I., and Clark, S. (2015). Visual bilingual lexicon induction with transferred ConvNet features. In *Proceedings of EMNLP*, pages 148–158.

Kiela, D., Wang, C., and Cho, K. (2018b). Dynamic Meta-Embeddings for Improved Sentence Representations. In *Proceedings of EMNLP 2018*.

Kifer, D., Ben-David, S., and Gehrke, J. (2004). Detecting change in data streams. In *Proceedings of the Thirtieth international conference on Very large data bases-Volume 30*, pages 180–191. VLDB Endowment.

Kim, J.-D., Ohta, T., Tsuruoka, Y., Tateisi, Y., and Collier, N. (2004). Introduction to the bio-entity recognition task at jnlpba. In *Proceedings of the international joint workshop on natural language processing in biomedicine and its applications*, pages 70–75. Association for Computational Linguistics.

Kim, S. and Xing, E. P. (2010). Tree-Guided Group Lasso for Multi-Task Regression with Structured Sparsity. *27th International Conference on Machine Learning*, pages 1–14.

Kim, Y. (2014). Convolutional Neural Networks for Sentence Classification. *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1746–1751.

Kim, Y.-B., Stratos, K., and Kim, D. (2017a). Adversarial adaptation of synthetic or stale data. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1297–1307, Vancouver, Canada. Association for Computational Linguistics.

Kim, Y.-b., Stratos, K., and Kim, D. (2017b). Domain Attention with an Ensemble of Experts. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, pages 643–653.

Kim, Y.-b., Stratos, K., and Sarikaya, R. (2016). Frustratingly Easy Neural Domain Adaptation. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 387–396.

Kim, Y.-B., Stratos, K., Sarikaya, R., and Jeong, M. (2015). New Transfer Learning Techniques for Disparate Label Sets. In *Proceedings of ACL*.

Kingma, D. P. and Ba, J. L. (2015). Adam: a Method for Stochastic Optimization. *Proceedings of ICLR*.

Kingsbury, P. and Palmer, M. (2002). From treebank to propbank. In *LREC*, pages 1989–1993. Citeseer.

Kiperwasser, E. and Ballesteros, M. (2018). Scheduled Multi-Task Learning : From Syntax to Translation. *Transactions of the Association for Computational Linguistics*, 6:225–240.

Kiperwasser, E. and Goldberg, Y. (2016). Simple and Accurate Dependency Parsing Using Bidirectional LSTM Feature Representations. *Transactions of the Association for Computational Linguistics*, 4:313–327.

Kirkpatrick, J., Pascanu, R., Rabinowitz, N., Veness, J., Desjardins, G., Rusu, A. A., Milan, K., Quan, J., Ramalho, T., Grabska-Barwinska, A., Hassabis, D., Clopath, C., Kumaran, D., and Hadsell, R. (2017). Overcoming catastrophic forgetting in neural networks. *PNAS*.

Kiros, J. R. and Chan, W. (2018). InferLite: Simple Universal Sentence Representations from Natural Language Inference Data. In *Proceedings of EMNLP 2018*, pages 4868–4874.

Kiros, R., Zhu, Y., Salakhutdinov, R., Zemel, R. S., Torralba, A., Urtasun, R., and Fidler, S. (2015). Skip-Thought Vectors. In *Proceedings of NIPS 2015*.

Klein, A., Falkner, S., and Hutter, F. (2017). Fast Bayesian Optimization of Machine Learning Hyperparameters on Large Datasets. In *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics (AISTATS 2017)*.

Klementiev, A., Irvine, A., Callison-Burch, C., and Yarowsky, D. (2012a). Toward statistical machine translation without parallel corpora. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pages 130–140, Avignon, France. Association for Computational Linguistics.

Klementiev, A., Titov, I., and Bhattarai, B. (2012b). Inducing crosslingual distributed representations of words. In *Proceedings of COLING*, pages 1459–1474.

Kočiský, T., Hermann, K. M., and Blunsom, P. (2014). Learning bilingual word representations by marginalizing alignments. In *Proceedings of ACL*, pages 224–229.

Koehn, P. (2005). Europarl: A parallel corpus for statistical machine translation. In *Proceedings of MT Summit*, pages 79–86.

Koehn, P. (2009). *Statistical Machine Translation*. Cambridge University Press.

Koehn, P., Och, F. J., and Marcu, D. (2003). Statistical phrase-based translation. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*, pages 48–54. Association for Computational Linguistics.

Koo, T., Carreras, X., and Collins, M. (2008). Simple semi-supervised dependency parsing. *Proceedings of ACL-08: HLT*, pages 595–603.

Kornblith, S., Shlens, J., Le, Q. V., and Brain, G. (2018). Do Better ImageNet Models Transfer Better? *arXiv preprint arXiv:1805.08974*.

Krallinger, M., Leitner, F., Rabal, O., Vazquez, M., Oyarzabal, J., and Valencia, A. (2013). Overview of the chemical compound and drug name recognition (chemdner) task. In *BioCreative challenge evaluation workshop*, volume 2, page 2.

Krishna, K., Jyothi, P., and Iyyer, M. (2018). Revisiting the Importance of Encoding Logic Rules in Sentiment Classification. In *Proceedings of EMNLP 2018*, pages 4743–4751.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012a). Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012b). ImageNet Classification with Deep Convolutional Neural Networks. *Advances In Neural Information Processing Systems*, pages 1–9.

Kuhn, H. W. (1955). The Hungarian method for the assignment problem. *Naval Research Logistics (NRL)*, 2(1-2):83–97.

Kumar, A. and Daumé III, H. (2012). Learning Task Grouping and Overlap in Multi-task Learning. *Proceedings of the 29th International Conference on Machine Learning*, pages 1383–1390.

Kumar, A., Irsoy, O., Su, J., Bradbury, J., English, R., Pierce, B., Ondruska, P., Iyyer, M., Gulrajani, I., and Socher, R. (2016a). Ask Me Anything: Dynamic Memory Networks for Natural Language Processing. In *Proceedings of ICML 2016*.

Kumar, A., Kohail, S., Kumar, A., Ekbal, A., and Biemann, C. (2016b). IIT-TUDA at SemEval-2016 Task 5: Beyond Sentiment Lexicon: Combining Domain Dependency and Distributional Semantics Features for Aspect Based Sentiment Analysis. *Proceedings of SemEval*.

Laine, S. and Aila, T. (2017). Temporal Ensembling for Semi-Supervised Learning. In *Proceedings of ICLR 2017*.

Lample, G., Ballesteros, M., Subramanian, S., Kawakami, K., and Dyer, C. (2016). Neural Architectures for Named Entity Recognition. In *NAACL-HLT 2016*.

Lample, G., Denoyer, L., and Ranzato, M. (2018a). Unsupervised machine translation using monolingual corpora only. In *Proceedings of ICLR (Conference Papers)*.

Lample, G., Ott, M., Conneau, A., Denoyer, L., and Ranzato, M. (2018b). Phrase-Based & Neural Unsupervised Machine Translation. In *Proceedings of EMNLP 2018*.

Lauly, S., Boulanger, A., and Larochelle, H. (2013). Learning multilingual word representations using a bag-of-words autoencoder. In *Proceedings of the NIPS Workshop on Deep Learning*, pages 1–8.

Lawrence, N. D. and Platt, J. C. (2004). Learning to learn with the informative vector machine. *Twenty-first international conference on Machine learning - ICML '04*, page 65.

Lazaridou, A., Dinu, G., and Baroni, M. (2015). Hubness and Pollution: Delving into Cross-Space Mapping for Zero-Shot Learning. *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, pages 270–280.

Le, Q. V. and Mikolov, T. (2014). Distributed Representations of Sentences and Documents. *International Conference on Machine Learning - ICML 2014*, 32:1188–1196.

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324.

Lee, L. (2001). On the Effectiveness of the Skew Divergence for Statistical Language Analysis. *AISTATS (Artificial Intelligence and Statistics)*, pages 65–72.

Levy, O. and Goldberg, Y. (2014). Neural Word Embedding as Implicit Matrix Factorization. *Advances in Neural Information Processing Systems (NIPS)*, pages 2177–2185.

Levy, O., Goldberg, Y., and Dagan, I. (2015). Improving distributional similarity with lessons learned from word embeddings. *Transactions of the ACL*, 3:211–225.

Levy, O., Søgaard, A., and Goldberg, Y. (2017). A Strong Baseline for Learning Cross-Lingual Word Embeddings from Sentence Alignments. In *EACL 2017*.

Li, M. and Zhou, Z.-H. (2007). Improve Computer-Aided Diagnosis With Machine Learning Techniques Using Undiagnosed Samples. *IEEE Transactions on Systems, Man and Cybernetics*, 37(6):1088–1098.

Li, S. and Zong, C. (2008a). Multi-domain Adaptation for Sentiment Classication: Using Multiple Classifier Combining Methods. In *International Conference on Natural Language Processing and Knowledge Engineering (NLP-KE'08)*. IEEE.

Li, S. and Zong, C. (2008b). Multi-domain sentiment classification. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics on Human Language Technologies: Short Papers*, pages 257–260. Association for Computational Linguistics.

Li, Y., Baldwin, T., and Cohn, T. (2018a). Learning Domain-Robust Text Representations using Adversarial Training. In *Proceedings of NAACL-HLT 2018*.

Li, Y., Baldwin, T., and Cohn, T. (2018b). Towards Robust and Privacy-preserving Text Representations. In *Proceedings of ACL 2018*.

Liang, J., Meyerson, E., and Miikkulainen, R. (2018). Evolutionary Architecture Search For Deep Multitask Networks. In *GECCO*.

Lin, J. (1991). Divergence Measures Based on the Shannon Entropy. *IEEE Transactions on Information theory*, 37(1):145–151.

Linzen, T., Dupoux, E., and Goldberg, Y. (2016). Assessing the ability of lstms to learn syntax-sensitive dependencies. *arXiv preprint arXiv:1611.01368*.

Liu, L., Shang, J., Xu, F., Ren, X., Gui, H., Peng, J., and Han, J. (2018a). Empower sequence labeling with task-aware neural language model. In *Proceedings of AAAI 2018*.

Liu, P., Qiu, X., and Huang, X. (2017). Adversarial Multi-task Learning for Text Classification. In *Proceedings of ACL*.

Liu, Q., Liu, B., Zhang, Y., Kim, D. S., and Gao, Z. (2016a). Improving Opinion Aspect Extraction Using Semantic Similarity and Aspect Associations. *Thirtieth AAAI Conference on Artificial Intelligence (AAAI-16) Improving*, pages 2986–2992.

Liu, Q., Zhang, Y., and Liu, J. (2018b). Learning Domain Representation for Multi-Domain Sentiment Classification. In *Proceedings of NAACL-HLT 2018*, pages 541–550.

Liu, S., Pan, S. J., and Ho, Q. (2016b). Distributed Multi-task Relationship Learning. In *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 751–760.

Liu, X., Gao, J., He, X., Deng, L., Duh, K., and Wang, Y.-Y. (2015). Representation Learning Using Multi-Task Deep Neural Networks for Semantic Classification and Information Retrieval. *NAACL-2015*, pages 912–921.

Ljubešić, N., Pirinen, T., and Toral, A. (2016). Finnish Web corpus fiWaC 1.0. Slovenian language resource repository CLARIN.SI.

Logeswaran, L. and Lee, H. (2018). An efficient framework for learning sentence representations. In *Proceedings of ICLR 2018*.

Long, J., Shelhamer, E., and Darrell, T. (2015a). Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3431–3440.

Long, M., Cao, Y., Wang, J., and Jordan, M. I. (2015b). Learning Transferable Features with Deep Adaptation Networks. In *Proceedings of ICML*, volume 37, Lille, France.

Long, M. and Wang, J. (2015). Learning Multiple Tasks with Deep Relationship Networks. *arXiv preprint arXiv:1506.02117*.

Lopez-Paz, D. and Ranzato, M. (2017). Gradient Episodic Memory for Continuum Learning. In *Proceedings of NIPS 2017*.

Loshchilov, I. and Hutter, F. (2017a). Fixing Weight Decay Regularization in Adam. *arXiv preprint arXiv:1711.05101*.

Loshchilov, I. and Hutter, F. (2017b). SGDR: Stochastic Gradient Descent with Warm Restarts. In *Proceedings of the Internal Conference on Learning Representations 2017*.

Louizos, C., Swersky, K., Li, Y., Welling, M., and Zemel, R. (2015). The variational fair autoencoder. *arXiv preprint arXiv:1511.00830*.

Lounici, K., Pontil, M., Tsybakov, A. B., and van de Geer, S. (2009). Taking Advantage of Sparsity in Multi-Task Learning. *Stat*, (1).

Lu, A., Wang, W., Bansal, M., Gimpel, K., and Livescu, K. (2015). Deep multilingual correlation for improved word embeddings. In *Proceedings of NAACL-HLT*, pages 250–256.

Lu, W., Chieu, H. L., and Jonathan, L. (2016). A General Regularization Framework for Domain Adaptation. *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP2016)*, (3):950–954.

Lu, W. and Zheng, V. W. (2017). A Simple Regularization-based Algorithm for Learning Cross-Domain Word Embeddings. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2888–2894.

Luong, M.-T., Le, Q. V., Sutskever, I., Vinyals, O., and Kaiser, L. (2016). Multi-task Sequence to Sequence Learning. In *Proceedings of ICLR 2016*.

Luong, M.-T., Pham, H., and Manning, C. D. (2015). Bilingual Word Representations with Monolingual Quality in Mind. *Workshop on Vector Modeling for NLP*, pages 151–159.

Lynn, V. E., Son, Y., Kulkarni, V., Balasubramanian, N., Schwartz, H. A., and Brook, S. (2017). Human Centered NLP with User-Factor Adaptation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*.

Ma, J., Zhang, Y., and Zhu, J. (2014). Tagging The Web: Building A Robust Web Tagger with Neural Network. *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 144–154.

Maas, A. L., Daly, R. E., Pham, P. T., Huang, D., Ng, A. Y., and Potts, C. (2011). Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 142–150. Association for Computational Linguistics.

MacKay, D. J. C. (1992). Information-Based Objective Functions for Active Data Selection. *Neural Computation*, 4(4):590–604.

Mahajan, D., Girshick, R., Ramanathan, V., He, K., Paluri, M., Li, Y., Bharambe, A., and van der Maaten, L. (2018). Exploring the Limits of Weakly Supervised Pretraining.

Malaviya, C., Neubig, G., and Littell, P. (2017). Learning Language Representations for Typology Prediction. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*.

Manning, C. D., , and Schütze, H. (1999). *Foundations of statistical natural language processing*. MIT press.

Manning, C. D. (2015). Computational linguistics and deep learning. *Computational Linguistics*, 41(4):701–707.

Mansour, Y. (2009). Domain Adaptation with Multiple Sources. *Neural Information Processing Systems Conference (NIPS 2009)*.

Marcus, M. P., Marcinkiewicz, M. A., and Santorini, B. (1993). Building a large annotated corpus of english: The penn treebank. *Computational linguistics*, 19(2):313–330.

Marelli, M., Menini, S., Baroni, M., Bentivogli, L., Bernardi, R., Zamparelli, R., et al. (2014). A sick cure for the evaluation of compositional distributional semantic models. In *LREC*, pages 216–223.

Margolis, A. (2011). A Literature Review of Domain Adaptation with Unlabeled Data.

Margolis, A., Livescu, K., and Ostendorf, M. (2010). Domain adaptation with unlabeled data for dialog act tagging. In *Proceedings of the 2010 Workshop on Domain Adaptation for Natural Language Processing*, pages 45–52. Association for Computational Linguistics.

Matthews, B. W. (1975). Comparison of the predicted and observed secondary structure of t4 phage lysozyme. *Biochimica et Biophysica Acta (BBA)-Protein Structure*, 405(2):442–451.

Maurer, A. (2007). Bounds for Linear Multi Task Learning. *JMLR*, 7:117–139.

Maurer, A., Pontil, M., and Romera-paredes, B. (2013). Sparse coding for multitask and transfer learning. In *ICML*, volume 28, pages 343–351.

Mayhew, S., Dan, C.-t. T., and Goodwin, N. (2017). Cheap Translation for Cross-Lingual Named Entity Recognition. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2526–2535.

McCann, B., Bradbury, J., Xiong, C., and Socher, R. (2017). Learned in Translation: Contextualized Word Vectors. In *Advances in Neural Information Processing Systems*.

McClosky, D., Charniak, E., and Johnson, M. (2006a). Effective self-training for parsing. *Proceedings of the main conference on human language technology conference of the North American Chapter of the Association of Computational Linguistics*, pages 152–159.

McClosky, D., Charniak, E., and Johnson, M. (2006b). Reranking and Self-Training for Parser Adaptation. *International Conference on Computational Linguistics (COLING) and Annual Meeting of the Association for Computational Linguistics (ACL)*, (July):337–344.

McClosky, D., Charniak, E., and Johnson, M. (2010). Automatic domain adaptation for parsing. In *Human Language Technologies: The 2010 Annual Conference of the*

North American Chapter of the Association for Computational Linguistics, pages 28–36. Association for Computational Linguistics.

Melis, G., Dyer, C., and Blunsom, P. (2017). On the State of the Art of Evaluation in Neural Language Models. In *arXiv preprint arXiv:1707.05589*.

Mena, G., Belanger, D., Linderman, S., and Snoek, J. (2018). Learning latent permutations with Gumbel-Sinkhorn networks. *arXiv preprint arXiv:1802.08665*.

Merity, S., Shirish Keskar, N., and Socher, R. (2017a). Regularizing and Optimizing LSTM Language Models. *arXiv preprint arXiv:1708.02182*.

Merity, S., Xiong, C., Bradbury, J., and Socher, R. (2017b). Pointer Sentinel Mixture Models. In *Proceedings of the International Conference on Learning Representations 2017*.

Meyerson, E. and Miikkulainen, R. (2018). Beyond Shared Hierarchies: Deep Multitask Learning through Soft Layer Ordering. In *ICPR*.

Michel, P. and Neubig, G. (2018). Extreme Adaptation for Personalized Neural Machine Translation. In *Proceedings of ACL 2018*.

Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013a). Distributed Representations of Words and Phrases and their Compositionality. In *Proceedings of NIPS*, pages 3111–3119.

Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013b). Efficient estimation of word representations in vector space. In *International Conference on Learning Representations (ICLR) Workshop*.

Mikolov, T., Le, Q. V., and Sutskever, I. (2013c). Exploiting similarities among languages for machine translation.

Miller, S., Guinness, J., and Zamanian, A. (2004). Name tagging with word clusters and discriminative training. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics: HLT-NAACL 2004*.

Min, S., Seo, M., and Hajishirzi, H. (2017). Question Answering through Transfer Learning from Large Fine-grained Supervision Data. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Short Papers)*.

Mintz, M., Bills, S., Snow, R., and Jurafsky, D. (2009). Distant supervision for relation extraction without labeled data. In *Proceedings of the 47th Annual Meeting of the ACL and the 4th IJCNLP of the AFNLP*, pages 1003–1011.

Mirkin, S. and Besacier, L. (2014). Data selection for compact adapted smt models. In *Eleventh Conference of the Association for Machine Translation in the Americas (AMTA)*.

Misra, I., Shrivastava, A., Gupta, A., and Hebert, M. (2016). Cross-stitch Networks for Multi-task Learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

Mitchell, T., Cohen, W., Hruschka, E., Talukdar, P., Yang, B., Betteridge, J., Carlson, A., Dalvi, B., Gardner, M., Kisiel, B., Krishnamurthy, J., Lao, N., Mazaitis, K., Mohamed, T., Nakashole, N., Platanios, E., Ritter, A., Samadi, M., Settles, B., Wang, R., Wijaya, D., Gu, A., and Welling, J. (2018). Never-Ending Learning. *Communications of the ACM*, 61(5).

Miyato, T., Dai, A. M., and Goodfellow, I. (2017). Adversarial Training Methods for Semi-supervised Text Classification. In *Proceedings of ICLR 2017*.

Mnih, A. and Teh, Y. W. (2012). A fast and simple algorithm for training neural probabilistic language models. *Proceedings of ICML*, pages 1751–1758.

Močkus, J. (1974). On bayesian methods for seeking the extremum. In G.I., M., editor, *Optimization Techniques IFIP Technical Conference Novosibirsk*.

Mogadala, A. and Rettinger, A. (2016). Bilingual Word Embeddings from Parallel and Non-parallel Corpora for Cross-Language Text Classification. *NAACL*, pages 692–702.

Mohammad, S., Kiritchenko, S., Sobhani, P., Zhu, X., and Cherry, C. (2016). Semeval-2016 task 6: Detecting stance in tweets. In *Proceedings of SemEval*.

Moore, R. C. and Lewis, W. D. (2010). Intelligent Selection of Language Model Training Data. *ACL-2010: 48th Annual Meeting of the Association for Computational Linguistics*, (July):220–224.

Mou, L., Meng, Z., Yan, R., Li, G., Xu, Y., Zhang, L., and Jin, Z. (2016). How Transferable are Neural Networks in NLP Applications? *Proceedings of 2016 Conference on Empirical Methods in Natural Language Processing*.

Mou, L., Peng, H., Li, G., Xu, Y., Zhang, L., and Jin, Z. (2015). Discriminative neural sentence modeling by tree-based convolution. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*.

Mrkšić, N., Vulić, I., Ó Séaghdha, D., Leviant, I., Reichart, R., Gašić, M., Korhonen, A., and Young, S. (2017). Semantic specialisation of distributional word vector spaces using monolingual and cross-lingual constraints. *Transactions of the ACL*, 5:309–324.

Mulcaire, P., Swayamdipta, S., and Smith, N. (2018). Polyglot Semantic Role Labeling. In *Proceedings of ACL 2018*.

Müller, A. (1997). Integral probability metrics and their generating classes of functions. *Advances in Applied Probability*, 29(2):429–443.

Munteanu, D. S. and Marcu, D. (2006). Extracting parallel sub-sentential fragments from non-parallel corpora. In *Proceedings of ACL*, pages 81–88.

Myers, J. L., Well, A., and Lorch, R. F. (2010). *Research Design and Statistical Analysis*. Routledge.

Nakov, P., Ritter, A., Rosenthal, S., Stoyanov, V., and Sebastiani, F. (2016). SemEval-2016 Task 4: Sentiment Analysis in Twitter. In *Proceedings of SemEval*, San Diego, California.

Nangia, N., Williams, A., Lazaridou, A., and Bowman, S. R. (2017). The RepEval 2017 Shared Task: Multi-Genre Natural Language Inference with Sentence Representations. In *Proceedings of RepEval*.

National Research Council and Automatic Language Processing Advisory Committee (1966). *Language and Machines: Computers in Translation and Linguistics; A Report*. National Academy of Sciences, National Research Council.

Neal, R. M. and Hinton, G. E. (1998). A new view of the EM algorithm that justifies incremental, sparse and other variants. In Jordan, M. I., editor, *Learning in Graphical Models*, pages 355–368. Kluwer Academic Publishers.

Negahban, S. and Wainwright, M. J. (2008). Joint support recovery under high-dimensional scaling: Benefits and perils of ℓ_1, ∞-regularization. *Advances in Neural Information Processing Systems*, pages 1161–1168.

Neubig, G., Dyer, C., Goldberg, Y., Matthews, A., Ammar, W., Anastasopoulos, A., Ballesteros, M., Chiang, D., Clothiaux, D., Cohn, T., et al. (2017). Dynet: The dynamic neural network toolkit. *arXiv preprint arXiv:1701.03980*.

Ngai, E., Hu, Y., Wong, Y., Chen, Y., and Sun, X. (2011). The application of data mining techniques in financial fraud detection: A classification framework and an academic review of literature. *Decision Support Systems*, 50(3):559–569.

Nichol, A. and Schulman, J. (2018). Reptile: a Scalable Metalearning Algorithm. *arXiv preprint arXiv:1803.02999*.

Nie, A., Bennett, E. D., and Goodman, N. D. (2017). DisSent: Sentence Representation Learning from Explicit Discourse Relations. *arXiv preprint arXiv:1710.04334*.

Niehues, J. and Cho, E. (2017). Exploiting Linguistic Resources for Neural Machine Translation Using Multi-task Learning. In *WMT 2017*.

Nigam, K. and Ghani, R. (2000). Analyzing the effectiveness and applicability of co-training. In *Proceedings of the ninth international conference on Information and knowledge management*, pages 86–93. ACM.

Nivre, J., de Marneffe, M.-C., Ginter, F., Goldberg, Y., Hajic, J., Manning, C. D., McDonald, R. T., Petrov, S., Pyysalo, S., Silveira, N., et al. (2016). Universal Dependencies v1: A multilingual treebank collection. In *Proceedings of LREC*.

Noshad, M., Zeng, Y., and III, A. O. H. (2018). Scalable Mutual Information Estimation using Dependence Graphs. *arXiv preprint arXiv:1801.09125*.

Oliver, A., Odena, A., Raffel, C., Cubuk, E. D., and Goodfellow, I. J. (2018). Realistic Evaluation of Semi-Supervised Learning Algorithms. In *ICLR Workshop 2018*.

Owoputi, O., Connor, B. O., Dyer, C., Gimpel, K., Schneider, N., and Smith, N. A. (2013). Improved Part-of-Speech Tagging for Online Conversational Text with Word Clusters. In *Proceedings of NAACL 2013*.

Pagliardini, M., Gupta, P., and Jaggi, M. (2018). Unsupervised Learning of Sentence Embeddings using Compositional n-Gram Features. In *Proceedings of NAACL-HLT 2018*, pages 528–540.

Palogiannidi, E., Kolovou, A., Christopoulou, F., Kokkinos, F., Iosif, E., Malandrakis, N., Papageorgiou, H., Narayanan, S., and Potamianos, A. (2016). Tweester at SemEval-2016 Task 4: Sentiment Analysis in Twitter Using Semantic-Affective Model Adaptation. In *Proceedings of SemEval*, pages 155–163.

Pan, S. J., Kwok, J. T., and Yang, Q. (2008). Transfer learning via dimensionality reduction. In *AAAI*, volume 8, pages 677–682.

Pan, S. J., Ni, X., Sun, J.-t., Yang, Q., and Chen, Z. (2010). Cross-Domain Sentiment Classification via Spectral Feature Alignment. In *Proceedings of the 19th International Conference on World Wide Web*, pages 751–760.

Pan, S. J., Tsang, I. W., Kwok, J. T., and Yang, Q. (2011). Domain adaptation via transfer component analysis. *IEEE Transactions on Neural Networks*, 22(2):199–210.

Pan, S. J. and Yang, Q. (2010). A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, 22(10):1345–1359.

Papandreou, G., Zhu, T., Kanazawa, N., Toshev, A., Tompson, J., Bregler, C., and Murphy, K. (2017). Towards accurate multi-person pose estimation in the wild. In *CVPR*, volume 3, page 6.

Pappas, N. and Popescu-belis, A. (2017). Multilingual Hierarchical Attention Networks for Document Classification. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*.

Pasunuru, R. and Bansal, M. (2017). Multi-Task Video Captioning with Video and Entailment Generation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL 2017)*.

Patel, V. and Gopalan, R. (2015). Visual Domain Adaptation: A survey of recent advances. *IEEE signal processing magazine*, 32(3):53–69.

Peirsman, Y. and Padó, S. (2011). Semantic relations in bilingual lexicons. *ACM Transactions on Speech and Language Processing (TSLP)*, 8(2):3.

Peng, H., Thomson, S., Smith, N. A., and Allen, P. G. (2017). Deep Multitask Learning for Semantic Dependency Parsing. In *Proceedings of ACL 2017*.

Peng, N. and Dredze, M. (2016). Multi-task Multi-domain Representation Learning for Sequence Tagging.

Pennington, J., Socher, R., and Manning, C. D. (2014). Glove: Global Vectors for Word Representation. *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pages 1532–1543.

Peters, M., Ruder, S., and Smith, N. A. (2019). To Tune or Not to Tune? Adapting Pretrained Representations to Diverse Tasks. *arXiv preprint arXiv:1903.05987*.

Peters, M. E., Ammar, W., Bhagavatula, C., and Power, R. (2017). Semi-supervised sequence tagging with bidirectional language models. In *Proceedings of ACL 2017*.

Peters, M. E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., and Zettlemoyer, L. (2018a). Deep contextualized word representations. In *Proceedings of NAACL 2018*.

Peters, M. E., Neumann, M., Zettlemoyer, L., Yih, W.-t., Allen, P. G., and Science, C. (2018b). Dissecting Contextual Word Embeddings: Architecture and Representation. In *Proceedings of EMNLP 2018*.

Petrov, S., Chang, P.-C., Ringgaard, M., and Alshawi, H. (2010). Uptraining for accurate deterministic question parsing. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 705–713. Association for Computational Linguistics.

Petrov, S., Das, D., and McDonald, R. (2012). A Universal Part-of-Speech Tagset. In *Proceedings of LREC 2012*.

Petrov, S. and McDonald, R. (2012). Overview of the 2012 shared task on parsing the web. *Notes of the First Workshop on Syntactic Analysis of Non-Canonical Language (SANCL)*, 59.

Pham, H., Luong, M.-T., and Manning, C. D. (2015). Learning distributed representations for multilingual text sequences. In *Proceedings of the Workshop on Vector Modeling for NLP*, pages 88–94.

Pinker, S. (2013). *Learnability and cognition: The acquisition of argument structure*. MIT press.

Plank, B. (2016). What to do about non-standard (or non-canonical) language in NLP. *KONVENS 2016*.

Plank, B., Johannsen, A., and Søgaard, A. (2014). Importance weighting and unsupervised domain adaptation of pos taggers: a negative result. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 968–973.

Plank, B., Klerkě, S., and Agic, Z. (2018). The Best of Both Worlds: Lexical Resources To Improve Low-Resource Part-of-Speech Tagging. *arXiv preprint arXiv:1811.08757*.

Plank, B., Søgaard, A., and Goldberg, Y. (2016). Multilingual Part-of-Speech Tagging with Bidirectional Long Short-Term Memory Models and Auxiliary Loss. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*.

Plank, B. and van Noord, G. (2011). Effective Measures of Domain Similarity for Parsing. *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, 1:1566–1576.

Poczos, B., Xiong, L., and Schneider, J. (2011). Nonparametric Divergence Estimation with Applications to Machine Learning on Distributions. In *Proceedings of the Twenty-Seventh Conference on Uncertainty in Artificial Intelligence*.

Ponomareva, N. and Thelwall, M. (2012). Do Neighbours Help? An Exploration of Graph-based Algorithms for Cross-domain Sentiment Classification. *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, (July):655–665.

Pontiki, M., Galanis, D., Papageorgiou, H., Androutsopoulos, I., Manandhar, S., AL-Smadi, M., Al-Ayyoub, M., Zhao, Y., Qin, B., De Clercq, O., Hoste, V., Apidianaki, M., Tannier, X., Loukachevitch, N., Kotelnikov, E., Bel, N., Jiménez-Zafra, S. M., and Eryiğit, G. (2016). SemEval-2016 Task 5: Aspect Based Sentiment Analysis. In *Proceedings of SemEval*.

Porteous, I., Newman, D., Ihler, A., Asuncion, A., Smyth, P., and Welling, M. (2008). Fast collapsed gibbs sampling for latent dirichlet allocation. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 569–577. ACM.

Prettenhofer, P. and Stein, B. (2010). Cross-language text classification using structural correspondence learning. In *Proceedings of the 48th annual meeting of the association for computational linguistics*, pages 1118–1127. Association for Computational Linguistics.

Qu, L., Ferraro, G., Zhou, L., Hou, W., and Baldwin, T. (2016). Named Entity Recognition for Novel Types by Transfer Learning. *Proceedings of EMNLP*.

Radford, A., Jozefowicz, R., and Sutskever, I. (2017). Learning to generate reviews and discovering sentiment. *arXiv preprint arXiv:1704.01444*.

Radford, A., Narasimhan, K., Salimans, T., and Sutskever, I. (2018). Improving Language Understanding by Generative Pre-Training.

Radovanović, M., Nanopoulos, A., and Ivanovic, M. (2010). Hubs in space: Popular nearest neighbors in high-dimensional data. *Journal of Machine Learning Research*, 11:2487–2531.

Raina, R., Battle, A., Lee, H., Packer, B., and Ng, A. Y. (2007). Self-taught learning: Transfer Learning from Unlabeled Data. *Proceedings of the 24th international conference on Machine learning - ICML '07*, pages 759–766.

Rajendran, J., Khapra, M. M., Chandar, S., and Ravindran, B. (2016). Bridge correlational neural networks for multilingual multimodal representation learning. In *Proceedings of NAACL-HLT*, pages 171–181.

Ramachandran, P., Liu, P. J., and Le, Q. V. (2017). Unsupervised Pretraining for Sequence to Sequence Learning. In *Proceedings of EMNLP 2017*.

Ramsundar, B., Kearnes, S., Riley, P., Webster, D., Konerding, D., and Pande, V. (2015). Massively Multitask Networks for Drug Discovery.

Rao, C. R. (1982). Diversity and dissimilarity coefficients: a unified approach. *Theoretical population biology*, 21(1):24–43.

Rasmussen, C. E. (2006). *Gaussian processes for machine learning*. MIT Press.

Ratinov, L. and Roth, D. (2009). Design challenges and misconceptions in named entity recognition. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning*, pages 147–155. Association for Computational Linguistics.

Ratner, A., De Sa, C., Wu, S., Selsam, D., and Ré, C. (2016). Data Programming: Creating Large Training Sets, Quickly. In *Advances in Neural Information Processing Systems 29 (NIPS 2016)*.

Ravi, S., Knight, K., and Soricut, R. (2008). Automatic prediction of parser accuracy. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 887–896. Association for Computational Linguistics.

Rebuffi, S.-A., Bilen, H., and Vedaldi, A. (2017). Learning multiple visual domains with residual adapters. In *Proceedings of NIPS 2017*.

Rebuffi, S.-A., Bilen, H., and Vedaldi, A. (2018). Efficient parametrization of multi-domain deep neural networks. In *Proceedings of CVPR 2018*.

Recht, B., Fazel, M., and Parrilo, P. A. (2010). Guaranteed minimum-rank solutions of linear matrix equations via nuclear norm minimization. *SIAM review*, 52(3):471–501.

Reddy, S., Chen, D., and Manning, C. D. (2018). CoQA: A Conversational Question Answering Challenge.

Rei, M. (2017). Semi-supervised Multitask Learning for Sequence Labeling. In *Proceedings of ACL 2017*.

Reichart, R. and Rappoport, A. (2007). Self-training for enhancement and domain adaptation of statistical parsers trained on small datasets. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 616–623.

Reimers, N. and Gurevych, I. (2017). Reporting score distributions makes a difference: Performance study of lstm-networks for sequence tagging. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 338–348, Copenhagen, Denmark. Association for Computational Linguistics.

Remus, R. (2012). Domain adaptation using Domain Similarity- and Domain Complexity-based Instance Selection for Cross-Domain Sentiment Analysis. In *IEEE ICDM SENTIRE-2012*.

Ren, M., Triantafillou, E., Ravi, S., Snell, J., Swersky, K., Tenenbaum, J. B., Larochelle, H., and Zemel, R. S. (2018). Meta-Learning for Semi-Supervised Few-Shot Classification. In *Proceedings of ICLR 2018*.

Rényi, A. (1961). On measures of entropy and information. In *Proceedings of the fourth Berkeley symposium on mathematical statistics and probability*, volume 1.

Riedel, B., Augenstein, I., Spithourakis, G. P., and Riedel, S. (2017). A simple but tough-to-beat baseline for the Fake News Challenge stance detection task. In *arXiv preprint arXiv:1707.03264*.

Riedel, S., Yao, L., McCallum, A., and Marlin, B. M. (2013). Relation Extraction with Matrix Factorization and Universal Schemas. *Proceedings of NAACL-HLT*, pages 74–84.

Riemer, M., Khabiri, E., and Goodwin, R. (2017). Representation Stability as a Regularizer for Improved Text Analytics Transfer Learning. *arXiv preprint arXiv:1704.03617*.

Roark, B. and Bacchiani, M. (2003). Supervised and unsupervised pcfg adaptation to novel domains. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*, pages 126–133. Association for Computational Linguistics.

Roitblat, H. L., Kershaw, A., and Oot, P. (2010). Document categorization in legal electronic discovery: computer classification vs. manual review. *Journal of the Association for Information Science and Technology*, 61(1):70–80.

Rosenstein, M. T., Marx, Z., Kaelbling, L. P., and Dietterich, T. G. (2005). To Transfer or Not To Transfer. *NIPS Workshop on Inductive Transfer*.

Roy, D. (2017). Twitter Demographic Classification Using Deep Multi-modal Multi-task Learning. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, pages 478–483.

Ruder, S. (2016). An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*.

Ruder, S. (2017). An Overview of Multi-Task Learning in Deep Neural Networks. In *arXiv preprint arXiv:1706.05098*.

Ruder, S. (2018). NLP's ImageNet moment has arrived. *The Gradient*.

Ruder, S., Bingel, J., Augenstein, I., and Søgaard, A. (2019a). Latent Multi-task Architecture Learning. In *Proceedings of AAAI 2019*.

Ruder, S., Cotterell, R., Kementchedjhieva, Y., and Søgaard, A. (2018). A Discriminative Latent-Variable Model for Bilingual Lexicon Induction. In *Proceedings of EMNLP 2018*.

Ruder, S., Ghaffari, P., and Breslin, J. G. (2016a). A Hierarchical Model of Reviews for Aspect-based Sentiment Analysis. *Proceedings of EMNLP*, pages 999–1005.

Ruder, S., Ghaffari, P., and Breslin, J. G. (2016b). Towards a continuous modeling of natural language domains. In *Uphill Battles in Language Processing Workshop, EMNLP 2016*.

Ruder, S., Ghaffari, P., and Breslin, J. G. (2017a). Data Selection Strategies for Multi-Domain Sentiment Analysis. In *arXiv preprint arXiv:1702.02426*.

Ruder, S., Ghaffari, P., and Breslin, J. G. (2017b). Knowledge Adaptation: Teaching to Adapt. In *arXiv preprint arXiv:1702.02052*.

Ruder, S. and Plank, B. (2017). Learning to select data for transfer learning with Bayesian Optimization. In *Proceedings of EMNLP*.

Ruder, S. and Plank, B. (2018). Strong Baselines for Neural Semi-supervised Learning under Domain Shift. In *Proceedings of ACL 2018*.

Ruder, S., Vulić, I., and Søgaard, A. (2019b). A survey of cross-lingual word embedding models. *Journal of Artificial Intelligence Research*.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986). Learning representations by back-propagating errors. *nature*, 323(6088):533.

Rusu, A. A., Rabinowitz, N. C., Desjardins, G., Soyer, H., Kirkpatrick, J., Kavukcuoglu, K., Pascanu, R., and Hadsell, R. (2016). Progressive Neural Networks. *arXiv preprint arXiv:1606.04671*.

Ruvolo, P. and Eaton, E. (2013). ELLA: An Efficient Lifelong Learning Algorithm. In *Proceedings of ICML 2013*.

Sagae, K. (2010). Self-training without reranking for parser domain adaptation and its impact on semantic role labeling. In *Proceedings of the 2010 Workshop on Domain Adaptation for Natural Language Processing*, pages 37–44. Association for Computational Linguistics.

Sagae, K. and Tsujii, J. (2007). Dependency parsing and domain adaptation with lr models and parser ensembles. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*.

Saha, A., Rai, P., Daumé, H., and Venkatasubramanian, S. (2011). Online learning of multiple tasks and their relationships. *Journal of Machine Learning Research*, 15:643–651.

Saito, K., Ushiku, Y., and Harada, T. (2017). Asymmetric Tri-training for Unsupervised Domain Adaptation. In *ICML 2017*.

Salakhutdinov, R. and Hinton, G. (2009). Deep boltzmann machines. In *Artificial Intelligence and Statistics*, pages 448–455.

Salzmann, M., Ek, C. H., Urtasun, R., and Darrell, T. (2010). Factorized Orthogonal Latent Spaces. *JMLR*, 9:701–708.

Samdani, R., Chang, M.-W., and Roth, D. (2012). Unified expectation maximization. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 688–698, Montréal, Canada. Association for Computational Linguistics.

Sandu, O., Carenini, G., Murray, G., and Ng, R. (2010). Domain adaptation to summarize human conversations. In *Proceedings of the 2010 Workshop on Domain Adaptation for Natural Language Processing*, pages 16–22. Association for Computational Linguistics.

Sang, E. F. T. K. and Meulder, F. D. (2003). Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. In *CoNLL*.

Sanh, V., Wolf, T., and Ruder, S. (2019). A Hierarchical Multi-task Approach for Learning Embeddings from Semantic Tasks. In *Proceedings of AAAI 2019*.

Satpal, S. and Sarawagi, S. (2007). Domain adaptation of conditional probability models via feature subsetting. In *European Conference on Principles of Data Mining and Knowledge Discovery*, pages 224–235. Springer.

Saxe, A. M., Bansal, Y., Dapello, J., Advani, M., Kolchinsky, A., Tracey, B. D., and Cox, D. D. (2018). On the Information Bottleneck Theory of Deep Learning. In *Proceedings of ICLR 2018*.

Schnabel, T. and Schütze, H. (2014). FLORS: Fast and Simple Domain Adaptation for Part-of-Speech Tagging. *TACL*, 2:15–26.

Schönemann, P. H. (1966). A generalized solution of the orthogonal Procrustes problem. *Psychometrika*, 31(1):1–10.

Schwartz, R., Reichart, R., and Rappoport, A. (2015). Symmetric pattern based word embeddings for improved word similarity prediction. In *Proceedings of CoNLL*, pages 258–267.

Schwarz, J., Luketina, J., Czarnecki, W. M., Grabska-Barwinska, A., Teh, Y. W., Pascanu, R., and Hadsell, R. (2018). Progress & Compress : A scalable framework for continual learning. In *Proceedings of ICML 2018*.

Sennrich, R., Haddow, B., and Birch, A. (2015). Improving neural machine translation models with monolingual data. *arXiv preprint arXiv:1511.06709*.

Settles, B. (2012). *Active learning literature survey*. Morgan and Claypool.

Severyn, A. and Moschitti, A. (2015). UNITN: Training Deep Convolutional Neural Network for Twitter Sentiment Classification. *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, pages 464–469.

Shah, D. J., Lei, T., Moschitti, A., Romeo, S., and Nakov, P. (2018). Adversarial Domain Adaptation for Duplicate Question Detection. In *Proceedings of EMNLP 2018*.

Shannon, C. E. (1948). A mathematical theory of communication. *The Bell System Technical Journal*, 27(July 1928):379–423.

Sharif Razavian, A., Azizpour, H., Sullivan, J., and Carlsson, S. (2014). Cnn features off-the-shelf: an astounding baseline for recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 806–813.

Shezaf, D. and Rappoport, A. (2010). Bilingual lexicon generation using non-aligned signatures. In *Proceedings of ACL*, pages 98–107.

Shi, T., Liu, Z., Liu, Y., and Sun, M. (2015). Learning cross-lingual word embeddings via matrix co-factorization. In *Proceedings of ACL*, pages 567–572.

Shigehalli, V. and Shettar, V. (2011). Spectral technique using normalized adjacency matrices for graph matching. *International Journal of Computational Science and Mathematics*, 3:371–378.

Shu, L., Xu, H., and Liu, B. (2017). Lifelong Learning CRF for Supervised Aspect Extraction. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL 2017)*.

Silver, D. L., Yang, Q., and Li, L. (2013). Lifelong Machine Learning Systems : Beyond Learning Algorithms. *AAAI Spring Symposium Series*, pages 49–55.

Simpson, E. H. (1949). Measurement of diversity. *Nature*.

Smith, L. N. (2017). Cyclical learning rates for training neural networks. In *Applications of Computer Vision (WACV), 2017 IEEE Winter Conference on*, pages 464–472. IEEE.

Smith, N. A. (2011). Linguistic structure prediction. *Synthesis lectures on human language technologies*, 4(2):1–274.

Smith, S. L., Turban, D. H. P., Hamblin, S., and Hammerla, N. Y. (2017). Offline bilingual word vectors, orthogonal transformations and the inverted softmax. In *Proceedings of ICLR (Conference Papers)*.

Snell, J., Swersky, K., and Zemel, R. S. (2017). Prototypical Networks for Few-shot Learning. In *Advances in Neural Information Processing Systems*.

Snoek, J., Larochelle, H., and Adams, R. P. (2012). Practical Bayesian Optimization of Machine Learning Algorithms. *Neural Information Processing Systems Conference (NIPS 2012)*.

Socher, R., Perelygin, A., Wu, J., Chuang, J., Manning, C. D., Ng, A., and Potts, C. (2013). Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1631–1642.

Søgaard, A. (2010). Simple semi-supervised training of part-of-speech taggers. In *Proceedings of the ACL 2010 Conference Short Papers*, pages 205–208.

Søgaard, A. (2011). Data Point Selection for Cross-Language Adaptation of Dependency Parsers. *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies (HLT '11): Short Papers*, pages 682–686.

Søgaard, A. (2013). Part-of-speech tagging with antagonistic adversaries. *Proceedings of ACL*, pages 640–644.

Søgaard, A., Agić, Z., Alonso, H. M., Plank, B., Bohnet, B., and Johannsen, A. (2015). Inverted indexing for cross-lingual NLP. In *Proceedings of ACL*, pages 1713–1722.

Søgaard, A. and Goldberg, Y. (2016). Deep multi-task learning with low level tasks supervised at lower layers. *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pages 231–235.

Søgaard, A. and Haulrich, M. (2011). Sentence-level instance-weighting for graph-based and transition-based dependency parsing. In *Proceedings of the 12th International Conference on Parsing Technologies*, pages 43–47. Association for Computational Linguistics.

Søgaard, A. and Rishøj, C. (2010). Semi-supervised dependency parsing using generalized tri-training. In *Proceedings of the 23rd International Conference on Computational Linguistics*, pages 1065–1073. Association for Computational Linguistics.

Søgaard, A., Ruder, S., and Vulić, I. (2018). On the Limitations of Unsupervised Bilingual Dictionary Induction. In *Proceedings of ACL 2018*.

Soyer, H., Stenetorp, P., and Aizawa, A. (2015). Leveraging monolingual data for crosslingual compositional word representations. In *Proceedings of ICLR (Conference Track)*.

Sprechmann, P., Jayakumar, S. M., Rae, J. W., Pritzel, A., Uria, B., and Vinyals, O. (2018). Memory-based Parameter Adaptation. In *Proceedings of ICLR 2018*.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research*, 15:1929–1958.

Su, Y. and Yan, X. (2017). Cross-domain Semantic Parsing via Paraphrasing. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*.

Subramanian, S., Trischler, A., Bengio, Y., and Pal, C. J. (2018). Learning General Purpose Distributed Sentence Representations via Large Scale Multi-task Learning. In *Proceedings of ICLR 2018*.

Sugiyama, M., Nakajima, S., Kashima, H., Buenau, P. V., and Kawanabe, M. (2008). Direct importance estimation with model selection and its application to covariate shift adaptation. In *Advances in neural information processing systems*, pages 1433–1440.

Sun, B., Feng, J., and Saenko, K. (2016). Return of Frustratingly Easy Domain Adaptation. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence (AAAI-16)*.

Suzuki, J. and Isozaki, H. (2008). Semi-supervised sequential labeling and segmentation using giga-word scale unlabeled data. pages 665–673.

Suzuki, J., Isozaki, H., Carreras, X., and Collins, M. (2009). An empirical study of semi-supervised structured conditional models for dependency parsing. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 2-Volume 2*, pages 551–560. Association for Computational Linguistics.

Swayamdipta, S., Thomson, S., Dyer, C., and Smith, N. A. (2017). Frame-Semantic Parsing with Softmax-Margin Segmental RNNs and a Syntactic Scaffold. *arXiv preprint arXiv:1706.09528*.

Swayamdipta, S., Thomson, S., Lee, K., Zettlemoyer, L., Dyer, C., and Smith, N. A. (2018). Syntactic Scaffolds for Semantic Structures. In *Proceedings of EMNLP 2018*.

Świrszcz, G. and Lozano, A. C. (2012). Multi-level Lasso for Sparse Multi-task Regression. In *Proceedings of ICML*.

Sylak-Glassman, J., Kirov, C., Yarowsky, D., and Que, R. (2015). A language-independent feature schema for inflectional morphology. In *Proceedings of ACL*, pages 674–680.

Tan, S. and Cheng, X. (2009). Improving scl model for sentiment-transfer learning. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics, Companion Volume: Short Papers*, pages 181–184. Association for Computational Linguistics.

Tan, S., Cheng, X., Wang, Y., and Xu, H. (2009). Adapting naive bayes to domain adaptation for sentiment analysis. In *European Conference on Information Retrieval*, pages 337–349. Springer.

Thrun, S. (1996). Is Learning The n-th Thing Any Easier Than Learning The First ? In *Proceedings of NIPS 1996*.

Thrun, S. (1998). Lifelong Learning Algorithms. In Thrun, S. and Pratt, L., editors, *Learning to Learn*, pages 181–209. Kluwer Academic Publishers.

Thrun, S. and O'Sullivan, J. (1996). Discovering Structure in Multiple Learning Tasks: The TC Algorithm. In *Proceedings of the Thirteenth International Conference on Machine Learning*, volume 28.

Tiedemann, J. (2009). News from OPUS - A collection of multilingual parallel corpora with tools and interfaces. In *Proceedings of RANLP*, pages 237–248.

Toshniwal, S., Tang, H., Lu, L., and Livescu, K. (2017). Multitask Learning with Low-Level Auxiliary Tasks for Encoder-Decoder Based Speech Recognition.

Tsuboi, Y., Kashima, H., Hido, S., Bickel, S., and Sugiyama, M. (2009). Direct density ratio estimation for large-scale covariate shift adaptation. *Journal of Information Processing*, 17:138–155.

Tsvetkov, Y., Faruqui, M., Ling, W., and Dyer, C. (2016). Learning the Curriculum with Bayesian Optimization for Task-Specific Word Representation Learning. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL 2016)*.

Turian, J., Ratinov, L., and Bengio, Y. (2010). Word representations: A simple and general method for semi-supervised learning. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics,*, number July, pages 384–394.

Tzeng, E., Hoffman, J., Zhang, N., Saenko, K., and Darrell, T. (2014). Deep Domain Confusion: Maximizing for Domain Invariance. *CoRR*.

Upadhyay, S., Faruqui, M., Dyer, C., and Roth, D. (2016). Cross-lingual models of word embeddings: An empirical comparison. In *Proceedings of ACL*, pages 1661–1670.

Valiant, L. G. (1979). The complexity of computing the permanent. *Theoretical Computer Science*, 8(2):189–201.

Van Asch, V. and Daelemans, W. (2010). Using Domain Similarity for Performance Estimation. *Computational Linguistics*, (July):31–36.

Van Asch, V. and Daelemans, W. (2016). Predicting the effectiveness of self-training: Application to sentiment classification. *arXiv preprint arXiv:1601.03288*.

van der Goot, R., Plank, B., and Nissim, M. (2017). To normalize, or not to normalize: The impact of normalization on part-of-speech tagging. In *Proceedings of the 3rd Workshop on Noisy User-generated Text*, pages 31–39, Copenhagen, Denmark. Association for Computational Linguistics.

van der Wees, M., Bisazza, A., and Monz, C. (2017). Dynamic data selection for neural machine translation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*.

Vapnik, V. N. and Chervonenkis, A. Y. (2015). On the uniform convergence of relative frequencies of events to their probabilities. In *Measures of complexity*, pages 11–30. Springer.

Vapnik, V. N. and Kotz, S. (1982). *Estimation of dependences based on empirical data*, volume 40. Springer-Verlag New York.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). Attention Is All You Need. In *Advances in Neural Information Processing Systems*.

Villani, C. (2008). *Optimal transport: Old and new*, volume 338. Springer Science & Business Media.

Vinyals, O., Blundell, C., Lillicrap, T., Kavukcuoglu, K., and Wierstra, D. (2016). Matching Networks for One Shot Learning. In *Advances in Neural Information Processing Systems 29 (NIPS 2016)*.

Vinyals, O., Kaiser, L., Koo, T., Petrov, S., Sutskever, I., and Hinton, G. (2015). Grammar as a Foreign Language. *Advances in Neural Information Processing Systems*.

Virtanen, S., Klami, A., and Kaski, S. (2011). Bayesian CCA via Group Sparsity. In *Proceedings of ICML*.

Vo, D.-T. and Zhang, Y. (2015). Target-Dependent Twitter Sentiment Classification with Rich Automatic Features. In *Proceedings of IJCAI*, pages 1347–1353.

Volgenant, A. (1996). Linear and semi-assignment problems: a core oriented approach. *Computers & Operations Research*, 23(10):917–932.

Voorhees, E. M. and Tice, D. M. (1999). The trec-8 question answering track evaluation. In *TREC*, volume 1999, page 82.

Vulić, I., De Smet, W., and Moens, M.-F. (2011). Identifying word translations from comparable corpora using latent topic models. In *Proceedings of ACL*, pages 479–484.

Vulić, I., Kiela, D., Clark, S., and Moens, M.-F. (2016). Multi-modal representations for improved bilingual lexicon learning. In *Proceedings of ACL*, pages 188–194.

Vulić, I. and Korhonen, A. (2016). On the Role of Seed Lexicons in Learning Bilingual Word Embeddings. *Proceedings of ACL*, pages 247–257.

Vulić, I. and Moens, M.-F. (2013). Cross-lingual semantic similarity of words as the similarity of their semantic word responses. In *Proceedings of NAACL-HLT*, pages 106–116.

Vulić, I. and Moens, M.-F. (2013). A study on bootstrapping bilingual vector spaces from non-parallel data (and nothing else). In *Proceedings of EMNLP*, pages 1613–1624.

Vulić, I. and Moens, M.-F. (2016). Bilingual distributed word representations from document-aligned comparable data. *Journal of Artificial Intelligence Research*, 55:953–994.

Vyas, Y. and Carpuat, M. (2016). Sparse bilingual word representations for cross-lingual lexical entailment. In *Proceedings of NAACL-HLT*, pages 1187–1197.

Wallis, G. and Bülthoff, H. (1999). Learning to recognize objects. *Trends in cognitive sciences*, 3(1):22–31.

Wang, A., Singh, A., Michael, J., Hill, F., Levy, O., and Bowman, S. R. (2018a). GLUE: A Multi-Task Benchmark and Analysis Platform for Natural Language Understanding.

Wang, D. and Zheng, T. F. (2015). Transfer learning for speech and language processing. *2015 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference, APSIPA ASC 2015*, pages 1225–1237.

Wang, S., Chen, Z., and Liu, B. (2016). Mining Aspect-Specific Opinion using a Holistic Lifelong Topic Model. *Proceedings of the 25th International Conference on World Wide Web - WWW '16*, pages 167–176.

Wang, S., Yu, M., Guo, X., Wang, Z., Klinger, T., and Zhang, W. (2017). R^3: Reinforced Reader-Ranker for Open-Domain Question Answering.

Wang, Y. and Bansal, M. (2018). Robust Machine Comprehension Models via Adversarial Training. In *Proceedings of NAACL-HLT 2018*.

Wang, Z., Qu, Y., Chen, L., Shen, J., Zhang, W., Zhang, S., Gao, Y., Gu, G., Chen, K., and Yu, Y. (2018b). Label-aware Double Transfer Learning for Cross-Specialty Medical Named Entity Recognition. In *Proceedings of NAACL-HLT 2018*.

Wei, B. and Pal, C. (2010). Cross lingual adaptation: an experiment on sentiment classifications. In *Proceedings of the ACL 2010 conference short papers*, pages 258–262. Association for Computational Linguistics.

Weischedel, R., Palmer, M., Marcus, M., Hovy, E., Pradhan, S., Ramshaw, L., Xue, N., Taylor, A., Kaufman, J., Franchini, M., El-Bachouti, M., Belvin, R., and Houston, A. (2013). OntoNotes Release 5.0 LDC2013T19. *Linguistic Data Consortium*.

Weng, R., Huang, S., Zheng, Z., Dai, X., and Chen, J. (2017). Neural Machine Translation with Word Predictions. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*.

Werbos, P. J. (1988). Generalization of backpropagation with application to a recurrent gas market model. *Neural networks*, 1(4):339–356.

West, D. B. (2000). *Introduction to Graph Theory*. Prentice Hall, 2 edition.

Weston, J., Chopra, S., and Bordes, A. (2015). Memory Networks. In *Proceedings of ICLR 2015*.

Wiese, G., Weissenborn, D., and Neves, M. (2017). Neural Domain Adaptation for Biomedical Question Answering. In *Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL 2017)*.

Wieting, J., Bansal, M., Gimpel, K., and Livescu, K. (2016). Towards Universal Paraphrastic Sentence Embeddings. In *ICLR*.

Wieting, J. and Gimpel, K. (2017). Revisiting Recurrent Networks for Paraphrastic Sentence Embeddings. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL 2017)*.

Williams, A., Nangia, N., and Bowman, S. (2018). A broad-coverage challenge corpus for sentence understanding through inference. In *NAACL*.

Winograd, T. (1972). Understanding natural language. *Cognitive psychology*, 3(1):1–191.

Wolf, T., Chaumond, J., and Delangue, C. (2018). Continuous Learning in a Hierarchical Multiscale Neural Network. In *Proceedings of ACL 2018*.

Wolpert, D. H. and Macready, W. G. (1997). No free lunch theorems for optimization. *IEEE transactions on evolutionary computation*, 1(1):67–82.

Wong, C. and Gesmundo, A. (2018). Transfer Learning to Learn with Multitask Neural Model Search.

Wu, F. and Huang, Y. (2016). Sentiment Domain Adaptation with Multiple Sources. *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL 2016)*, pages 301–310.

Wu, S., Zhang, D., Yang, N., Li, M., and Zhou, M. (2017). Sequence-to-Dependency Neural Machine Translation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, pages 698–707.

Wu, Y., Schuster, M., Chen, Z., Le, Q. V., Norouzi, M., Macherey, W., Krikun, M., Cao, Y., Gao, Q., Macherey, K., Klingner, J., Shah, A., Johnson, M., Liu, X., Kaiser, Ł., Gouws, S., Kato, Y., Kudo, T., Kazawa, H., Stevens, K., Kurian, G., Patil, N., Wang, W., Young, C., Smith, J., Riesa, J., Rudnick, A., Vinyals, O., Corrado, G., Hughes, M., and Dean, J. (2016). Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation. *arXiv preprint arXiv:1609.08144*.

Xia, R., Zong, C., Hu, X., and Cambria, E. (2015). Feature Ensemble plus Sample Selection: A Comprehensive Approach to Domain Adaptation for Sentiment Classification. *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence (IJCAI 2015) Feature*, PP(99):1.

Xia, Y., He, D., Qin, T., Wang, L., Yu, N., Liu, T.-Y., and Ma, W.-Y. (2016). Dual Learning for Machine Translation. In *Advances in Neural Information Processing Systems 29 (NIPS 2016)*, number Nips.

Xia, Y., Qin, T., Chen, W., Bian, J., Yu, N., and Liu, T.-y. (2017). Dual Supervised Learning. In *ICML*.

Xiao, M. and Guo, Y. (2014). Distributed word representation learning for cross-lingual dependency parsing. In *Proceedings of CoNLL*, pages 119–129.

Xing, C., Liu, C., Wang, D., and Lin, Y. (2015). Normalized word embedding and orthogonal transform for bilingual word translation. In *Proceedings of NAACL-HLT*, pages 1005–1010.

Xu, P., Karakos, D., and Khudanpur, S. (2009). Self-supervised discriminative training of statistical language models. In *Automatic Speech Recognition & Understanding, 2009. ASRU 2009. IEEE Workshop on*, pages 317–322. IEEE.

Xue, Y., Liao, X., Carin, L., and Krishnapuram, B. (2007). Multi-Task Learning for Classification with Dirichlet Process Priors. *Journal of Machine Learning Research*, 8:35–63.

Yang, B. and Mitchell, T. (2017). A Joint Sequential and Relational Model for Frame-Semantic Parsing. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*.

Yang, J., Zhang, Y., and Dong, F. (2017a). Neural Word Segmentation with Rich Pretraining. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL 2017)*.

Yang, Y. and Eisenstein, J. (2014). Fast Easy Unsupervised Domain Adaptation with Marginalized Structured Dropout. *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, 1836:538–544.

Yang, Y. and Eisenstein, J. (2015). Unsupervised Multi-Domain Adaptation with Feature Embeddings. In *Proceedings of NAACL-HLT 2015*, pages 503–513.

Yang, Y. and Hospedales, T. M. (2017). Trace Norm Regularised Deep Multi-Task Learning. In *Workshop track - ICLR 2017*.

Yang, Z., Salakhutdinov, R., and Cohen, W. (2016). Multi-Task Cross-Lingual Sequence Tagging from Scratch.

Yang, Z., Salakhutdinov, R., and Cohen, W. W. (2017b). Transfer Learning for Sequence Tagging with Hierarchical Recurrent Networks. In *Proceedings of ICLR 2017*.

Yarowsky, D. (1995). Unsupervised Word Sense Disambiguation Rivaling Supervised Methods. In *Proceedings of the 33rd annual meeting on Association for Computational Linguistics*.

Yeh, C.-K., Wu, W.-C., Ko, W.-J., and Wang, Y.-C. F. (2017). Learning Deep Latent Space for Multi-Label Classification. In *Proceedings of AAAI*.

Yin, W., Schnabel, T., and Schütze, H. (2015). Online Updating of Word Representations for Part-of-Speech Tagging. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, number September, pages 1329–1334.

Yoshida, Y., Hirao, T., Iwata, T., Nagata, M., and Matsumoto, Y. (2011). Transfer Learning for Multiple-Domain Sentiment Analysis - Identifying Domain Dependent/Independent Word Polarity. In *Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence Transfer*, pages 1286–1291.

Yosinski, J., Clune, J., Bengio, Y., and Lipson, H. (2014). How transferable are features in deep neural networks? In *Advances in neural information processing systems*, pages 3320–3328.

Young, P., Lai, A., Hodosh, M., and Hockenmaier, J. (2014). From image descriptions to visual denotations: New similarity metrics for semantic inference over event descriptions. *Transactions of the ACL*, 2:67–78.

Yu, J. and Jiang, J. (2016). Learning Sentence Embeddings with Auxiliary Tasks for Cross-Domain Sentiment Classification. *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP2016)*, pages 236–246.

Yu, K., Tresp, V., and Schwaighofer, A. (2005). Learning Gaussian processes from multiple tasks. *Proceedings of the International Conference on Machine Learning (ICML)*, 22:1012–1019.

Yuan, M. and Lin, Y. (2006). Model selection and estimation in regression with grouped variables. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 68(1):49–67.

Zamir, A. R., Sax, A., Shen, W., Guibas, L., Malik, J., and Savarese, S. (2018). Taskonomy: Disentangling Task Transfer Learning. In *Proceedings of CVPR 2018*.

Zhang, C. H. and Huang, J. (2008). The sparsity and bias of the lasso selection in high-dimensional linear regression. *Annals of Statistics*, 36(4):1567–1594.

Zhang, K. W. and Bowman, S. R. (2018). Language Modeling Teaches You More Syntax than Translation Does: Lessons Learned Through Auxiliary Task Analysis. *arXiv preprint arXiv:1809.10040*.

Zhang, M., Liu, Y., Luan, H., and Sun, M. (2017). Adversarial training for unsupervised bilingual lexicon induction. In *Proceedings of ACL*, pages 1959–1970.

Zhang, R., Isola, P., and Efros, A. A. (2016a). Colorful image colorization. In *European Conference on Computer Vision*, pages 649–666. Springer.

Zhang, X., Zhao, J., and LeCun, Y. (2015). Character-level convolutional networks for text classification. In *Advances in neural information processing systems*, pages 649–657.

Zhang, Y., Gaddy, D., Barzilay, R., and Jaakkola, T. (2016b). Ten Pairs to Tag – Multilingual POS tagging via coarse mapping between embeddings. In *Proceedings of NAACL-HLT*, pages 1307–1317.

Zhang, Y., Reichart, R., Barzilay, R., and Globerson, A. (2012). Learning to Map into a Universal POS Tagset. In *Proceedings of EMNLP*.

Zhang, Y. and Yeung, D.-y. (2010). A Convex Formulation for Learning Task Relationships in Multi-Task Learning. *Uai*, pages 733–442.

Zhang, Z., Luo, P., Loy, C. C., and Tang, X. (2014). Facial Landmark Detection by Deep Multi-task Learning. In *European Conference on Computer Vision*, pages 94–108.

Zhao, H., Shi, J., Qi, X., Wang, X., and Jia, J. (2017). Pyramid scene parsing network. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 2881–2890.

Zhao, K. and Huang, L. (2017). Joint Syntacto-Discourse Parsing and the Syntacto-Discourse Treebank. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*.

Zhou, G., He, T., Wu, W., and Hu, X. T. (2015). Linking Heterogeneous Input Features with Pivots for Domain Adaptation. *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence (IJCAI 2015)*, pages 1419–1425.

Zhou, G., Xie, Z., Huang, J. X., and He, T. (2016a). Bi-Transferring Deep Neural Networks for Domain Adaptation. *Proceedings of ACL*, pages 322–332.

Zhou, J. T., Pan, S. J., Tsang, I. W., and Yan, Y. (2014). Hybrid heterogeneous transfer learning through deep learning. In *AAAI*, pages 2213–2220.

Zhou, P., Qi, Z., Zheng, S., Xu, J., Bao, H., and Xu, B. (2016b). Text classification improved by integrating bidirectional lstm with two-dimensional max pooling. In *Proceedings of COLING 2016*.

Zhou, Y. and Goldman, S. (2004). Democratic co-learning. In *Tools with Artificial Intelligence, 2004. ICTAI 2004. 16th IEEE International Conference on*, pages 594–602. IEEE.

Zhou, Y., Jin, R., and Hoi, S. (2010). Exclusive Lasso for Multi-task Feature Selection. In *Proceedings of AISTATS*.

Zhou, Z.-H. and Li, M. (2005). Tri-Training: Exploiting Unlabeled Data Using Three Classifiers. *IEEE Trans.Data Eng.*, 17(11):1529–1541.

Zhu, J.-y., Park, T., Efros, A. A., Ai, B., and Berkeley, U. C. (2017). Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks.

Zhu, X. (2005). Semi-Supervised Learning Literature Survey. Technical Report 1530, Computer Sciences, University of Wisconsin-Madison.

Zhu, X. and Goldberg, A. B. (2009). Introduction to semi-supervised learning. *Synthesis lectures on artificial intelligence and machine learning*, 3(1):1–130.

Zhuang, F., Cheng, X., Luo, P., Pan, S. J., and He, Q. (2015). Supervised Representation Learning: Transfer Learning with Deep Autoencoders. *IJCAI International Joint Conference on Artificial Intelligence*, pages 4119–4125.

Ziser, Y. and Reichart, R. (2017). Neural Structural Correspondence Learning for Domain Adaptation. In *Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL 2017)*.

Ziser, Y. and Reichart, R. (2018). Pivot Based Language Modeling for Improved Neural Domain Adaptation. In *Proceedings of NAACL-HLT 2018*, pages 1241–1251.

Zoph, B. and Knight, K. (2016). Multi-Source Neural Translation. *NAACL*, pages 30–34.

Zoph, B. and Le, Q. (2017). Neural Architecture Search with Reinforcement Learning. In *Proceedings of ICLR 2017*.

Zoph, B., Yuret, D., May, J., and Knight, K. (2016). Transfer Learning for Low-Resource Neural Machine Translation. In *Proceedings of EMNLP 2016*.

Zou, W. Y., Socher, R., Cer, D., and Manning, C. D. (2013). Bilingual word embeddings for phrase-based machine translation. In *Proceedings of EMNLP*, pages 1393–1398.