<table>
<thead>
<tr>
<th>Title</th>
<th>Leveraging Wikipedia-based features for entity relatedness and recommendation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Author(s)</td>
<td>Aggarwal, Nitish</td>
</tr>
<tr>
<td>Publication Date</td>
<td>2018-01-31</td>
</tr>
<tr>
<td>Item record</td>
<td><a href="http://hdl.handle.net/10379/7126">http://hdl.handle.net/10379/7126</a></td>
</tr>
</tbody>
</table>

Some rights reserved. For more information, please see the item record link above.
Leveraging Wikipedia-based Features for Entity Relatedness and Recommendation

Nitish Aggarwal

Supervisor:
Dr. Paul Buitelaar

This dissertation is submitted for the degree of

Doctor of Philosophy

January 2018
Abstract

Entities such as people, locations, organizations play a key role in natural language understanding. Most of the approaches that deal with natural language processing tasks, require a method to measure the relatedness between such entities. Is “Tom Cruise” more related to “Brad Pitt” than “Steve Jobs”? A human may easily provide their judgement by using common sense and their knowledge about these entities. However, a computer would require an immense amount of world knowledge to reason about the semantic relatedness between such entities. Moreover, a human has the ability of using their knowledge to understand an entity. On the other hand, a computer requires an algorithmic approach to process the background knowledge to do the same. Wikipedia is a great source to obtain background knowledge about millions of such entities. In this thesis, we introduce Wikipedia-based Distributional Semantics for Entity Relatedness (DiSER), which analyzes the semantics of an entity by its distribution in a high dimensional concept space derived from Wikipedia. DiSER measures the semantic relatedness between two entities by quantifying the distance between the corresponding high-dimensional vectors. The DiSER model is built by considering only the manually linked entities provided in a corpus such as Wikipedia. Thus, it provides an unambiguous and more accurate distributional vector for an entity comparing to existing approaches which do not distinguish between an entity and its textual surface form. We evaluate the approach on a benchmark dataset that contains relative entity relatedness scores for 420 entity pairs. DiSER improves the accuracy by more than 10% on state of the art methods for computing entity relatedness.

In order to provide a resource that can be used to obtain the related entities for a given entity, we construct a graph called Entity Relatedness Graph (EnRG), where nodes represent Wikipedia entities and the relatedness scores are represented by the edges. Wikipedia contains more than 4 million entities, which requires efficient computation of the relatedness scores between the corresponding 16 trillions of entity-pairs in a fully connected graph. We present the processing behind EnRG to efficiently compute the relatedness scores between Wikipedia entities. EnRG can be seen as an entity recom-
mendation system similar to the entity explorer provided by commercial search engines. However, most of the current approaches make use of search engine specific features such as co-occurrence information in query logs and user-click logs. Therefore, only major companies that have a large user-base and associated activities, can build entity recommendation systems with the existing approaches. However, publicly available knowledge resources such as Wikipedia can also provide an immense amount of associativity information about millions of entities. We propose Wikipedia-based Features for Entity Recommendation (WiFER) that combines different features extracted from Wikipedia and DiSER based relatedness scores. We evaluate EnRG and WiFER on a dataset of 4.5K search queries where each query has around 10 related entities tagged by human experts. We investigate the contribution of different features and compare Wikipedia-based features with the ones extracted from proprietary data like query logs and user activities.

Since DiSER provides relatedness scores between Wikipedia entities, it can be used to compute the pairwise similarity between concepts in a distributional concept space built over Wikipedia entities. On this basis, we present Non-orthogonal explicit semantic analysis (NESA) that improves over the existing text relatedness model by considering correlation between explicit concepts. We compare NESA with several WordNet-based relatedness measures and other distributional semantic models against different gold standard datasets of word and text relatedness. We perform experiments with different entity relatedness measures used in NESA, and show that NESA with DiSER outperforms state of the art approaches.

In order to demonstrate the use cases of the work presented in this thesis, we present several applications including EnRG-UI which is an entity recommendation system. EnRG-UI provides different functionalities to users for exploring related information about their favourite topics. We use DBpedia and the Yago ontology to obtain the different filters and facets which can be used to narrow down the search in EnRG-UI. Further, we present an approach to perform Medical Concept Resolution (MCR) to find the most appropriate medical concept in the Unified Medical Language System (UMLS), for a specific natural language query (e.g. a diagnosis report). To rank the concept candidate, MCR calculates relatedness scores between the context around the mention in a query and the context in UMLS. We evaluate MCR on a gold standard dataset that contains 100 medical queries annotated by human experts, and show that MCR outperforms the state of the art methods. We also present a Cross-Lingual Natural Language Querying (CroNL) approach to retrieve answers from a structured knowledge base for a natural language query in another language than that of the knowledge base. CroNL
uses a cross-lingual extension of our relatedness measure to calculate relatedness scores between terms appearing in the query and the properties in a knowledge base. We evaluate CroNL over 50 natural language queries in German. We show that our cross-lingual relatedness measure outperforms the automatic translation based methods, for cross-lingual NL-Querying over DBpedia.
Declaration

I hereby declare that except where specific reference is made to the work of others, the contents of this dissertation are original and have not been submitted in whole or in part for consideration for any other degree or qualification in this, or any other university. This dissertation is my own work and contains nothing which is the outcome of work done in collaboration with others, except as specified in the text and Acknowledgements.

Nitish Aggarwal

January 2018
Table of contents

1 Introduction 1
   1.1 Problem statement ........................................... 4
       1.1.1 Research challenges .................................. 6
       1.1.2 Scope of the thesis .................................. 7
   1.2 Research Questions ....................................... 8
   1.3 Proposed solution ........................................ 9
       1.3.1 Entity Relatedness .................................... 10
       1.3.2 Entity Recommendation ............................... 11
   1.4 Research contributions .................................... 13
   1.5 Thesis outline ............................................ 16
   1.6 Publications .............................................. 18

2 Background 21
   2.1 Entity ....................................................... 21
   2.2 Semantic similarity vs relatedness ....................... 23
   2.3 Distributional Semantics ................................ 27

3 Related work 31
   3.1 Text similarity and relatedness ............................ 31
       3.1.1 WordNet-based similarity measures .................. 32
       3.1.2 WordNet based relatedness measure ................. 34
       3.1.3 Corpus-based measures ............................... 35
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.2</td>
<td>Entity relatedness</td>
<td>39</td>
</tr>
<tr>
<td>3.2.1</td>
<td>WikiRelate</td>
<td>39</td>
</tr>
<tr>
<td>3.2.2</td>
<td>Wikipedia Link-based Measure (WLM)</td>
<td>40</td>
</tr>
<tr>
<td>3.2.3</td>
<td>Key-phrase overlap for entity relatedness (KORE)</td>
<td>41</td>
</tr>
<tr>
<td>3.2.4</td>
<td>Exclusivity in knowledge graph</td>
<td>42</td>
</tr>
<tr>
<td>3.2.5</td>
<td>Entity embeddings</td>
<td>43</td>
</tr>
<tr>
<td>3.3</td>
<td>Analysis over different dimensions of relatedness measures</td>
<td>44</td>
</tr>
<tr>
<td>3.4</td>
<td>Entity recommendation</td>
<td>45</td>
</tr>
<tr>
<td>4</td>
<td>Computing entity relatedness</td>
<td>49</td>
</tr>
<tr>
<td>4.1</td>
<td>Entity relatedness using Wikipedia</td>
<td>51</td>
</tr>
<tr>
<td>4.1.1</td>
<td>Wikipedia-based Distributional Semantics for Entity Relatedness (DiSER)</td>
<td>51</td>
</tr>
<tr>
<td>4.1.2</td>
<td>Context-based DiSER</td>
<td>53</td>
</tr>
<tr>
<td>4.1.3</td>
<td>Implementation</td>
<td>54</td>
</tr>
<tr>
<td>4.2</td>
<td>Evaluation in Ranking Related Entities</td>
<td>56</td>
</tr>
<tr>
<td>4.2.1</td>
<td>Dataset</td>
<td>56</td>
</tr>
<tr>
<td>4.2.2</td>
<td>Experiments</td>
<td>57</td>
</tr>
<tr>
<td>4.2.3</td>
<td>Results and Discussion</td>
<td>59</td>
</tr>
<tr>
<td>4.3</td>
<td>Evaluation in Entity Disambiguation</td>
<td>63</td>
</tr>
<tr>
<td>4.3.1</td>
<td>Dataset</td>
<td>64</td>
</tr>
<tr>
<td>4.3.2</td>
<td>Experiment</td>
<td>64</td>
</tr>
<tr>
<td>4.3.3</td>
<td>Results and Discussion</td>
<td>65</td>
</tr>
<tr>
<td>4.4</td>
<td>Evaluation in Top-N books recommendation</td>
<td>66</td>
</tr>
<tr>
<td>4.4.1</td>
<td>Dataset</td>
<td>67</td>
</tr>
<tr>
<td>4.4.2</td>
<td>Experiment</td>
<td>67</td>
</tr>
<tr>
<td>4.4.3</td>
<td>Results and Discussion</td>
<td>68</td>
</tr>
<tr>
<td>4.5</td>
<td>Efficiency</td>
<td>70</td>
</tr>
<tr>
<td>4.6</td>
<td>Summary</td>
<td>71</td>
</tr>
<tr>
<td>5</td>
<td>Leveraging Wikipedia Knowledge for Entity recommendation</td>
<td>73</td>
</tr>
<tr>
<td>5.1</td>
<td>Entity recommendation</td>
<td>74</td>
</tr>
<tr>
<td>5.1.1</td>
<td>Entity Relatedness Graph (EnRG)</td>
<td>74</td>
</tr>
<tr>
<td>5.1.2</td>
<td>Wikipedia Features for Entity Recommendation (WiFER)</td>
<td>75</td>
</tr>
<tr>
<td>5.1.3</td>
<td>Spark</td>
<td>77</td>
</tr>
<tr>
<td>5.1.4</td>
<td>Ranking</td>
<td>83</td>
</tr>
</tbody>
</table>
Table of contents

5.2 Evaluation ....................................................... 83
  5.2.1 Datasets .................................................. 84
  5.2.2 Experiments .............................................. 85
  5.2.3 Result and Discussion .................................... 87
5.3 Summary ...................................................... 93

6 Using entity relatedness for non-orthogonal explicit semantic analysis 95
  6.1 Non-Orthogonal Explicit Semantic Analysis ....................... 97
  6.2 Computing Concept Relatedness ................................. 98
    6.2.1 VSM-Text .............................................. 98
    6.2.2 VSM-Anchors .......................................... 98
    6.2.3 ESA-WikiTitle .......................................... 99
    6.2.4 DiSER ................................................ 99
  6.3 Evaluation .................................................. 99
    6.3.1 Word Relatedness Dataset ............................... 100
    6.3.2 Text Relatedness Dataset ................................ 101
    6.3.3 Experiment ............................................. 101
    6.3.4 Results and Discussion ................................ 102
  6.4 Summary .................................................... 106

7 Application ..................................................... 109
  7.1 EnRG: Entity relatedness graph .................................. 109
    7.1.1 Web User Interface ..................................... 110
  7.2 Medical concept recognition .................................... 115
    7.2.1 Approach ............................................... 117
    7.2.2 Evaluation ............................................. 119
    7.2.3 Results and Discussion ................................ 120
  7.3 Cross-lingual natural language querying ......................... 120
    7.3.1 Related work ........................................... 121
    7.3.2 Approach ................................................ 122
    7.3.3 Evaluation ............................................. 125
  7.4 Summary .................................................... 130

8 Conclusion ...................................................... 133
  8.1 Summary .................................................... 133
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>8.2 Lessons Learned</td>
<td>136</td>
</tr>
<tr>
<td>8.3 Directions for Future Research</td>
<td>137</td>
</tr>
<tr>
<td>8.3.1 Relationship Explanation</td>
<td>138</td>
</tr>
<tr>
<td>8.3.2 Entity Similarity vs Relatedness</td>
<td>138</td>
</tr>
<tr>
<td>References</td>
<td>141</td>
</tr>
</tbody>
</table>
List of figures

1.1 Thesis technical overview ................................................. 13
1.2 Research contributions overview ...................................... 16

2.1 A snapshot of WordNet 3.1 for word “car” .............................. 24
2.2 Similarity and Relatedness .................................................. 25
2.3 The usage of word “bardiwac” in different sentences ................. 28
2.4 Vector representation of different words in three dimensional space 28
2.5 Word distribution in different documents ............................... 29
2.6 Co-occurrence based vector representation of different words in three dimensional space .............................................. 29

3.1 Explicit Semantic Analysis .................................................... 37

4.1 Wikipedia pre-processing for entity tagged Wikipedia corpus ....... 55
4.2 F1 scores of different systems submitted to ESWC books recommendation challenge ..................................................... 69

7.1 Related entities to Brad Pitt .................................................. 111
7.2 Ranked list of american vegans related to Brad Pitt .................. 112
7.3 Ranked list of films related to Brad Pitt .................................. 113
7.4 Hardware companies related to Apple Inc. .............................. 114
7.5 Educational institutions established in 1845, related to NUIG .......... 114
7.6 Query interpretation pipeline for the German NL-Query “Mit wem is die Tochter von Bill Clinton verheiratet?” (“Who is the daughter of Bill Clinton married to?”@en) .............................................. 124
List of tables

1.1 Top 5 Wikipedia concepts for entity “NeXT” .......................... 11
2.1 Number of instances per classes in DBpedia .......................... 23
2.2 Co-occurrence matrix ...................................................... 30
3.1 Characteristics of relatedness measures ............................... 45
4.1 10 most relevant Wikipedia concepts for entity “NeXT” by ESA and DiSER .................................................. 53
4.2 Example seed entity and its related entity candidates ............... 57
4.3 Spearman rank correlation of relatedness measures with gold standard .................................................. 60
4.4 Spearman rank correlation of relatedness measures with gold standard .................................................. 61
4.5 Entity Disambiguation accuracy on KORE50 dataset ............... 66
4.6 Top N books recommendation ............................................. 67
4.7 Results of efficiency experiments ....................................... 70
5.1 Dataset details ............................................................... 85
5.2 Retrieval performance of unsupervised entity recommendation . 86
5.3 Retrieval performance of supervised entity recommendation on all entity types ........................................ 86
5.4 Retrieval performance of supervised entity recommendation on the person entity types .................................... 87
5.5 Retrieval performance of supervised entity recommendation on the location entity types .................................... 87
5.6 Top 15 features sorted by rank according to their importance in WiFER for all types ........................................ 88
5.7 Top 15 features sorted by rank according to their importance in WiFER for person types ........................................... 89
5.8 Top 15 features sorted by rank according to their importance in WiFER for location types ........................................... 89
5.9 Top 20 features sorted by rank according to their importance in SparkWiki for all types ........................................... 90
5.10 Top 20 features sorted by rank according to their importance in SparkWiki for person types ........................................... 91
5.11 Top 20 features sorted by rank according to their importance in SparkWiki for location types ........................................... 92
5.12 Retrieval performance per feature type ........................................... 92
5.13 Retrieval performance per data source ........................................... 93

6.1 Top 5 Wikipedia concepts for “football” and “soccer” in the ESA vector ........................................... 96
6.2 Spearman rank correlation of relatedness measures with word relatedness gold standard datasets ........................................... 102
6.3 Analysis over synonyms ........................................... 104
6.4 Analysis over hyponyms ........................................... 104
6.5 Analysis over word phrases ........................................... 105
6.6 Analysis over implicit relations ........................................... 106
6.7 Spearman rank correlation of relatedness measures with text relatedness gold standard datasets ........................................... 106

7.1 Results of Medical Concept Resolution on human annotated clinical factors ........................................... 120
7.2 Query categorization of training and test dataset ........................................... 126
7.3 Error type and its distribution over 50 natural language queries and 28 selected natural language queries in German ........................................... 127
7.4 Evaluation on 28 German NL-Queries ........................................... 128
7.5 Evaluation on 50 German NL-Queries ........................................... 130
Chapter 1

Introduction

The rapid growth of large knowledge repositories like Wikipedia and its derived structured knowledge bases offer a wealth of semantic information about millions of real world entities. This knowledge can be used to semantically enrich the current Web [32, 99]. Most of the approaches that deal with natural language understanding (NLU) [16, 143] and its related fields, make use of this knowledge to obtain the background information about entities appearing in the text. For instance, Gabrilovich and Markovitch [53] make use of Wikipedia knowledge for semantic interpretation of a natural language text by building a distributional vector over Wikipedia article topics. Strube and Ponzetto [134] use the Wikipedia hyperlink graph to represent the semantics of a word. Hassan and Mihalcea [60] utilize the neighboring entities appearing in Wikipedia article to build a semantic interpreter. Moreover, Wikipedia knowledge has been utilized for different entity oriented tasks such as retrieving answers for open domain questions [45], user interests profiling [80], and mining domain knowledge [42]. The recently popular IBM Watson question answering system [44] uses Wikipedia to retrieve entities as answer candidates. Chu-Carroll and Fan [27] showed that 85% of the questions in the TREC question answering challenge [140] can be answered by using only Wikipedia data. This shows a wide coverage of real world entities in Wikipedia.

The derived structured knowledge bases such as DBpedia [13], Freebase [24], and YAGO [136] provide additional background knowledge about Wikipedia en-
intities, such as semantic types of these entities and predefined relations between them. This additional information has been used in various tasks like answer type evidence scoring in question answering [79], labeling the topics in topic modeling [65] and in query understanding [119]. Kalyanpur et al. [79] make use of YAGO types information to match the answer candidate type with the relevant answer type for performing answer type evidence scoring in IBM Watson. Hulpus et al. [65] utilize the DBpedia graph to find the most appropriate label for a topic in topic modeling. Pound et al. [119] showed that query understanding can derive benefit from using this structured knowledge. Moreover, the major search engine companies have started to build their own structured knowledge bases to semantically enrich the indexed articles to perform an intelligent search. For instance, Google and Yahoo! are building their own knowledge graphs [38] to improve classical document retrieval. Microsoft is building a knowledge base named “Satori” to improve their local (business oriented entities) search [95].

In particular, entities like people, locations, organizations etc. play a key role in building a semantic interpreter of natural language text. Most of the approaches that deal with natural language interpretation and understanding, require a method to measure the relatedness between two entities. The significance of measuring semantic relatedness between entities has been shown in various subtasks of NLU, such as entity linking [62, 63, 83, 110], entity recommendation [20, 150], query expansion [4, 107], knowledge base population [39, 74], and semantic search [22, 33]. For instance, entity disambiguation [62, 63, 83] mainly relies on the quality of entity relatedness measures to jointly map the entity mentions\(^1\) with their defining entities available in knowledge bases. The measures are used to obtain related entities to perform query expansion [4, 107], knowledge base population [39, 74] and semantic search [22, 33].

In order to compute the relatedness score between two entities, semantic representation of the entities is required. The semantic representation of an entity can be generated by using available background knowledge. It is very intuitive

\(^1\)Entity mention is a span of text that refers to an entity in knowledge base, e.g. in a sentence “Portugal won the Euro cup”, the mention Portugal refers to “Portugal national football team”.
to adapt existing methods of text relatedness for computing entity relatedness scores. Some of the existing approaches [52, 85] calculate the semantic relatedness between two natural language texts by using words distribution in a large text corpus like Wikipedia. However, they are limited to perform only at surface level of the entities. Due to this limitation, these methods may produce an inappropriate representation for ambiguous surface forms like “apple” or “next”. For instance, if we retrieve Wikipedia articles that contain the term “next”, we will get articles like *Linked List, Railway Station, Train Schedule*. However, if we retrieve the articles which only contain “next” as Wikipedia anchor and link to the entity “NeXT”, we will get articles like *Music Kit, NeXT, NeXT Computer*. Therefore, it is important to consider the appropriate sense of a surface form to generate the semantic representation of an entity.

We define following terminologies which will be used in this thesis.

**Definition 1.** Wikipedia concept: can be defined as a mental image that corresponds to some distinct entity or class of entities, which helps in identifying its associated features and characteristics. For example, “Apple Inc.” is an entity and “Sport” is a class of different type of sports like football and cricket.

**Definition 2.** Wikipedia article: contains textual description to define a concept, name of the concept as its title, and all outgoing links to other concepts.

**Definition 3.** Anchor: is a span of text that links to the corresponding Wikipedia concept.

**Definition 4.** Surface form: is a textual representation of an entity name. One surface form may refer to multiple entities, e.g. “apple” can refer to Apple Inc., apple fruit, apple (surname), and many more.

**Definition 5.** Wikipedia hyperlink graph: is a graph where every node represents a Wikipedia concept, and their outgoing and incoming links are represented by its edges.

As we described above, Wikipedia provides background information about millions of real world entities which has been shown to be beneficial in various NLU
oriented tasks. Similarly, existing entity relatedness measures also make use of Wikipedia knowledge to calculate the relatedness scores between two Wikipedia concepts. Witten and Milne [144] calculate the relatedness scores by counting the shared incoming links in the articles corresponding to given Wikipedia concepts. Similarly, Hoffart et al. [62] uses the shared outgoing links and important key-phrases in the articles of interest. Wikipedia provides background knowledge besides the hyperlink graph which is not utilized by existing measures. For instance, it provides a category structure, implicit importance of outgoing and incoming links for an article, co-occurring anchors, and popularity of entities. These features can be used to develop an entity relatedness measure. In this thesis, we introduce a novel method to compute the relatedness scores between entities using Wikipedia knowledge. In particular, our approach relies on co-occurrence information of anchors and uses associative weights to represent the importance of an anchor for a given Wikipedia entity. We present an entity recommendation approach that uses several features extracted from Wikipedia along with our entity relatedness measure. Moreover, we introduce a new model “Non-Orthogonal Explicit Semantic Analysis (NESA)” for computing text relatedness that considers the correlation between topics by using our entity relatedness measure.

1.1 Problem statement

Computing semantic relatedness is used for quantifying the strength of a relation between two text units. These units can be two words, sentences, documents or entities. There have been several efforts [23, 52, 85] in building models to compute relatedness scores between natural language texts. However, the problem of measuring relatedness between two entities is relatively new. Due to the applicability of the entity relatedness measures, it has received high attention in recent years [20, 32, 62]. In particular, applications like entity recommendation in web search mainly rely on the quality of entity relatedness measures [20, 150]. Therefore, in this thesis we focus on developing a novel entity relatedness measure and using it for entity recommendation.
Entity relatedness can be defined as computing the relatedness score between two given entities $e_1$ and $e_2$. We consider every Wikipedia concept as an entity. However, the notion of “entity” can have different interpretations in different communities. For instance, standard natural language processing (NLP) tools mainly work with named entities such as people, organizations and locations, whereas domain specific applications deal with concept level definitions such as names of symptoms or side effects in the bio-medical domain. We discuss this in detail in Chapter 2. A relatedness score represents the strength of a relation between entities $e_1$ and $e_2$. However, as it is difficult to obtain an absolute strength, we compute the relative relatedness score. For instance, if we ask a human to score the relatedness between two entities “Tom Cruise” and “Brad Pitt”, it will be hard to get consistent judgement. However, humans can easily judge if “Tom Cruise” is more related to “Brad Pitt” than “Steve Jobs”. Moreover, most applications require relative scores to rank related entities with respect to a given one.

Entity recommendation is the task of retrieving a ranked list of entities that are most associated with the given entity. Major search engines are incorporating entity recommendation in their standard web search platforms to increase user engagement. Entity recommendation in web search can be defined by recommending a list of entities that are most associated to the search query provided by a user. Most of the approaches make use of search engine specific features such as co-occurrence information in query logs and user-click logs. Therefore, only major companies that have a large user-base and associated activities, can build entity recommendation systems with the existing approaches. However, publicly available knowledge resources like Wikipedia can also provide an immense amount of associativity information about millions of entities. Therefore, we focus on developing an entity recommendation approach by using only Wikipedia.
1.1.1 Research challenges

Is “Tom Cruise” more related to “Brad Pitt” than “Steve Jobs”? A human may easily estimate it by using common sense and their knowledge about these entities. However, a computer requires an immense amount of world knowledge to reason about the semantic relatedness of these entities. Further, a human has the ability of using their knowledge to understand an entity. On the other hand, a computer requires an algorithmic approach to process the background information for extracting the related entities. Wikipedia provides a large amount of such background information in several different ways such as plain unstructured text, anchor text, and structured knowledge extracted from information boxes in the form of DBpedia [13]. However, it is not trivial to build a model that can process this information like humans do by learning from their experiences in the world. Therefore, a major challenge is to build a model that can reason about the relatedness between two entities by considering the semantics associated with them.

With the advent of large structured knowledge bases like DBpedia [13] and Freebase [24], it is very intuitive to build an entity recommendation system by obtaining all the neighboring entities of a given entity. However, most of the entities can easily have more than 1,000 neighboring entities and to retrieve a ranked list of related entities, it requires some ranking method. For instance, popular entities like “Berlin” can have more than 35,000 directly connected neighboring entities in the DBpedia graph. Further, such knowledge bases will not cover every relation, for example, “Tom Cruise” and “Brad Pitt” are not connected directly in the DBpedia graph as they do not have a specific relation between them. Moreover, in order to find all the related entities, we can not only rely on directly connected entities in the knowledge graphs. Therefore, the major challenge is to build a system that can retrieve related entities beyond their direct connections in the knowledge graphs, and rank them according to their appropriate relatedness strength with the given entity.
1.1 Problem statement

1.1.2 Scope of the thesis

Several efforts have been made to compute semantic relatedness scores between two entities. However, it is very difficult even to get a high agreement between human annotators about the relatedness of two entities, as each annotator have their own perspective. For instance, someone may state that “Steve Jobs” is more related to “Apple Inc.” than to his wife “Laurene Powell Jobs”, on which others may disagree. It depends upon what relation is considered more important by an individual. If a user knows “Steve Jobs” as an entrepreneur and inventor, then they might consider the “co-founder” relation more important than “spouse” or vice-versa. This calls for personalization which is out of the scope of this thesis. We focus instead on a global view to make decisions about relatedness and ranking.

Previous studies have shown that the recommendation of related entities is useful to increase user engagement in web search [20, 150]. More than 50% of web search queries pivot around a single entity [120]. Therefore, entity recommendation in web search can be defined by retrieving a list of entities that is related to the entities appearing in a search query. However, this task leads to a discussion of a broad area of semantic search that concerns the retrieval of relevant information from structured and unstructured data for a given search query. The Semantic Search Challenge\(^2\) organized an entity retrieval task where the query was focused on a single target entity. However, this task becomes more challenging if a user provides the description of target entities. Therefore, the semantic search task has two major steps: the interpretation of query structure and entity retrieval for the interpreted query. In order to address this complex task, TREC [14] has organized a shared tasks: Related Entity Finding (REF).

“REF: Given an input entity, by its name and homepage, the type of the target entity, as well as the nature of their relation, described in free text, find related entities that are of target type, standing in the required relation to the input entity.”

The REF task involves finding related entities for a given type where the type

\(^2\)http://km.aifb.kit.edu/ws/semsearch11/
needs to be interpreted from the provided free text. However, we limit our work to the entity recommendation task that is retrieving a list of entities related to a given entity query, and we do not deal with the extraction of their types from free text.

1.2 Research Questions

- **RQ 1.** Can we create an algorithmic process to utilize the background information about entities to compute their relatedness?

Wikipedia contains background knowledge about millions of entities. However, this knowledge is not explicitly available to obtain the semantics of an entity. For instance, an entity in Wikipedia has outgoing links, incoming links, categories, redirected links, textual description, and information box properties. Therefore, investigation is required to determine which form of knowledge can be used to obtain the semantics of entities. While the type of knowledge plays a key role in building entity relatedness, it is also important to effectively exploit the identified knowledge to calculate relatedness between entities.

- **RQ 2.** Can we build an entity recommendation using publicly available dataset like Wikipedia?

Motivated by huge demand of entity recommendation in lack of expensive users’ activity datasets, it is required to analyze if an entity recommendation approach can be developed by using publicly available dataset like Wikipedia. As Wikipedia consists of different type of information that can be utilized in building an entity recommendation, it is important to identified effective Wikipedia-based features to build the system. However, a supervised model requires a great amount of manually labelled data specific to target domain, for training a machine learning algorithm. It can be quite expensive to create a training dataset for different applications, therefore, it is worthwhile to investigate if an unsupervised approach to
1.3 Proposed solution

entity recommendation can be formulated.

• **RQ 3.** *Can we improve the Explicit Semantic Analysis to compute text relatedness?*

Explicit Semantic Analysis (ESA) is one of the top performing methods for computing relatedness between natural language texts including word and sentence level relatedness. Therefore, in recent years, some modifications have been proposed to improve ESA. ESA model assumes that the topics are orthogonal and not related to each other, which is one of the major issues with its formalism. Therefore, investigation is needed to further explore this orthogonality issue and to extend ESA model by nulling this assumption. In order to do that, we need to formulate an effective method to compute the topic relatedness, and explore how to exploit it to incorporate the co-relatedness between the topics in ESA model.

### 1.3 Proposed solution

This thesis focuses on quantifying the relatedness between entities and obtaining a ranked list of related entities to build an entity recommendation system. In order to obtain a ranked list of related entities, we explore several features from Wikipedia. Other existing methods [20, 150] make use of proprietary data such as query logs and user-click logs. Instead, we focus on utilizing only publicly available datasets such as Wikipedia to model entity recommendation. The entity recommendation task requires two major steps: finding all the candidates that can be considered related to the given entity, and ranking them to obtain the top N most relevant ones. To rank the candidates, we require a measure for computing the relatedness scores between each candidate and the given entity. However, the entity relatedness measure is a fundamental requirement to many other tasks that we mentioned above. Consequently, our primary focus is to develop an entity relatedness measure that utilizes the hyperlink graph and article content from Wikipedia as background knowledge.
1.3.1 Entity Relatedness

Reasoning about the semantic relatedness of “Brat Pitt” and “Tom Cruise” requires an immense amount of world knowledge about the entities represented by these two surface forms. In the case of popular and unambiguous surface forms like “Brad Pitt” and “Tom Cruise”, it might be easier to manually judge the relatedness score between them. However, this task becomes more challenging if the entities contain very ambiguous surface forms like “apple” and “next”. The semantics of “apple” may refer to a fruit, a person’s surname or a company. Similarly, “next” refers to more than 20 different entities on Wikipedia but the most common meaning of “next” that generally comes first in mind is “succeeding item”. Thus it is hard to assess the relatedness between “apple” and “next”. However, if we are given explicitly that both “apple” and “next” are software companies founded by Steve Jobs, we can manually judge their appropriate relatedness.

The semantics of an entity can be inferred from its distribution in a high dimensional space of concepts derived from Wikipedia, as it is a constantly growing encyclopaedia, containing world knowledge about millions of entities. Thus the semantic meaning of an entity can be obtained from its usage in Wikipedia, in the form of a high dimensional distributional vector [59] over Wikipedia concepts by taking every concept as a dimension. The associativity weight of an entity with a concept can be taken as the magnitude of the corresponding dimension in the vector. The associativity strength can be computed with different scoring schemes such as Term Frequency (TF), Pointwise Mutual Information (PMI), Term Frequency with Inverse Document Frequency (TF-IDF), Language Model (LM) and others. To obtain the semantic relatedness between two entities, we can simply quantify the distance between their distributional vectors. In order to retrieve distributional vectors of an entity, several associativity weighing schemes can be used. For instance, the weight can be term frequency (TF) which states that the total number of occurrences of an entity reflects its associativity.

1.3 Proposed solution

<table>
<thead>
<tr>
<th>NeXT as text</th>
<th>NeXT as anchor</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Doubly linked list</td>
<td>Music Kit</td>
</tr>
<tr>
<td>2. Gare de Rennes</td>
<td>NeXTSTEP</td>
</tr>
<tr>
<td>3. Brugge railway station</td>
<td>NeXT Laser Printer</td>
</tr>
<tr>
<td>4. Gare d'Avignon Centre</td>
<td>NeXT Computer</td>
</tr>
<tr>
<td>5. Gare de Toulon</td>
<td>Shelf (computing)</td>
</tr>
</tbody>
</table>

Table 1.1 Top 5 Wikipedia concepts for entity “NeXT”

strength. To investigate our intuition about using only hyperlinks in building the distributional vector, we compare the top 5 Wikipedia articles retrieved for “NeXT” as a hyperlink and as plain text. Table 1.1 shows that articles retrieved for “NeXT” as plain text are not relevant, however, all the articles obtained for “NeXT” as hyperlink are relevant to the entity. These articles are ranked according to the term frequency of “NeXT”. It allows us to investigate the distributional representation of an entity in further detail.

In particular, our approach builds a distributional vector by using Wikipedia to represent the semantics of an entity. Hence, we call it Wikipedia-based Distributional Semantics for Entity Relatedness (DiSER). Table 1.1 shows that building the vector by considering only the hyperlinks may benefit our approach. However, it has a limitation that a vector can be created only if an entity has a Wikipedia page. To overcome this issue, we also present an extension of “DiSER” called Context-DiSER that can build a distributional vector of an entity that does not have a Wikipedia page. Context-DiSER builds the distributional vector of the context of an entity where context can be obtained from textual descriptions found in any external resources like news articles and websites.

1.3.2 Entity Recommendation

Entity recommendation is the task of obtaining a ranked list of entities that are related to a given entity. As we mentioned in section 1.1.1, an obvious way to obtain all the related entities can be to retrieve all the neighbors of a given entity in knowledge graphs. However, the major challenge is to develop a method to rank these neighbors for obtaining a smaller set of most related ones. Moreover,
we can not rely on neighbors only as they may not cover all the potential candidates that can actually be considered related. Therefore, to cover the relations beyond knowledge graphs, we assume that every Wikipedia entity can be related to any other Wikipedia entity. Thus, we compute relatedness scores between all Wikipedia entities, by considering every Wikipedia entity as a potential candidate of a given entity. The English Wikipedia\(^5\) corpus consists of more than 4 million entities, therefore every entity has more than 4 million candidates. We use DiSER to compute relatedness scores. In order to obtain a ranked list for all the Wikipedia entities, we compute more than 16 trillion (4 million x 4 million) relatedness scores. Finally, we obtain a very big graph which we call EnRG (Entity Relatedness Graph). It contains the ranked lists of related entities for every Wikipedia entity. EnRG can be used directly to explore entity recommendations. However, a more sophisticated technique of getting recommendations can be achieved by using training data that reflects user-search behavior. Particular tasks like web search assistance require a model which can obtain the recommendations that users will click most likely as their next search query. Therefore, we build a model that uses many different features and train over a dataset that contains more than 40K recommendation instances in web search [20]. All the features are extracted from Wikipedia as our goal is to develop a method by using only publicly available datasets. Most of the features are utilized by existing commercial systems for entity recommendation. However, the datasets used to obtain feature values are not publicly available. Moreover, our intuition is that a distributional model (DiSER) may improve the accuracy significantly. We use a learning to rank method to combine these features through training.

Figure 1.1 illustrates the overall process of our proposed solutions. Although, Wikipedia consists of annotated entities, it does not contain all the potential mentions linked with entities. Therefore, we tag the potential mentions with entities in all the Wikipedia articles. The distributional vectors of entities are built

\(^5\)We use snapshot of English Wikipedia from Oct 1\(^{st}\), 2013. This snapshot consists of 13,872,614 total articles, in which 5,659,383 are Wikipedia redirects, 3,571,206 namespace pages. We obtained 4,102,442 articles (4.1M) after removing all the Wikipedia redirects, namespaces and short articles.
over these processed Wikipedia articles that we refer as Entity tagged Wikipedia. Finally, DiSER retrieves the entity vectors and compute the relatedness scores based on vector cosine. Context-DiSER obtains the Wikipedia entities appear in context of a given entity and builds the corresponding DiSER vector. We pre-compute DiSER scores between all the Wikipedia entities to generate Entity Relatedness Graph (EnRG) [2] which can be used to get a ranked list of related entities for a given entity. In order to build entity recommendation, we extract text-based features from Wikipedia and entity based features from our Entity tagged Wikipedia. The features are combined by using machine learning method trained on a manually prepared dataset of entity recommendation. Finally, we use the DiSER to compute the relatedness between explicit concepts to improve the text relatedness in explicit concept space.

Fig. 1.1 Thesis technical overview

1.4 Research contributions

This thesis explores different techniques to exploit Wikipedia knowledge in developing a novel method for entity relatedness and recommendation. We de-
velop DiSER that builds on the distributional hypothesis [59] which states that “two co-occurring text units tend to have similar and related meaning”. The developed relatedness measure DiSER is used to build an entity relatedness graph (EnRG) that provides the ranked list of related entities. Further, we combine DiSER with other features extracted from Wikipedia to build an entity recommendation method specifically for the web search scenario. We evaluated DiSER in different tasks like entity ranking, entity linking and top N recommendation. Moreover, in order to measure the impact of our entity relatedness measure, we also present an improved text relatedness model that takes benefit from DiSER. In particular, this thesis includes three main contributions as follows:

1. **Wikipedia based Entity Relatedness** that represents the semantics of an entity by using its distribution over Wikipedia.

   - An entity relatedness measure called “DiSER” that builds high dimensional vectors of entities to calculate their relatedness score. Although DiSER is inspired by an existing model of computing text relatedness called Explicit Semantic Analysis (ESA) [52], it focuses on the semantic representation of an entity rather than free text. The main differences and a comprehensive comparison is presented in Chapter 4.

   - DiSER is limited to generate the distributional vectors for entities that have Wikipedia pages. Although the English Wikipedia has more than 4 million entities, it will be incomplete and may not contain for instance many of the entities that are relatively new or recently became popular. Therefore, to overcome this issue, we present an alternative approach called **Context-DiSER** that builds the DiSER vector for an entity which does not have a Wikipedia article, by using the entities available in the context. The context of an entity can be obtained from any external resource that provides some textual description about that entity.

   - Evaluation of the proposed entity relatedness measure in ranking related entities, entity linking, and top N book recommendation is presented in Chapter 4.
2. **Entity Recommendation** that uses several features extracted from Wikipedia along with DiSER.

   - A very big graph called **Entity Relatedness Graph (EnRG)** is built by using “DiSER”. EnRG provides a ranked list of related entities. Similar to major search engines, EnRG recommends related entities for a given entity that provides an opportunity to users to explore about things related to their favorite topics.

   - DiSER is combined with other Wikipedia based features by using a learning to rank method to build an **entity recommendation approach** called “Wikipedia Features for Entity Recommendation (WiFER)”. All the features are investigated in detail to provide a deep insight in their characteristics.

   - Evaluation of our approach for entity recommendation is by comparing DiSER and other Wikipedia-based features with a commercial entity recommendation system which uses several features from proprietary data sources like query logs and search sessions. A comprehensive study of the importance of different features is provided in chapter 5.

3. **Non-orthogonal explicit semantic analysis (NESA)** that improves over the existing text relatedness method ESA by considering correlation between explicit topics.

   - EnRG provides precomputed relatedness scores between all Wikipedia entities. These scores can be used to improve existing text relatedness measures that use Wikipedia to build a concept space and assumes that **the topics are orthogonal**. For instance, ESA builds an explicit concept space and does not consider the correlation between explicit concepts. Most of the ESA implementations use Wikipedia to build an explicit concept space, therefore, we investigate how to overcome topic orthogonality using “DiSER”, and its effect on text relatedness. We present a model called **“Non-orthogonal explicit concept space (NESA)”** that considers the relatedness between explicit topics
We perform several experiments with different gold standard datasets, and compare the results with other existing methods in Chapter 6.

1.5 Thesis outline

The remainder of the thesis is organized as follows. Chapter 2 provides the background of this thesis, including a discussion of different interpretations of an entity in different tasks and communities. It also describes the basics of distributional semantics which is at the core of the work presented in this thesis. Further, it presents different aspects of semantic similarity and relatedness. Chapter 3 presents a survey of related work on semantic relatedness measures followed by the specific work on relatedness between entities. We then discuss the existing work on entity recommendation, focusing on the approaches that fall in the same category as ours.

The next two chapters are the core contributions of this thesis. Chapter 4
introduces a novel entity relatedness measure “DiSER”. It describes the pipeline and implementation of our approach that is used to build a distributional semantic space over Wikipedia, and computes the relatedness score between two entities by taking the vector cosine between their corresponding high dimensional vectors. We present several evaluations of DiSER by applying it in entity ranking, entity disambiguation and top-N books recommendation. In chapter 5, we present two approaches for entity recommendation i.e. “EnRG” and “WiFER”. It explains the process to build EnRG and the machine learning based method to combine several features extracted from Wikipedia, used in WiFER. We present an analysis of the individual contribution of different features and compare them with the features used in a commercial entity recommendation system. Chapter 6 presents “NESA” that considers the correlation between explicit concepts in a distributional concept space. We discuss the evaluation of NESA on several gold standard datasets of word and text relatedness. Figure 1.2 illustrates the use of Wikipedia in the research contributions discussed in this thesis. It shows that we build a distributional representation of entities to compute their relatedness, which is used to compute the entity recommendations and to develop NESA. Chapter 7 presents “EnRG-UI” and other applications of the techniques proposed in this thesis. EnRG-UI is a web user interface that provides different functionalities to explore related entities through EnRG graph. We describe dynamic set of filters and facets for exploring the related entities in EnRG-UI. Further, we describe “Medical Concept Resolution (MCR)” that makes use of the similarity measure presented in Chapter 6. We describe our approach of finding the most appropriate medical concepts for a given medical text span using text similarity based candidate ranking. We evaluate MCR on a gold standard dataset that consists of 100 medical questions annotated by medical experts. We discuss the effect of using different contexts and relatedness scores in MCR. Chapter 7 also presents “Cross-lingual natural language querying (CroNL)” that retrieves answers from English DBpedia for natural language queries in German. CroNL uses a cross-lingual extension of the relatedness measure presented in Chapter 6. We present a comparison of cross-lingual relatedness measures and automatic translation for performing cross-lingual natural language querying over DBpedia.
Finally, we summarize the main contributions of this thesis and present future directions of research in Chapter 8.

1.6 Publications

We have published the following multiple research papers on above discussed contributions:

- **Connecting the Dots: Explaining Relationships Between Unconnected Entities in a Knowledge Graph**
  Nitish Aggarwal, Sumit Bhatia, Vinith Misra
  In Proceeding of the 13th European Semantic Web Conference (ESWC), Crete, Greece, Jun, 2016

- **Medical Concept Resolution**
  Nitish Aggarwal, Ken Barker, Chris Welty
  In Proceeding of the 14th International Semantic Web Conference (ISWC), Pennsylvania, United States, Oct, 2015 *(Best poster award)*

- **Top-N Books Recommendation Using Wikipedia**
  Nitish Aggarwal, Kartik Asooja, Jyoti Jha, Paul Buitelaar
  In Proceeding of the 14th International Semantic Web Conference (ISWC), Pennsylvania, United States, Oct, 2015

- **Insights into Entity Recommendation in Web Search**
  Nitish Aggarwal, Peter Mika, Roi Blanco, Paul Buitelaar

- **Leveraging Wikipedia Knowledge for Entity Recommendations**
  Nitish Aggarwal, Peter Mika, Roi Blanco, Paul Buitelaar
  In Proceeding of the 14th International Semantic Web Conference (ISWC), Pennsylvania, United States, Oct, 2015
• **Who are the American Vegans related to Brad Pitt? Exploring Related Entities**  
Nitish Aggarwal, Kartik Asooja, Housam Ziad, Paul Buitelaar  

• **Non-Orthogonal Explicit Semantic Analysis**  
Nitish Aggarwal, Kartik Asooja, Georgeta Bordea, Paul Buitelaar  

• **Wikipedia-based Distributional Semantics for Entity Relatedness**  
Nitish Aggarwal, Paul Buitelaar  
In Proceeding of the Association for the Advancement of Artificial Intelligence (AAAI) Fall Symposium, Washington, DC, United States, Nov, 2014.

• **Is Brad Pitt Related to Backstreet Boys? Exploring Related Entities**  
Nitish Aggarwal, Kartik Asooja, Paul Buitelaar, Gabriela Vulcu  

• **Exploring ESA to Improve Word Relatedness**  
Nitish Aggarwal, Kartik Asooja, Paul Buitelaar  

• **Using Distributional Semantics to Trace Influence and Imitation in Romantic Orientalist Poetry**  
Nitish Aggarwal, Justin Tonra, Paul Buitelaar  

• **Cross-Lingual Natural Language Querying over the Web of Data**  
Nitish Aggarwal, Tamara Polajnar, Paul Buitelaar

• **Improving ESA with Document Similarity**  
  Tamara Polajnar, Nitish Aggarwal, Kartik Asooja, Paul Buitelaar  
  In Proceedings of the 35th European Conference on Information Retrieval (ECIR), Moscow, Russia, Mar, 2013.

• **Cross Lingual Semantic Search by Improving Semantic Similarity and Relatedness Measures**  
  Nitish Aggarwal  

• **Query Expansion using Wikipedia and DBpedia**  
  Nitish Aggarwal and Paul Buitelaar  
  In The Cultural Heritage in CLEF (CHiC), Rome, Italy, 2012.

• **A System Description of Natural Language Query over DBpedia**  
  Nitish Aggarwal and Paul Buitelaar  
  In ILD 2012 at 9th Extended Semantic Web Conference (ESWC), Crete, Greece. *(Best system)*
Chapter 2

Background

2.1 Entity

The Oxford dictionary defines an entity by “A thing with distinct and independent existence”. Moreover, an entity can be defined by an individual thing that belongs to a class or category. For instance, anything that refers to a type of person, location, organization, event, movie, music band, field of study and many more, can be considered as an entity. Although, most of the previous work [75, 138] focuses on named entities, some recent studies [82, 98] define the more generic scenarios beyond the named entities. The term “named entities” is coined by message understanding conference (MUC) [56] in 1995, for Named Entity Recognition (NER) task. It referred to a thing that belongs to predefined classes i.e. person, location, organization, time, and quantities. However, the research community is not stuck to this definition and it has been refined by the time. Therefore, several different definitions have been proposed for named entities. Standard NER [37, 46] community focuses on three types of entity: person, location, organization. However, the domain specific applications require the recognition of entities [82] beyond these three standard classes. For instance, biomedical domain defines the task of entity recognition with identification of anatomy, drugs, disorders, decease and others. Although, these categories can also be considered as named entities, sometimes domain specific applications require to define the task beyond the named entities recognition. For example,
Background

symptoms identification in biomedical domain [Perera et al.] and legal acts detection in law domain need to deal with recognition of text phrases such as “pain after eating”, “nose bleeding during nights”, and “Time value of money”, which can not be considered as named entities. Moreover, the identification of these domain specific terms deal with domain entities and concepts.

Due to publicly available resource like Wikipedia, the usage of background knowledge receives high attention in natural language text interpretation and understanding. Wikipedia is an encyclopedia that contains the background definition about millions of real world entities. These definitions are collected by collaborative efforts of many Wikipedia volunteers. Further, it consists of hyperlinks to describe the Wikipedia entities with the help of other related entities. This knowledge allows us to build an entity disambiguation system that provides the appropriate definition links in addition to the recognized entities using NER. Bunescu and Pasca [26] introduced the task of Named Entity Disambiguation (NED) where the potential entity candidates are linked to Wikipedia. Since, many Wikipedia entries can not be considered named entities, the task of linking of a potential candidate to Wikipedia called “Wikification” by Mihalcea and Csomai [98]. Moreover, Wikipedia contains very generic things that may not fall under the category of entities. For instance, things like “Sport” and “Politics” can be considered as the instances of a class “Activity”, however, they refer to the classes themselves. Therefore, Wikification task defines the entities by including all the entries on Wikipedia without further distinction between entity, class and others.

Semantic Web community derives the standards to extract the structured data from unstructured data sources. In particular, the structured knowledge bases like DBpedia [13] and Yago [136] that are extracted from Wikipedia, provide the opportunity to identify different categories of the entities. DBpedia contains the semantic types of entity appear on Wikipedia. Each semantic type belongs to a class in DBpedia ontology\(^1\) which is a hierarchy of different semantic types. Therefore, DBpedia provides a hierarchy of types that describes the generic and

\(^1\)http://dbpedia.org/ontology/
specific semantic types of Wikipedia entities. For instance, the most specific semantic type of “Jackie Chan” defined by DBpedia is an actor, which refers that “Jackie Chan” is an artist, person and agent in addition to an actor. With these semantic types, we can investigate the distribution of different types of entities in Wikipedia. Wikipedia consists of 4.2M entities. Table 2.1 shows that out of these 4.2M entities, there are 1,445,000 people, 735,000 places, and 241,000 organizations. It means 55% of Wikipedia entities are the standard types of named entities, however, it contains more than 2M other entities that are beyond the standard types such as 251,000 species and 6,000 diseases. Consequently, this thesis focuses on Wikipedia entities and considers every Wikipedia entry as an entity.

<table>
<thead>
<tr>
<th>DBpedia Class</th>
<th>Total Instances</th>
</tr>
</thead>
<tbody>
<tr>
<td>Person</td>
<td>763,643</td>
</tr>
<tr>
<td>Athlete</td>
<td>185,126</td>
</tr>
<tr>
<td>Artist</td>
<td>61,073</td>
</tr>
<tr>
<td>Politician</td>
<td>23,096</td>
</tr>
<tr>
<td>Place</td>
<td>572,728</td>
</tr>
<tr>
<td>Populated Place</td>
<td>387,166</td>
</tr>
<tr>
<td>Building</td>
<td>60,514</td>
</tr>
<tr>
<td>River</td>
<td>24,267</td>
</tr>
<tr>
<td>Organisation</td>
<td>192,832</td>
</tr>
<tr>
<td>Company</td>
<td>44,516</td>
</tr>
<tr>
<td>Educational Institute</td>
<td>42,270</td>
</tr>
<tr>
<td>Band</td>
<td>27,061</td>
</tr>
<tr>
<td>Work</td>
<td>333,269</td>
</tr>
<tr>
<td>Musical Work</td>
<td>159,070</td>
</tr>
<tr>
<td>Film</td>
<td>71,715</td>
</tr>
<tr>
<td>Software</td>
<td>27,947</td>
</tr>
</tbody>
</table>

Table 2.1 Number of instances per classes in DBpedia

2.2 Semantic similarity vs relatedness

Measuring the semantic distance between two natural language expressions is one of the primary interests of computational linguistic and cognitive science
In linguistics, semantics are defined by “how the components of language i.e. words and phrases map to the meaning in communicator’s mind”. However, cognitive science describes the semantics with the people’s conceptual space that is built by them with their experiences in the real world. Both of these theoretical backgrounds lead the discussion that semantic meaning of an expression can be interpreted by mapping them in a common conceptual space. The fundamental research questions of this study are “if two natural language expressions have similar meaning” and “what are the relations between these natural language expressions”. Further, the natural language processing (NLP) research derives the methods to solve these questions. Therefore, it addresses “how to measure if two natural language expressions are similar or dissimilar” and “what are the relation between them”.

The notion of similarity do not cover all the relations between two words. Similarity is usually reflected by the degree of “substitutability”. For instance, in a sentence “I want a cup of coffee”, the word “cup” can be substituted with

---

2.2 Semantic similarity vs relatedness

Fig. 2.2 Similarity and Relatedness

another similar word “mug” without changing the semantic meaning of this sentence. However, a broader range of relations can be covered by defining relatedness between two words. Two words can be considered related if they have any relation between them like cup is related with tea, coffee, plate and many more. Relatedness considers the related functions of two things rather than the same or similar functions. Therefore, relatedness can be considered as a superset of similarity. We can conclude that all the similar things are related but it is not necessary that all the related things are always similar.

In lexical semantics, similarity mainly reflects the taxonomic relations while relatedness covers all the lexical relations. For instance, figure 2.1 shows a snapshot of WordNet 3.1 for the word “car” with its neighboring terms. Car has synonyms, automobile and auto, that are replaceable in a context without changing the semantic sense. Moreover, the hypernym “motor vehicle” can be used to substitute the word “car”. However, “car window” and “car seat” have different semantic sense than car. They are meronyms of “car” representing the “part-of” relation. Therefore, these terms can be considered related to “car”
but not similar. The lexical resources like WordNet allows us to derive a notion about synonymy, similarity and relatedness. It may conclude that synonymy defines the replaceable relation and the lexical entries in taxonomic relations can be considered similar to each other depending upon their strength of similarity. The lexical entries connected with other relations come under the notion of relatedness. Therefore, all the synonyms can be considered similar and all the similar terms can be considered related to each other. Figure 2.2 shows that related words are less substitutable compare to synonyms. Further, we define below different notions of similarity and relatedness for text and entity.

- **Text Similarity** measures takes two natural language texts as input and provide a score corresponding to their degree of substitutability. For example, most of the WordNet-based measures give high scores to synonyms compare to meronyms. We describe WordNet-based similarity measure in detail in section 3.1.1.

- **Text Relatedness** measures takes two natural language texts as input and compute the degree of associativity between them. These measures consider the broad range of relations to capture the associativity between given texts.

- **Entity Similarity** measures provide the scores similar to text similarity measures. However, it takes two entities as input rather than natural language texts. Therefore, these measures can take advantage of additional information provided through disambiguated surface forms with their corresponding definitions in a knowledge base.

- **Entity Relatedness** measures takes two entities as input and compute the degree of connectivity between them.

Since, relatedness addresses the broad range of relations and also captures the notion of similarity, this thesis focuses on developing a relatedness measure for entities which considers different characteristics of entities other than their surface forms provided as input.
2.3 Distributional Semantics

The distributional hypothesis states that the words co-occur together more often, tend to have similar semantic meaning. The statement is derived from the linguistic theory [48] “a word is characterized by the company it keeps” popularized by Firth. Due to the simplicity of distributional hypothesis, it has received huge attention and has shown a remarkable impact in Computational Linguistics. Simply analyzing the language usage of a word allows us to obtain its semantic representation. For instance, the locally popular or globally unknown words like “bardiwac” may not be trivial to understand for everyone. However, it is easy to guess the meaning with its usage in different context. Figure 2.3 shows different sentences that help a reader in understanding the meaning of word “bardiwac”\(^3\).

The importance of using distributional representation of words has been shown in many applications like document retrieval [41, 137], sentiment analysis [89], and text categorization [51]. Most of these applications utilize the distributional representations to obtain the semantic relatedness scores between natural language texts i.e. words, phrases and sentences. The Distributional Semantic Models (DSM) generate the vectors to represent the semantics of a text. Therefore, it can be used to calculate the relatedness score between two corresponding vectors by quantifying the distance between them. Although, the distributional representation originated from linguistic theory, it also has the foundational background in cognitive science. The similarity based generalization [149] in cognitive science states that children learn the meaning of new words by their usage and distribution with similar words. Therefore, we can assume that humans also map words on other words or concepts that they are familiar with. For instance, the word orange can be mapped on color and fruit, as it usually refers to a fruit or a color.

Figure 2.4 shows that the words orange, peach and avocado can be interpreted with three concepts “color”, “fruit” and “tree”. It illustrates that orange usually refers to a fruit or the color but not to a tree. However, peach refers more to a fruit than color and tree, and avocado mainly refers to a fruit besides it be-

\(^3\)Figure is taken from Sketch engine
Background

Fig. 2.3 The usage of word “bardiwac” in different sentences

Fig. 2.4 Vector representation of different words in three dimensional space
2.3 Distributional Semantics

**<Doc title="Color">** Color is the visual perceptual property corresponding in humans to the categories like red, blue, **orange**, yellow, **peach** and many more. **Orange** and **peach** are the popular colors. **Orange** is the combination of red and yellow.

**<Doc title="Fruit">** Fruit is the sweet and fleshy product of a tree or other plant that can be eaten as food like **orange**, banana, **peach**, **avocado** and apple. **Orange** is a citrus fruit and **peach** is a prunus fruit. China is the world’s largest producer of **peaches** while **oranges** are produced world wide.

**<Doc title="Tree">** Tree generally has a wooden structure. There are several type of trees like Pine, Oak, **Avocado** and **Peach**. The **avocado** is a tree native to Mexico and Central America.

Fig. 2.5 Word distribution in different documents

![Word distribution in different documents](image)

Fig. 2.6 Co-occurrence based vector representation of different words in three dimensional space

![Co-occurrence based vector representation](image)
Table 2.2 Co-occurrence matrix

<table>
<thead>
<tr>
<th></th>
<th>fruit</th>
<th>color</th>
<th>tree</th>
</tr>
</thead>
<tbody>
<tr>
<td>orange</td>
<td>3</td>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>peach</td>
<td>3</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>avocado</td>
<td>1</td>
<td>0</td>
<td>2</td>
</tr>
</tbody>
</table>

longs to a particular shade\(^4\) of green color and a tree. With this representation of different things, it is trivial to identify that orange is more related to peach than avocado. Although, humans learn this representation by their experience with real world, it may be obtained from the word usage in the language. For instance, figure 2.5 shows three documents about fruit, color and tree. A term-document matrix can be obtained from the word usage in these documents. Table 2.2 shows the matrix which is based on the occurrences of words in the given documents. This matrix allows us to represent orange, peach and avocado on three dimensional space of fruit, color and tree. Figure 2.6 illustrates that we can obtain similar representation by word usage in the language that we deduce in the human brain (Figure 2.4). However, the concept space built by using given three documents fails in interpreting avocado as a color. DSM models reply on co-occurrence information to calculate relatedness between two words. Long history of co-occurrence model in computational linguistics. GVSM: extension of VSM to get co-occurrence from external corpus. Matrix factorization to smoothen the sparse representation. LSA (details in chap 3) issue in latent topics, so ESA came with high quality documents for explicit topics. Further, now NN is adapted the co-occurrence in NN architecture, more towards by learning representation for word individually from the context in which the word is used.

\(^4\)http://en.wikipedia.org/wiki/Chartreuse_(color)#Avocado
Chapter 3

Related work

This chapter discusses different state of the art methods for entity relatedness and recommendation. Entity relatedness is a relatively new problem. Therefore, there is not much previous work that directly focuses on measuring the relatedness between two entities. However, most of the existing methods of computing similarity and relatedness scores between words can be adapted for measuring entity relatedness. Therefore, in this chapter, first we discuss the existing methods of calculating relatedness between words or natural language texts, followed by a discussion of state of the art methods for entity relatedness.

3.1 Text similarity and relatedness

In recent years, there have been a variety of efforts in improving similarity and relatedness measures, specifically in the context of measuring relatedness between text documents [18, 87, 94].

Most of these approaches address this problem from the viewpoint of word representation in a knowledge base or word distribution in documents based on statistical corpus-based analysis. Classical knowledge-based approaches make use of lexical resources such as WordNet [43] and Open Roget [70]. However, in recent years, there have been some text relatedness measures proposed which utilize encyclopedic knowledge like Wikipedia and DBpedia [13]. In order to use resources like DBpedia, these measures require disambiguated entities instead of
just the surface forms. Thus, we discuss these methods in the entity relatedness section (3.2).

Although most of the WordNet-based measures mainly focus on computing semantic similarity scores on the basis of taxonomic (is-a) relations of two concepts, Pirro and Euzenat [116] make use of other WordNet relations to compute relatedness scores between words.

3.1.1 WordNet-based similarity measures

WordNet-based methods calculate similarity on the basis of a given semantic (taxonomic or ontological) structure. WordNet is a large lexical database, where concepts are connected to each other through a set of relations. It is intended to take psycho linguistic findings into account in its design [43]. All concepts follow the taxonomic structure that employs the IS-A (inheritance) relation. Every concept in WordNet is defined by its synsets and gloss. Synset is the set of synonyms for a word, i.e. words with a similar meaning.

Many different methods are proposed for calculating similarity scores, using the hierarchical structure and different glosses associated with each WordNet synset. They can be divided into three types of similarity measures as follows:

- Path based similarity
  A simple way to define similarity is the path length between concepts. The minimum number of edges to connect two different concepts can be used to reflect the degree of their similarity. This similarity is proposed by Rada et al. [121], where they only use the hypernym relation to count the edges. The similarity score between two words are computed using equation 3.1, where the shortest path is the minimum number of edges connected through any synset in WordNet.

$$sim_{Path}(c_1, c_2) = \frac{1}{\text{path-length}(c_1, c_2)}$$ (3.1)
This approach relies on the assumption that edges represent uniform distances, which is not always the case. For instance, concepts like “animal” and “plant” have a common hypernym “organism” but may not be considered very similar. To overcome this problem, Wu and Palmer [145] proposed a similarity measure, which uses the number of edges and depth of Least Common Subsumer (LCS)\(^1\) of corresponding concepts being compared and defined the similarity by their ratio. Thus, this approach distinguishes generic and specific concepts by using the depth of the hierarchy. The similarity score is calculated by equation 3.2

\[
sim_{W&P}(c_1, c_2) = \frac{2 \times \text{depth}(\text{LCS})}{\text{depth}(c_1) + \text{depth}(c_2)}
\]  
(3.2)

- Information content based similarity

These methods use corpus evidence to remedy the problem of non-uniform distance by using the notion of “information content”. Information content (IC) is measured by using the negative log likelihood of a concept considering that the more probable a concept is of appearing in a corpus, the less information it conveys. The IC of a concept is defined by its specificity. This idea is introduced by Resnik [125], where IC is calculated by counting the frequency of a given concept in a reference corpus. This measure defines the similarity of two concepts in proportion of the information they share, thus reflecting the commonality of two concepts. This similarity is calculated as the information content of the LCS of two concepts in a reference corpus (equation 3.3).

\[
sim_{Res}(c_1, c_2) = IC(\text{LCS}(c_1, c_2))
\]  
(3.3)

However, only considering the commonality does not differentiate between the similarity computations of two specific concepts and two abstract concepts, if they all share the same LCS. To overcome this issue, Jiang and Corath [76] define similarity as inverse of the difference between the sum of individual concept ICs and the IC of LCS of both concepts (equation 3.4).

\[1\text{LCS is the closest common hypernym of both the concepts}\]
\[
sim_{J\&C}(c_1, c_2) = \frac{1}{IC(c_1) + IC(c_2) - 2 \times IC(LCS(c_1, c_2))}
\]  
(3.4)

Lin [90] also makes use of the IC of the given concepts by using equation 3.5, where IC is computed from an additional reference corpus.

\[
\sim_{Lin}(c_1, c_2) = \frac{2 \times IC(LCS(c_1, c_2))}{IC(c_1) + IC(c_2)}
\]  
(3.5)

The main drawback of these measures is the requirement of an additional large corpus to obtain the IC. To overcome this issue, Pirro [114] calculates similarity by taking the intrinsic information content (iIC) into account, where iIC of a given concept is calculated by counting its subconcepts (equation 3.6).

\[
\sim_{Pirro}(c_1, c_2) = \frac{iIC(LCS(c_1, c_2))}{iIC(c_1) + iIC(c_2) - iIC(LCS(c_1, c_2))}
\]  
(3.6)

- Gloss based similarity

This measure uses the gloss associated with the concepts of WordNet. Lesk [15] calculates similarity by taking the overlap between glosses of concepts being compared. As every word can have multiple senses in WordNet, gloss based measures also use corresponding part of speech (POS) information to identify the most appropriate synset to obtain the gloss.

### 3.1.2 WordNet based relatedness measure

As we explain above, WordNet is a large lexical database, where all concepts are connected through a set of relations. Wordnet defines additional relations beyond taxonomic relations, which is also useful to assess in what extent two concepts are alike. Pirro and Euzenat [116] proposed extending intrinsic information content (eIC) by investigating all kinds of taxonomic and non-taxonomic relationships between concepts. By taking other relations into account, it can also provide the relatedness of two concepts. For instance, car and scooter are related as they both serve to transport people or objects. They share all features,
which are applicable for the concept with a type “vehicle”. Beyond the taxonomic relation, we can find different kinds of relations such as meronym and holonym, e.g. axle, brake, wheel, and engine are the meronym of both car and scooter.

3.1.3 Corpus-based measures

Corpus-based measures rely on co-occurrence information, which assumes that related words will appear together in a text. Semantic relatedness of two given text fragments (word, phrase, sentence etc.) can be obtained by calculating the similarity between their high dimensional vectors in a distributed semantic space. Distributional representation is based on the underlying idea proposed by Firth [47] that “the semantic meaning of a word can (at least to a certain extent) be inferred from its usage in context”, i.e. its distribution in text. This semantic representation is built through a statistical analysis over the large contextual information in which a word occurs. Distributional Semantic Models (DSM) compute the relatedness scores by using distributional representation. DSMs are based on the distributional hypothesis introduced by Harris [59], i.e. words that occur in the same contexts tend to have similar meaning.

3.1.3.1 Latent Semantic Analysis

The most common DSM model is Latent Semantic Analysis (LSA) ([85], [49]). LSA constructs a semantic space of a large text corpus by first casting it onto a rectangular matrix of words by documents, where each cell contains the association strength of a word in a given document and each row represents a unique word. This matrix is decomposed by using a well known algebraic matrix factorization method called Singular Value Decomposition (SVD) in which the k largest singular values are retained and the remainder are set to 0. LSA relies on word distribution represented by a word-document matrix and calculates similarity between words and documents by taking the cosine of the two corresponding vectors in this k-dimensional space.

Let \( M = T_i * D_j \) represent the rectangular word-document matrix, where \( T_i \) and \( D_j \) refer to \( i^{th} \) word and \( j^{th} \) document respectively. This matrix \( M \) can be
decomposed into a product of three matrices $U \Sigma$ and $V$, where $U$ and $V$ are orthogonal matrices and $\Sigma$ represents the singular value matrix. This matrix $\Sigma$ turns the high dimensions space into the $k$ dimensional space by retaining the $k$ largest singular values. This can be represented by the following equation:

$$M = U \Sigma V^T$$ (3.7)

Here, $U$ and $V$ contain the Eigen vectors of $MM^T$ and $M^TM$ respectively. Therefore the relatedness between two terms can be represented as

$$rel(T_i, T_r) = U_i \Sigma_k \ast U_r \Sigma_k$$ (3.8)

Similarly, the relatedness between two documents as

$$rel(D_i, D_r) = \Sigma_k V_i \ast \Sigma_k V_r$$ (3.9)

3.1.3.2 Explicit Semantic Analysis

Gabrilovich and Markovitch [52] introduced Explicit Semantic Analysis (ESA) which attempts to represent the semantics of a given word in a high dimensional distributional semantic space similar to LSA. LSA performs dimensionality reduction to obtain the latent concepts. On the contrary, ESA directly uses supervised topics such as Wikipedia concepts that are built manually, and considers that every concept represents a unique topic. ESA creates a high dimensional vector to represent the semantics of a word, where every dimension reflects a unique Wikipedia concept/article. This high dimensional vector is created by taking the TF-IDF weight of a given word in the corresponding Wikipedia article. The semantic relatedness between two words is expressed by a cosine score between the corresponding vectors. ESA represents composite semantics by creating a high dimensional vector of a document, which is the vector addition of the vectors of each word appearing in the given document.

Figure 3.1 illustrates the process of building an ESA vector and calculating the relatedness scores. ESA requires to preprocess the data to build the inverted
3.1 Text similarity and relatedness

Fig. 3.1 Explicit Semantic Analysis

Index of every word appearing in a corpus. For instance, the figure shows that it creates an inverted index over Wikipedia articles. Every entry in the index represents a word and its DSM vector, where $W_{ij}$ is the tf-idf weight of word $i$ with Wikipedia article content that has URI $j$. The length of each vector is $N$, as there are $N$ articles in Wikipedia. Thus, ESA generates a very big and sparse vector, in comparison to LSA. With the built inverted index, ESA retrieves the vectors for all the words and by adding them the semantic interpreter generates the vector for the given text documents. Finally, it computes a cosine score between the obtained vectors. There are several other corpus based measures, which use the probability distribution of the term over a large corpus. Thomas Hoffman proposed Probabilistic Latent Semantic Analysis (PLSA) [64], which extends the classical concept of LSA with a strong statistical foundation by calculating the probability distribution. The accept model is used to maximize the distribution. PLSA does not include the prior distributions of the topics. By using Dirichlet prior distribution, Latent Dirichlet Allocation [23] showed significant improvement over PLSA. On the same basis, there are more machine leaning based models [100, 109, 123] that perform topic modeling and represent the semantics of
a word by using dense vectors over these hidden topics.

### 3.1.3.3 Word embeddings

Learning semantic representations of words using neural network architecture has recently received very high popularity due to its ability to learn high quality semantics in a dense low dimensional space. The most popular and easy to use method is called Word2Vec [100] that learns the representations by using language model over a very large corpora. Mikolov et al. [100] proposed the Skip-gram model to learn the representations, which is trained to an objective of predicting the nearby words. Let $w_i$ is a given word at $i^{th}$ position, the Skip-gram model would predict the adjacent words $w_{i-2}$, $w_{i-1}$, $w_{i+1}$ and $w_{i+2}$. Although, Word2Vec achieved high accuracy in different NLP tasks, it relies only on the local context around the words. Therefore, the learned representations are very sophisticated to the local context and do not take benefit from other words which are a bit far in a bigger context window. Pennington et al. [109] proposed GloVe that considers a global context similar to other matrix factorization based methods. However, GloVe also added a context representation with the word vector by giving a higher preference to the local context. Levy et al. [88] showed that GloVe can be seen as a matrix factorization method similar to LSA. However, considering extra context representations in GloVe makes it to perform better than LSA and Word2Vec in word similarity task.

Since, word embeddings are learned by giving preferences to local context, they tend to capture the words with high substitutability. Therefore, these methods improve the accuracy in finding similar words but they may fail to capture the related words, which do not appear in the same or similar context and generally appear far from each other in a bigger context window. In chapter 6, we analyze the performance of Word2Vec and GloVe in calculating word similarity and relatedness.
3.2 Entity relatedness

As mentioned above, entity relatedness is a relatively new problem but most of the existing text relatedness measures can be adapted to calculate the relatedness scores between entities. Recently, some approaches have been presented that mainly focus on calculating relatedness between entities, where two URIs or referenced ids in a knowledge base are given instead of their surface forms\(^2\), to compute the degree of their relatedness. Strube and Ponzetto [135] introduce a model that makes use of the hyperlink graph in Wikipedia to calculate the relatedness scores between two Wikipedia articles. In order to compute the relatedness scores between two words, it first requires to link the given words to their corresponding Wikipedia articles. This approach was presented to calculate word relatedness instead of entity relatedness, however, due to its wide coverage of real world named entities, Wikipedia often adopted for computing the relatedness scores between entities. Moreover, many existing word relatedness measures rely on lexical resources like WordNet which are limited to perform only for standard English dictionary words. Therefore, following previous works we focus on calculating entity relatedness.

3.2.1 WikiRelate

WikiRelate is the first model that makes use of referenced URI of a word in knowledge bases [135]. It shows the effectiveness of using disambiguated surface forms. For instance, comparing the surface forms “apple” and “orange” will be more accurate if their senses are given explicitly. Thus, Strube and Ponzetto [135] use Wikipedia links corresponding to given words, and adapt the previously proposed methods of calculating semantic relatedness by using the WordNet hierarchy. This model adapts two existing WordNet-based measures, i.e. the modified path-based measure proposed by Leacock and Chodorow [86], and Resnik’s measure [124] that uses information content. In order to adapt these measures, it formulates the WordNet-based equations for Wikipedia hyperlink graph. Therefore, WikiRelate computes the relatedness between two Wikipedia

\(^2\)label in natural language text
Related work

Links by using equation 3.10, where length(c1, c2) is the number of edges in the shortest path between c1 and c2, and D is the maximum depth of the Wikipedia category structure.

\[
sim_{L\&C}(c_1, c_2) = -\log \frac{\text{length}(c_1, c_2)}{2D}
\]  

(3.10)

To adapt Resnik’s measure, it uses the intrinsic information content instead of information content in a referenced corpus. It uses the Wikipedia categories to obtain the relatedness scores by using equation 3.11, where hyponyn(MSC(c1, c2)) represents the number of hyponyms of MSC (Most Specific Category) of given concepts in the Wikipedia category graph.

\[
sim_{Res-Wiki}(c_1, c_2) = 1 - \frac{\log(\text{hyponyn}(\text{MSC}(c_1, c_2)))}{\log(C)}
\]  

(3.11)

3.2.2 Wikipedia Link-based Measure (WLM)

Witten and Milne [144] introduce a low-cost semantic relatedness measure that uses Wikipedia links. The measure is low-cost as it requires only the incoming links in the corresponding Wikipedia article instead of full text content. Moreover, WLM takes advantage of more closely tied concepts that are manually tagged to define the semantics of the concept. Incoming links carry important information as links are referenced to explain the concepts in Wikipedia. For instance, “Apple” is referenced to explain concepts such as “iPhone”, “Mac” and “iPad”, and the semantic of “Apple” can be defined with such concepts. Thus, WLM relies on the hypothesis that the semantic relatedness of two concepts can be defined by the number of incoming links they share. WLM adapts the Google distance measure proposed by Cilibrasi and Vitanyi [28] to calculate the relatedness scores. The name stems from the use of Google search results, however, it can be adapted to any measure that uses a list of features to define the semantics of given concepts. For instance, WLM uses a list of incoming links of given concepts instead of a list of search results. Formally it can be defined by equation
3.2 Entity relatedness

\[ sim_{WLM}(c_1, c_2) = \frac{\log(max(|I_{c_1}|, |I_{c_2}|)) - \log(|I_{c_1} \cap I_{c_2}|)}{\log(|W|) - \log(min(|I_{c_1}|, |I_{c_2}|))} \]  \hspace{1cm} (3.12)

where \( I_{c_1} \) and \( I_{c_2} \) are the sets of all Wikipedia concepts that link to given article \( c_1 \) and \( c_2 \) respectively, and \( W \) is the total number of Wikipedia concepts.

### 3.2.3 Key-phrase overlap for entity relatedness (KORE)

Hoffart et al. [62] calculate entity relatedness scores by taking the overlap between sets of weighted key-phrases. The key-phrases are extracted from Wikipedia articles corresponding to the entities of interest. In order to obtain key-phrases, this approach gathers all the anchors that are linked to the entity, category labels, and titles of citations, in an article. Moreover, this approach can be applied to non-Wikipedia entities, as every noun phrase can be considered a key-phrase. Considering all noun phrases may introduce noise which can be kept low by using a weighing scheme. KORE computes the importance of a key-phrase by using Inverse Document Frequency (IDF) and Mutual Information (MI). IDF captures the overall importance of an entity in a corpus and MI represents the associativity of a key-phrase with a given entity. The IDF of an entity \( e \) is calculated by using equation 3.13, where \( df(e) \) is document frequency of entity \( e \).

\[ idf(e) = \log_2 \frac{N}{df(e)} \]  \hspace{1cm} (3.13)

In order to compute MI scores, it calculates the ratio of events of an entity key-phrase that co-occur relative to appearing individually. The MI score \( \mu \) between entity \( e \) and key-phrase \( p \) is calculated by equation 3.14, where \( E(e) \) and \( E(p) \) represent the marginal entropy of entity \( e \) and key-phrase \( p \), and \( E(e, p) \) is the joint entropy.

\[ \mu(e, p) = 2 \times \frac{E(e) + E(p) - E(e, p)}{E(e) + E(p)} \]  \hspace{1cm} (3.14)

Since key-phrases can be partially overlapping, KORE calculates a Phrase Overlap (PO) score that uses MI scores of each word that occurs in a key-phrase. For instance, the key-phrases “unsupervised machine learning” and “unsupervised-
vised learning” are not exactly overlapping even though they have very similar meanings. Therefore, KORE considers partial overlap between extracted key-phrases to measure entity relatedness. Let entities $e$ and $f$ have key-phrase sets $P_e = \{p_1, p_2, \ldots\}$ and $P_f = \{q_1, q_2, \ldots\}$ respectively, and a key-phrase contains word sequence $\{w_1, w_2, \ldots\}$. The PO score between key-phrases $p$ and $q$ is calculated by equation 3.15, where $\gamma_e(w)$ is the MI score of a word $w$ with entity $e$.

$$PO(p, q) = \frac{\sum_{w \in p \cap q} \min\{\gamma_e(w), \gamma_f(w)\}}{\sum_{w \in p \cup q} \max\{\gamma_e(w), \gamma_f(w)\}}$$  (3.15)

Finally, to compute the relatedness scores between entities, KORE makes use of both scores explained above, i.e. MI weights of key-phrase $p$ and the scores obtained from partially overlapping key-phrases. Equation 3.16 is used to calculate the relatedness scores, where $\varphi_e(p)$ represents the MI score of key-phrase $p$ with entity $e$ and $PO(p, q)$ is the partial overlapped score between key-phrases $p$ and $q$ corresponding to entity $e$ and $f$ respectively. Note that KORE considers the partial overlap between all the extracted key-phrases.

$$KORE(e, f) = \frac{\sum_{p \in P_e, q \in P_f} PO(p, q)^2 \times \min\{\varphi_e(p), \varphi_e(q)\}}{\sum_{p \in P_e} \varphi_e(p) + \sum_{q \in P_f} \varphi_f(q)}$$  (3.16)

### 3.2.4 Exclusivity in knowledge graph

Recently, Hulpuş et al. [67] presented an approach that makes use of the DBpedia graph instead of the Wikipedia hyperlink graph. In particular, it adapts Katz centrality measure [81] that is commonly used in social network graphs. Katz’ score can be considered as a generalization of the standard path-based relatedness measure presented by Rada et al. [121]. Equation 3.17 is used to calculate the relatedness scores between entity $e$ and $f$ by using the Katz measure, where $\text{length}(p)$ is the number of edges in a path from entity $e$ to $f$. This score is accumulated over top-$k$ shortest paths from $e$ to $f$. Katz’ parameter $\alpha$ represents the constant probability that penalizes the longer paths. Hulpuş et al. [67] experimented with different values of $\alpha$.

$$rel_{Katz}^{(k)}(e, f) = \frac{\sum_{p \in SP_{ef}^{(k)}} \alpha^{\text{length}(p)}}{k}$$  (3.17)
Knowledge bases like DBpedia and Freebase consist of many different types of relations and all the directly connected relationships can not be considered equally important. For instance, a city connected to a person as his/her birth place can be considered more related to the person than a country connected to a person as his/her birth place. Thus, this approach considers exclusivity of every relationship as it occurs in the path between entities of interest. The exclusivity of an edge from a node $x$ to node $y$ is defined by equation 3.18, where $Count(x \rightarrow R \ast)$ is the number of nodes that are connected from node $x$ with a relationship $R$, and $Count(\ast \rightarrow y)$ is the number of nodes that are connected to node $y$ with a relationship $R$.

$$exclusivity(x \rightarrow R \rightarrow y) = \frac{1}{Count(x \rightarrow R \ast) + Count(\ast \rightarrow y) - 1}$$ (3.18)

Finally it uses equation 3.19 to compute the relatedness scores, where $P_i$ is the $i^{th}$ shortest path from entity $e$ to $f$.

$$rel_{excl}^{(k)}(e, f) = \sum_{P_i \in P_e^{(k)}} \alpha^{length(P_i)} \times \frac{1}{exclusivity(P_i)}$$ (3.19)

### 3.2.5 Entity embeddings

In recent years, there have been a lot of efforts in building embeddings for entities. However, most of them focus on knowledge base completion. Therefore, the existing neural network architectures are built to predict new links for a given entity and their relations, and to classify the credibility of newly extracted triplets. Many models have been proposed for knowledge base representation learning [25, 104, 142]. Nickel et al. [104] proposed the collective learning for relational representations using tensor factorization. Bordes et al. [25] made use of text and structured knowledge base to perform relation extraction by learning joint representations of entities and relations. Yang et al. [148] learned the entity and relation embeddings without explicit logical constraints and mined logical
rules directly from a knowledge base. Most of the knowledge bases are very sparse and do not contain all the entities that are presented in their referenced text. To deal with the sparsity of knowledge bases, Wang et al. [142] proposed a model that finds missing entities in text during representation learning. Since, these models learn the entity representations with an objective function to identify appropriate entity for a given entity and relationship pair, it is difficult to use them directly for comparing two entities without providing their relationships. However, recently, Blanco et al. [21] learned the entity representations using a large entity annotated corpus collected from Wikipedia, news and weblogs, to perform entity disambiguation in search queries. Further, Iacobacci et al. [68] proposed SensEmbed that learns the entity and word embeddings on an annotated Wikipedia with entities and word senses. In order to obtain an annotated Wikipedia, SensEmbed uses BabelNet [102] to annotate all the entities and words appearing in all the Wikipedia articles, with their appropriate senses. SensEmbed achieves high accuracy for word similarity and word sense disambiguation [69]. Therefore, in order to compare our entity relatedness measure with entity embeddings based model, we build the entity representations similar to SensEmbed model.

3.3 Analysis over different dimensions of relatedness measures

Although the above described entity relatedness measures make use of Wikipedia to obtain background knowledge about entities, they particularly rely on the hyperlink structure of Wikipedia. Besides the hyperlink structure, Wikipedia also contains other background knowledge that is not utilized by these measures. For instance, it provides the Wikipedia category structure, importance of hyperlinks for an article, co-occurring hyperlinks, and popularity of entities. Hulpuş et al. [67] make use of the Wikipedia category structure along with the DBpedia knowledge base. However, their method performs comparable to the previously presented measures. Moreover, the WikiRelate, WLM and exclusivity measures are limited to perform only for the entities which appear on Wikipedia. KORE
Table 3.1 Characteristics of relatedness measures

<table>
<thead>
<tr>
<th>Method</th>
<th>DSM</th>
<th>Distributional representation type</th>
<th>Graph Specific</th>
<th>Wikipedia Specific</th>
<th>Pre-training</th>
</tr>
</thead>
<tbody>
<tr>
<td>WordNet-based</td>
<td>No</td>
<td>-</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>LSA</td>
<td>Yes</td>
<td>Dense</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>ESA</td>
<td>Yes</td>
<td>Sparse</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Word embeddings</td>
<td>Yes</td>
<td>Dense</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>WikiRelate</td>
<td>No</td>
<td>-</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>WLM</td>
<td>No</td>
<td>-</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>KORE</td>
<td>No</td>
<td>-</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Exclusivity</td>
<td>No</td>
<td>-</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Entity embeddings</td>
<td>Yes</td>
<td>Dense</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>DiSER</td>
<td>Yes</td>
<td>Sparse</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Context-DiSER</td>
<td>Yes</td>
<td>Sparse</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
</tbody>
</table>

[62] overcomes this issue by using the key-phrase sets extracted from the contexts of the given entities. The extracted key-phrases can be considered as outgoing links in Wikipedia articles, which provide the background knowledge available in the article content corresponding to a given entity. However, it does not use the background knowledge available across all other articles, i.e., incoming links to the article. For instance, the Wikipedia article of the entity “Apple” provides important entities to describe it but there are many other entities which are described by referencing “Apple”. Although WLM uses the incoming links, it does not utilize the importance of hyperlinks for an entity. Thus, we focus on developing an entity relatedness measure utilizing the incoming links with their importance weights. In particular, we adapt distributional semantic model to calculate relatedness between entities.

### 3.4 Entity recommendation

The task of entity recommendation can be defined by finding a ranked list of related entities for a given entity. Although existing classical recommendation systems [126] also deal with entities such as movies, books and TV shows, they mainly focus on personalized recommendations by using user preferences. These systems define user preference by their liked and disliked items, and recommend
the list of items that are similar to their liked ones. Therefore, these systems rely on existing user-item preferences and generate the recommendations by using a standard approach called collaborative-filtering [130]. Moreover, the goal of such recommendation systems is to increase online shopping on e-commerce websites like Amazon.com\(^3\) and Netflix\(^4\). Since collaborative-filtering requires a large data sets of user-item preferences, it is limited to perform recommendations for a new user or a new item. This is a common issue with collaborative-filtering which is called the cold-start problem [131]. Therefore, another line of research in recommendation systems deals with content-based recommendation that generates the recommendations by using the similarity between implicit features of the items. It calculates the similarity of items with the items liked by a user. Thus, it can even deal with a small number of preferences. For instance, content-based recommendation systems can generate recommendations by using a single item preferred by the user [34].

Entity recommendation focuses on generating recommendations where only one entity (item) is given at a time. Particularly, entity recommendation is used in Web search [19, 20, 151, 152] to increase user engagement and improve their experience with search engines. Blanco et al. [20] introduced an entity recommendation for Yahoo! search called “Spark” which makes use of more than 100 features extracted from different resources such as Yahoo! query logs, user search sessions, Flickr tags and tweets. However, data sources like query logs and search sessions are not publicly available. Therefore, we focus on developing an approach to perform recommendations by using publicly available data like Wikipedia. Our approach is inspired by the entity recommendation approach presented by Blanco et al. [20]. In chapter 5, we present a comprehensive study on different features used in “Spark”, and compare them with the features used in our approach. Since, major search engines can access the user search history, some recent work addresses personalized entity recommendation in Web search [19, 151]. For instance, Yu et al. [151] presented a personalized entity recommendation in Web search that utilizes the search click logs and entity pane logs.

\(^{3}\)www.amazon.com

\(^{4}\)www.netflix.com
for individual users. The approach is evaluated against the user clicks obtained from Bing search. Therefore, it is hard to evaluate these approaches as such gold standard datasets are not publicly available. In this thesis, we focus on generating the same recommendations for a given entity to different users and do not deal with personalization.
Chapter 4

Computing entity relatedness

Wikipedia provides background knowledge about millions of real world entities. This knowledge can be used in measuring the relatedness between two entities. Although Wikipedia contains different forms of knowledge including textual descriptions, interlinked articles and extracted structured knowledge like DBpedia, the main challenge is to utilize the appropriate knowledge to build an entity relatedness model. Several efforts [52, 60, 66, 115] have been made in exploiting Wikipedia knowledge to measure the semantic relatedness between words or texts. However, these methods require an adaptation of appropriate structured knowledge to build a model for computing entity relatedness scores. Most of the methods [52, 60] that measure the semantic relatedness between texts, make use of textual content by building distributional semantic vectors of the texts. There are some works on graph based methods also using the DBpedia graph [66, 115]. However, Distributional Semantic Models (DSM) outperform the graph based methods in text relatedness [115]. We focus on DSM using the text content and links provided through Wikipedia anchors. Moreover, we perform a comprehensive evaluation of using different types of information to measure entity relatedness.

As the distributional hypothesis [47] states that “the semantic meaning of a word can be derived from the company it keeps”, we represent the semantics of an entity through its distribution in a high dimensional concept space derived from Wikipedia. Although Wikipedia based DSMs [52, 60] have been proposed for
computing relatedness scores, they are limited to perform at surface levels of the entities. For instance, these methods may produce an ambiguous distributional vector for ambiguous surface forms like “apple” or “next”. If we retrieve the most associated Wikipedia concepts for the term “next”, we will get concepts like linked list, railway station, train schedule. However, if we retrieve the concepts which only contain an anchor “next” linked to the entity “NeXT” in Wikipedia, we will get concepts like Music Kit, NeXT, NeXT Computer. Therefore, we focus on distributional semantics using Wikipedia anchors in textual descriptions.

In this chapter, we explain our method of calculating semantic relatedness scores between two entities and evaluating them in different scenarios. In particular, we present Wikipedia-based Distributional Semantics for Entity Relatedness (DiSER), which represents the semantics of an entity by a high dimensional vector. DiSER builds this semantic vector over Wikipedia concepts by taking only anchors annotated by Wikipedia entities. Therefore, it eliminates an important limitation in the existing approaches [52, 85], which do not differentiate between the “apple” fruit and the “Apple” company. As DiSER is inspired by an existing model Explicit Semantic Analysis (ESA) [52], we perform an extensive comparison between DiSER and ESA for entity relatedness. Since DiSER can generate the vectors only for those entities which appear on Wikipedia, we also propose an alternative approach called Context-DiSER that builds a DiSER vector by taking additional resources such as music portal, personal websites, blogs or social network websites into account for retrieving the context. Context-DiSER eliminates the dependency on predefined direct interlinkages between the given entities, which is required for existing approaches [118, 144]. Hoffart et al. [62] utilize the context of entities to calculate semantic relatedness by taking the overlap between corresponding contexts, however, their method does not semantically interpret the context. Our goal is to develop an entity relatedness measure which overcomes these limitations and outperforms existing algorithms. We evaluate our approach in different tasks: ranking related entities, entity disambiguation, and top N books (entity) recommendation.
4.1 Entity relatedness using Wikipedia

4.1.1 Wikipedia-based Distributional Semantics for Entity Relatedness (DiSER)

We developed an approach called Distributional Semantics for Entity Relatedness (DiSER), which builds the semantic profile of an entity by using the high dimensional concept space derived from Wikipedia. DiSER generates a high dimensional vector by taking every Wikipedia concept as dimension, and the associativity weight of an entity with the concept as the magnitude of the corresponding dimension. To measure the semantic relatedness between two entities, we simply calculate the cosine [154] score between their corresponding DiSER vectors. DiSER considers only human annotated entities in Wikipedia, thus keeping all the canonical entities that appear with their anchors in Wikipedia articles. The tf-idf weight of an entity with every Wikipedia article is calculated and used to build the corresponding semantic profile, which is represented by the Wikipedia concepts as vector dimensions and their tf-idf scores as the magnitude. Let \( A \) is an index collection of \( n \) Wikipedia articles, where each article \( a_i \) represents an article describing a single Wikipedia concept. DiSER represents entity \( e \) as a semantic vector \( v \) where each entry in \( v \) corresponds to the tf-idf weight of entity \( e \) with \( a_i \in A \).

\[
v^T = (\langle a_1, e \rangle, \langle a_2, e \rangle, \ldots, \langle a_n, e \rangle)
\] (4.1)

DiSER calculates the relatedness between entities \( e \) and \( f \) by taking cosine between their corresponding vectors \( v \) and \( u \).

\[
Rel_{DiSER}(e, f) = \frac{\langle v, u \rangle}{|v| \cdot |u|}
\] (4.2)

\[
Rel_{DiSER}(e, f) = \frac{1}{|v| \cdot |u|} \sum_{i=0}^{n} \langle a_i, e \rangle \cdot \langle a_i, f \rangle
\] (4.3)

\[
Rel_{DiSER}(e, f) = \frac{1}{|v| \cdot |u|} \sum_{j=0}^{m} \sum_{k=0}^{m} w(t_j, e) \cdot (t_k, f)
\] (4.4)
We pre-process all Wikipedia articles to keep only the manually tagged entities. Moreover, we also identify those instances of entities in the article which are not tagged by Wikipedia volunteers. Finally, we obtain an entity-tagged Wikipedia corpus containing only the bag of entities in every article. We explain the pre-processing in detail in section 4.1.3. DiSER is able to capture the semantic meaning of entities like “Apple” and “NeXT”. Therefore, it can improve over existing algorithms that build the distributional vector by taking only the surface forms into account. For instance, ESA generates the distributional vector of a term by calculating its tf-idf weight with all the Wikipedia articles. Table 4.1 shows the 10 most associated concepts of the entity “NeXT” obtained by ESA and DiSER. It illustrates that existing systems can not handle ambiguous terms as they generate the vector by considering the surface form of an entity. Manual analysis of the vectors generated by ESA and DiSER reveals that all the concepts retrieved by DiSER are relevant to the entity “NeXT”. However, ESA did not get any relevant concept as it is more biased towards the global meaning of the given term. Table 4.1 shows that ESA retrieved all concepts for the word “next” with the semantic meaning “succeeding item”. Therefore, it retrieved concepts which contain phrases like “next item in linked list” or “next train”.

Similar to WLM [144], DiSER vector can be seen as a vector of incoming links with their relevance weight with the given entity. For instance, to build the vector of an entity $e$, DiSER retrieves all the articles where entity $e$ appear as an outgoing link provided through an anchor, and calculates the tf-idf of $e$ with these articles. It means that every dimension of vector is represented by an incoming Wikipedia link to entity $e$, and the tf-idf weight gives the relevancy score of corresponding incoming link. In contrast, the best performing state of the art method KORE [62] considers only the outgoing links. KORE calculates the relatedness score by taking an overlap between important outgoing links. Therefore, we compare DiSER with KORE and WLM in section 4.2.
4.1 Entity relatedness using Wikipedia

<table>
<thead>
<tr>
<th>NeXT ESA vector</th>
<th>NeXT DiSER vector</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Doubly linked list</td>
<td>Music Kit</td>
</tr>
<tr>
<td>2. Gare de Rennes</td>
<td>NeXTSTEP</td>
</tr>
<tr>
<td>3. Brugge railway station</td>
<td>NeXT Laser Printer</td>
</tr>
<tr>
<td>4. Gare d’Avignon Centre</td>
<td>NeXT Computer</td>
</tr>
<tr>
<td>5. Gare de Toulon</td>
<td>Shelf (computing)</td>
</tr>
<tr>
<td>6. Szczecin Glowny railway station</td>
<td>RichPage</td>
</tr>
<tr>
<td>7. Gare de Clermont Ferrand</td>
<td>ISPW</td>
</tr>
<tr>
<td>8. Leipzig Central Station</td>
<td>Nancy R. Heinen</td>
</tr>
<tr>
<td>9. Brussels North railway station</td>
<td>Enterprise Objects Framework</td>
</tr>
<tr>
<td>10. Gare de Strasbourg</td>
<td>Lotus Improv</td>
</tr>
</tbody>
</table>

Table 4.1 10 most relevant Wikipedia concepts for entity “NeXT” by ESA and DiSER

4.1.2 Context-based DiSER

Since the model relies on the distributional hypothesis and makes use of co-occurring entities, it is limited to perform only for the entities which appear and co-occur in Wikipedia. We resolve this issue by generating the DiSER vector of entities appearing in the context of a given entity. For instance, Wikipedia does not have a page for “Matt Lasky” who is a Hollywood actor. Therefore, DiSER cannot generate the semantic profile for “Matt Lasky”. However, the IMDB page\(^1\) and his website\(^2\) define him with the entities appearing on Wikipedia such as “Hollywood”, “Actor”, “Pirates of the Caribbean” and “Princess of Mars”. These entities can be identified by using any existing entity linking tool such as Alchemy\(^3\) or Zemanta\(^4\). With these Wikipedia concepts, we can build the semantic profile of “Matt Lasky”. We described above the method to build a DiSER vector for one entity. Similarly, we can build it for more, by treating them as a bag of entities.

\(^1\)http://www.imdb.com/name/nm2359377/
\(^2\)http://mattlasky.nowcasting.com/
\(^3\)http://www.alchemyapi.com/
\(^4\)http://www.zemanta.com/blog/demo/
4.1.3 Implementation

We implemented our approach by using a snapshot of the English Wikipedia from Oct 1st, 2013. This snapshot consists of 13,872,614 articles, in which 5,659,383 are Wikipedia redirects. Wikipedia redirects are those articles, which do not contain any content, and just link to the article or the section of the article that defines a similar or related Wikipedia concept, for example, U.S.A. redirects to United States. Wikipedia also contains namespace pages, which are reserved by MediaWiki to describe Wikipedia projects like Wikipedia help and File pages. We filtered out all the namespace pages by using the articles title as they have specific namespace patterns. There were 3,571,206 namespace pages in this snapshot. We also removed all those articles which contain less than two anchors linked to entities; such articles would not provide any co-occurrence information for the entities. Finally, we obtained 4,102,442 articles after removing all the Wikipedia redirects, namespaces and short articles.

In order to use only manually annotated entities for generating the DiSER vectors, it is required to retain only those entities which are manually linked through Wikipedia anchors by volunteers. However, the volunteers may not create a link for every surface form appearing in the article content. For instance, “apple” occurs 213 times in the “Steve Jobs” Wikipedia page, but only 7 out of these 213 are linked to the “Apple Inc.” Wikipedia page. This term frequency of “apple” is calculated without considering the partial matches, for example, we do not count if “apple” appears as a substring of any annotated entities like “Apple Store” or “Apple Lisa”. Since we measure the associativity of an entity with the article by computing its tf-idf weight, this difference in term frequencies may have a major effect on the semantic profile interpreter. Therefore, to obtain the actual term frequency of every entity, we apply the “one sense per discourse” heuristic [54]. According to which, a term tends to have the same meaning in the same discourse. We annotated every additional unannotated occurrence of a term with the link appearing several times for the same term as an anchor in the discourse.
4.1 Entity relatedness using Wikipedia

Figure 4.1 illustrates the process of obtaining entity-tagged Wikipedia corpus. It shows an example of processing the Wikipedia article of “Steve Jobs”. First we identify all the anchors that link to other Wikipedia article, for instance, the anchors “Steve Jobs” and “Apple” are linked to Wikipedia articles of “Steve Jobs” and “Apple Inc.” respectively. However, we can see that other instance of “Apple” is not linked to “Apple Inc.”. Therefore, by following “one sense per document”, we also consider all instances of “Apple” as an anchor of “Apple Inc.”. We only consider the unannotated instances as anchors if they exactly match to an annotated anchor in the article. For instance, in the given article an anchor “Jobs” is linked to Wikipedia article of “Steve Jobs”, therefore, we considers all the instances of “Jobs” as an anchor of “Steve Jobs”.

Since DiSER builds the distributional vector for the given entity by calculating its tf-idf scores with every article, it may take a very long time to process 4.1 millions articles. Therefore, we built an inverted index of 4.1 million pre-processed arti-
icles by using Lucene\textsuperscript{5}. As this indexing is a one time process to build the DiSER vector, we can calculate entity relatedness for several thousand pairs within a second.

### 4.2 Evaluation in Ranking Related Entities

Ranking related entities is the task to rank the entity candidates retrieved for one entity. For instance, table 4.2 shows 20 entity candidates that can be considered related to “Apple Inc.”. These candidates are ranked by human annotators. A perfect ranking system would be able to rank these candidates exactly in this order. This task is similar to entity recommendation, however, this task is more challenging as it requires to get the same order rather than a relevant set of related entities.

#### 4.2.1 Dataset

In order to compare our approach against existing entity relatedness measures, we performed our experiments on the same gold standard dataset KORE \cite{KORE} that has been used by state of the art methods for calculating entity relatedness.

The KORE dataset consists of 21 seed entities selected from the YAGO knowledge base. Every seed entity has a ranked list of 20 related entities. The seed entities are selected from 4 different domains: IT companies, Hollywood celebrities, video games, and television series. The dataset consists of very popular entities from these 4 domains like Google and IBM as IT companies, Brad Pitt and Angelina Jolie as Hollywood celebrities, Max Payne and Quake as video games, and Futurama and The Sopranos as television series. 20 entity candidates are selected for every seed entity. It is very difficult to judge an absolute relatedness score between two entities, and most of the applications require a ranked list of entities. Therefore, these candidates were given to human evaluators on crowdsourcing to score the relative comparison between two candidates against the corresponding seed entity. For instance, human evaluators provide

\textsuperscript{5}https://lucene.apache.org/
4.2 Evaluation in Ranking Related Entities

<table>
<thead>
<tr>
<th>Entity</th>
<th>Candidates</th>
</tr>
</thead>
<tbody>
<tr>
<td>Apple Inc.</td>
<td>Steve Jobs</td>
</tr>
<tr>
<td>Apple Inc.</td>
<td>Steve Wozniak</td>
</tr>
<tr>
<td>Apple Inc.</td>
<td>Jonathan Ive</td>
</tr>
<tr>
<td>Apple Inc.</td>
<td>Mac Pro</td>
</tr>
<tr>
<td>Apple Inc.</td>
<td>Mike Markkula</td>
</tr>
<tr>
<td>Apple Inc.</td>
<td>Infinite Loop (street)</td>
</tr>
<tr>
<td>Apple Inc.</td>
<td>Silicon Valley</td>
</tr>
<tr>
<td>Apple Inc.</td>
<td>NeXT</td>
</tr>
<tr>
<td>Apple Inc.</td>
<td>Safari (web browser)</td>
</tr>
<tr>
<td>Apple Inc.</td>
<td>FileMaker Inc.</td>
</tr>
<tr>
<td>Apple Inc.</td>
<td>Guy Kawasaki</td>
</tr>
<tr>
<td>Apple Inc.</td>
<td>Microsoft</td>
</tr>
<tr>
<td>Apple Inc.</td>
<td>Ridley Scott</td>
</tr>
<tr>
<td>Apple Inc.</td>
<td>Isaac Newton</td>
</tr>
<tr>
<td>Apple Inc.</td>
<td>Alan Turing</td>
</tr>
<tr>
<td>Apple Inc.</td>
<td>United States Environmental</td>
</tr>
<tr>
<td>Apple Inc.</td>
<td>Protection Agency</td>
</tr>
<tr>
<td>Apple Inc.</td>
<td>Greenpeace</td>
</tr>
<tr>
<td>Apple Inc.</td>
<td>Ginza</td>
</tr>
<tr>
<td>Apple Inc.</td>
<td>Sears</td>
</tr>
<tr>
<td>Apple Inc.</td>
<td>Ford Motor Company</td>
</tr>
</tbody>
</table>

Table 4.2 Example seed entity and its related entity candidates

their judgement if “Mark Zuckerberg” is more related to “Facebook” than “Sean Parker”. With the answers of these types of binary questions, a ranked list was prepared for every seed entity. The KORE dataset consists of 420 entity pairs and their relative semantic relatedness scores. This dataset mainly includes popular entity pairs because getting a judgement about the relatedness between less popular entities may require domain expertise.

4.2.2 Experiments

To determine the effect of taking only the annotated entities for generating a DiSER vector in high dimensional concept space, we performed experiments with a similar model i.e. ESA, which calculates the relatedness between natural language texts by using Wikipedia concepts. The main difference between DiSER
and ESA is that DiSER builds the vector by taking only the annotated entities while ESA takes the full article content. We implemented ESA as it is described in [51]. To generate the DiSER vector, we perform a search for the given entity in our inverted index of entities, and retrieve only the top 1,000 Wikipedia concepts. We chose the top 1,000 articles, as most of the entities in the KORE dataset have less than 1,000 articles in which they occur. The tf-idf score of an entity is used to select the most relevant articles. We apply the same process in building the ESA vector by performing the search in the inverted index of article contents.

In order to compare DiSER with recently popular methods that learn the semantic representation using neural network architecture, we perform experiments with two most popular methods Word2Vec [100] and GloVe [109]. Further, Levy et al. [88] showed that these methods are comparable to the traditional latent model like Latent Semantic Analysis (LSA) [85]. Therefore, we consider these three models to compare DiSER for entity relatedness. We train these models on same Wikipedia corpus that contains 4,102,442 articles after preprocessing. The entity vector is obtained by retrieving the vectors for individual words appear in entity followed by vector addition. Since DiSER is built over Wikipedia corpus that contains only annotated entities, we also train these models on the same entity-tagged Wikipedia corpus. Therefore, we experiment with six methods: Word2Vec, GloVe, LSA, Entity2Vec, Entity-GloVe, and Entity-LSA. Existing methods for calculating entity relatedness utilize context associated with the given entity. We therefore conducted three experiments by utilizing the entity’s context. In order to obtain the entity’s context, an entity linker is required to extract the entities from an external resource. We use Alchemy\textsuperscript{6} to identify Wikipedia entities in given context. However, all of the entities in the KORE dataset have a Wikipedia page, therefore, we used the Wikipedia page to retrieve the context. Since, Wikipedia article contains manually annotated entities, we can also experiment by considering them as context, which eliminates errors produced by the entity linking step. Therefore, we experiment with both the methods of obtaining the context: by extracting entities using Alchemy, and by retrieving the

\textsuperscript{6}http://www.alchemyapi.com/
manually annotated entities. For instance, by manual process, we obtain iPad, iTunes, Steve Jobs, Apple Store, OS X etc. as the context of “Apple Inc.”. We created the context vector by using three different methods: Vector Space Model (VSM), ESA and DiSER. To quantify the entity relatedness, a cosine score between the generated vectors is calculated.

To generate the context vector using VSM, we consider every appearing concept in the context as a dimension. This approach is similar to the best performing state of the art method KPCS [62]. However, KPCS defines the magnitude of the dimensions by using Mutual Information that captures the importance of the concepts for a given entity. In the second experiment, we built the ESA vector of retrieved context by considering a bag of words approach. As ESA does not distinguish between words and entities, we performed a search with each individual word. We identified that some of the retrieved articles by ESA for a given entity “Apple Inc.” were completely irrelevant, which is due to words like “jobs” or “store” appearing in the context of “Apple Inc.”. Similarly, we built the DiSER vector by taking the retrieved context as a bag of concepts.

We computed semantic relatedness scores for all entity pairs provided by the KORE gold standard. We generate scores for all above mentioned methods: Word2Vec, GloVe, LSA, Entity2Vec, Entity-GloVe, Entity-LSA, DiSER, Context-VSM (Manual), Context-ESA (Manual), Context-DiSER (Manual), Context-VSM (Automatic), Context-ESA (Automatic), and Context-DiSER (Automatic). Since the gold standard dataset consists of human judgement about the ranking of 20 entities for each seed entity, we can only quantify the entity relatedness by obtaining similar judgements about the rankings from our experiments. Therefore, we calculated the Spearman Rank correlation between the gold standard dataset and the results obtained from our experiments.

4.2.3 Results and Discussion

Experimental results are shown in Table 4.3 and 4.4. Table 4.3 shows results for three graph-based methods: Wikipedia Link Measure (WLM) [144], Path-
Computing entity relatedness

Table 4.3 Spearman rank correlation of relatedness measures with gold standard DBpedia [67], and Personalize Page Rank (PPR) [10]. WLM calculates the relatedness by applying Normalized Google Distance (NGD) over all the incoming Wikipedia links (see for more details section 3.2.2). Path-DBpedia counts the distance between two given entities in DBpedia graph. PPR applies the personalize page rank over Wikipedia graph to capture the importance of incoming and outgoing links. Further, we compare DiSER against eight corpus-based measures: Word2Vec, GloVe, LSA, Entity2Vec, Entity-GloVe, Entity-LSA, ESA and KORE. KORE [62] represents the keyphrase overlap relatedness which computes the relatedness score by taking cosine similarity on MI-weighted keyphrases appear in the context of a given entity. KORE is a similar approach to Context-VSM. Besides, KORE assigns MI-weights to capture the generality and specificity of entities in context.

Many entities in the gold standard dataset are defined by ambiguous surface forms such as “Next” and “Nice”, or they have ambiguous text segments in their surface forms like “Jobs” in “Steve Jobs” and “Guitar” in the “Guitar Hero” video game. Therefore, the effect of building a distributional vector by only considering annotated entities can be observed with the remarkable difference between the results obtained by ESA and DiSER. Moreover, other models Word2Vec,
GloVe and LSA showed a significant improvement by training them on entity-tagged Wikipedia corpus. The scores illustrate that ESA and other existing methods fail in generating the appropriate distributional vector for ambiguous terms. Table 4.3 also showed that corpus-based measures are better than graph-based measures, which could be because the knowledge in a graph is very much dependent on the corresponding text, and its hard to encode all the information from text into a graph. While Word2Vec and GloVe usually generates very high correlation for word similarity task, it is surprising that they perform very poorly for computing entity relatedness. The possible reason behind it could be that these methods are based on neural network architectures which mainly rely on very small context window and tend to find very closely similar words but not related. Similar to the results reported in original ESA paper [52], LSA could not perform well because it learns the latent topics by dimensionality reduction, and do not make use of the manually defined topics. Table 4.4 shows comparison 

<table>
<thead>
<tr>
<th>Entity Relatedness Measures</th>
<th>Spearman Rank Correlation</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Manual</td>
</tr>
<tr>
<td>Context-VSM</td>
<td>0.637</td>
</tr>
<tr>
<td>Context-ESA</td>
<td>0.684</td>
</tr>
<tr>
<td>Context-DiSER</td>
<td>0.769</td>
</tr>
</tbody>
</table>

Table 4.4 Spearman rank correlation of relatedness measures with gold standard between different methods that calculates relatedness by considering context of the given entities. Context-VSM computes relatedness score between entities context, where entity context is represented by bag-of-entities appear in the corresponding Wikipedia article. Similarly, Context-ESA and Context-DiSER are the approaches that calculate an ESA score and DiSER score between entity context vectors respectively. We also analyze if the error generated by Alchemy in automatic entity linking has any effect in the overall performance of context-based entity relatedness. Context-VSM does not capture the semantics of entities appearing in the context and calculates the relatedness scores by taking the overlap between these entities. Context-ESA creates the distributional vector of context, therefore, it improves accuracy over Context-VSM. Context-DiSER
generates significantly higher scores, which shows that considering annotated
tentities in the context may reduces the issues of ambiguity and compositionality
in entity names. Although, the overall performance of all methods drop by using
automatic entity linking to obtain the context, Context-DiSER (Automatic) still
performs better than the best performing existing method KORE. Moreover, we
show that Context-DiSER outperforms KORE and improves the accuracy of the
text relatedness measure significantly. KORE achieved higher score in compar-
ison to Context-VSM, which indicates that generality and specificity of entities in
the context are very influential features for entity relatedness measures. KORE
captures the semantics of an entity by considering entities or keyphrases in the
context. However, the entities in the context do not cover enough background
knowledge to define the given entity. Therefore, it leads to the problem of topic
mismatch in vector comparison. For instance, “Apple llc” does not occur in the
“Apple Inc.” Wikipedia article. However, the DiSER vector of the context of “Ap-
ple Inc.” retrieves “Apple llc” as a dimension. On the other hand, KORE cannot
capture the weakly related entities whose appropriate relatedness can be quant-
tified only by considering a greater amount of background knowledge. This can
be a major reason for getting significant improvement over KORE, as Context-
DiSER is able to capture the relatedness between weakly related entities such as
“Microsoft” and “Helmut Panke”.

DiSER achieved the best correlation with human ranking in this dataset. How-
ever, Context-DiSER also achieved similar accuracy. The KORE dataset consists
of popular entities, therefore DiSER can exploit the distributional knowledge
from Wikipedia. However, it may fail to obtain significant background informa-
tion about the long tail entities. Instead, Context-DiSER may perform better by
building the distributional vector of popular entities found in the context of the
long tail entity.

4.3 Evaluation in Entity Disambiguation

Entity disambiguation is a subtask of entity linking [57]. Entity linking usually includes two major steps: mention recognition and entity disambiguation. Mention recognition is a task of finding the potential spans of text which can be mapped to an entity in knowledge bases. Entity Disambiguation can be defined as disambiguating an ambiguous mention to an entity defined in a knowledge base. A named entity recognition tool like Stanford Named Entity Recognition (NER) [46] can be used to obtain the mentions. Moreover, some of the entity linking methods [40, 132] perform mention finding and disambiguation at the same time. However, in open domain entity linking, Stanford NER performs with 97% of accuracy for mention recognition for selected entity types. Therefore, the research community focuses more towards the entity disambiguation problem [72, 73]. Classical approaches [96, 128] rely on textual similarity and calculate similarity scores between the context around the mention in text and the context of an entity in a knowledge base. However, all the mentions appearing in the text can be considered related to each other as a text document or a sentence usually focuses on related topics. This leads to collective inferencing for entity disambiguation that considers all the entities appearing in a text to be related. Cucerzan [31] introduced the method of collective inferencing in entity disambiguation that aims at joint mapping of several entities together in a related entity space by using entity relatedness measures. For instance, there are three mentions “Desire”, “Harris” and “Joey” in a sentence “Desire contains a duet with Harris in the song Joey.”; these mentions would be mapped to “Desire (Bob Dylan album)”, “Emmylou Harris” and “Joey (Bob Dylan song)” as they are related to each other. Many different methods [31, 63, 84] based on collective inference mapping have been proposed. Most of the methods rely on the quality of an entity relatedness measure. AIDA [63] and KORE [62] uses three different features: prior probability, text similarity and entity relatedness. These features are combined by using a supervised leaning method such as Support Vector Machine (SVM) [30]. However, we present an unsupervised approach to perform entity disambiguation that does not require any training data.
4.3.1 Dataset

We performed experiments to evaluate if our approach for entity relatedness can improve the accuracy of state of the art methods which use entity relatedness measures for the entity disambiguation task. Hoffart et al. [62] showed that different entity relatedness measures obtained significant differences in accuracy for short sentences in comparison to long news documents. Therefore, we used the KORE50 [62] dataset which consists of 50 short sentences with highly ambiguous mentions. There are only 14 words and nearly 3 mentions per sentence on average. Every mention has around 631 candidates on average to disambiguate. Sentences also contain non-popular entities which have very few incoming links. As we evaluated our approach for entity disambiguation, we assume that all the mentions are given.

4.3.2 Experiment

We applied different entity relatedness measures to calculate relatedness scores of all the candidates of a mention to the candidates of other mentions. For instance, to find entities for the mentions “Desire”, “Harris” and “Joey”, we calculate relatedness scores of all the candidates of “Desire” to all the candidates of “Joey” and “Harris”. We use the “AIDA-Means” [63] dictionary to find out the candidates for a mention; it contains 35 candidates for “Desire”, 267 candidates for “Joey” and 1043 candidates for “Harris”. We obtain the confidence scores for each set of candidates by multiplying the relatedness scores of individual candidate pairs. As an example computation, we multiply the relatedness scores of “Desire (Bob Dylan album)” and “Harris (Emmylou Harris)”, “Desire (Bob Dylan album)” and “Joey (Bob Dylan song)”, and “Harris (Emmylou Harris)” and “Joey (Bob Dylan song)”, to get the final confidence score for the candidate set {“Desire (Bob Dylan album)”, “Harris (Emmylou Harris)”, “Joey (Bob Dylan song)”}. In order to get the best set of entities for the above example, we need to compute 9.7 million (35x267x1043) confidence scores. We evaluated three different approaches of entity relatedness to jointly map the entities: Joint-Context-VSM, Joint-ESA and Joint-DiSER, which use Context-VSM, ESA and DiSER respectively for entity relatedness. Since these approaches need to calculate relatedness
scores between all the candidates, it may take a long time to compute the confidence scores for very ambiguous mentions. For instance, we need to calculate more than 3 billion confidence scores for the mentions “Steve”, “Bill”, “Sergey” and “Larry” in a given sentence “Steve, Bill, Sergey, and Larry have drawn a great deal of admiration these days for their pioneering successes that changed the world we live in”. Additionally, these approaches only make use of the mentions and their candidates, thus not utilizing the text around the mentions. Therefore, we propose a layered approach of entity disambiguation that performs a candidate selection before collective inferencing. Candidate selection is performed by calculating mention-entity relatedness scores. We rank all the candidates of a given mention by calculating the ESA score between the candidate and the text around that mention in the given sentence. As the dataset contains short sentences, we take the full sentence as the context of the mention. We select the top 10 candidates for each mention and perform joint mapping by using Joint-DiSER, which we refer to as Joint-DiSER-TopN. The candidate selection process selects the top 10 candidates for each mention in the sentence “Steve, Bill, Sergey, and Larry have drawn a great deal of admiration these days for their pioneering successes that changed the world we live in”, by using ESA ranking. This selection process reduces the total number of computations in entity relatedness from 3 billion to 10,000.

### 4.3.3 Results and Discussion

Table 4.5 shows the results of our approach for entity disambiguation. Results show that different entity relatedness measures effect the accuracy of the entity disambiguation task. Joint-Context-VSM, Joint-ESA and Joint-DiSER achieved an accuracy in the same order as that of the entity ranking task, which demonstrates a consistency in results. AIDA [63] combines three different features: popularity, mention-entity relatedness, and entity-entity relatedness. AIDA-WLM [63], AIDA-KORE and AIDA-KPCS [62] use WLM [144], KORE, KPCS respectively, to perform entity relatedness. Although, AIDA-WLM and AIDA-KPCS use entity relatedness with a combination of several other features, Joint-DiSER outperforms these approaches. This shows that DiSER stands as an important fea-
ture in performing entity disambiguation. Joint-DiSER-TopN achieved the best precision and improved around 10% over state of the art methods, which shows that performing disambiguation in filtered candidates affects the performance significantly. As Joint-DiSER-TopN performs disambiguation only for selected candidates, it reduces the performance time considerably. For instance, it computes only 10K confidence scores in comparison to 3.1 billions scores for the mentions “Steve”, “Bill”, “Sergey”, and “Larry”.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Joint-Context-VSM</td>
<td>35.42%</td>
<td>34.66%</td>
</tr>
<tr>
<td>Joint-ESA</td>
<td>52.41%</td>
<td>51.74%</td>
</tr>
<tr>
<td>Joint-DiSER</td>
<td>58.33%</td>
<td>57.45%</td>
</tr>
<tr>
<td>Joint-DiSER-TopN</td>
<td><strong>70.83%</strong></td>
<td><strong>70.10%</strong></td>
</tr>
<tr>
<td>AIDA-WLM</td>
<td>57.64%</td>
<td>56.00%</td>
</tr>
<tr>
<td>AIDA-KORE</td>
<td>64.58%</td>
<td>62.60%</td>
</tr>
<tr>
<td>AIDA-KPCS</td>
<td>55.64%</td>
<td>54.70%</td>
</tr>
</tbody>
</table>

Table 4.5 Entity Disambiguation accuracy on KORE50 dataset

4.4 Evaluation in Top-N books recommendation

Top-N recommendation systems deal with finding a set of N items that best match a user profile. The user profile is defined by a few liked and disliked items. We assume that the user might like the items that are similar or related to his/her liked ones. Therefore, the task can be defined as to find out a ranked list of items which are most related to the liked items. In recent years, there have been several efforts [35, 36] in utilizing external knowledge bases such as Wikipedia and DBpedia for item recommendation such as movies and books. Most of this work focuses on boosting collaborative filtering approaches or to improve content-based recommendation systems [108]. In particular, they have shown some benefits in solving cold start and data sparsity issues in conventional collaborative filtering methods. Di Noia et al. [36] have shown the effectiveness of using Linked Open Data in boosting collaborative filtering for Top N movies recommendation. Therefore, we evaluate DiSER in a Top-N books recommendation as it requiring semantic relatedness scores of a given book with the liked
and disliked books. As explained above, DiSER computes the relatedness between Wikipedia entities, and every book can be seen as a Wikipedia entity if it contains a corresponding Wikipedia page.

### 4.4.1 Dataset

We perform experiments on a dataset provided by the “Linked Open Data-enabled Recommender Systems” challenge organized at ESWC 2014. There were three different tasks, where task 2 was “Top-N recommendation from binary user feedback”. The dataset consists of 6,181 users, 8,171 books and 67,990 user-item pairs to predict the rating. All the books contain their corresponding DBpedia and Wikipedia links.

<table>
<thead>
<tr>
<th>Relatedness</th>
<th>Precision</th>
<th>Recall</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Average</td>
<td>ESA</td>
<td>0.6132</td>
<td>0.4573</td>
</tr>
<tr>
<td></td>
<td>Context-VSM</td>
<td>0.6149</td>
<td>0.4562</td>
</tr>
<tr>
<td></td>
<td>DiSER</td>
<td><strong>0.6254</strong></td>
<td><strong>0.4718</strong></td>
</tr>
<tr>
<td>Maximum</td>
<td>ESA</td>
<td>0.6115</td>
<td>0.4587</td>
</tr>
<tr>
<td></td>
<td>Context-VSM</td>
<td>0.6152</td>
<td>0.4551</td>
</tr>
<tr>
<td></td>
<td>DiSER</td>
<td><strong>0.6241</strong></td>
<td><strong>0.4683</strong></td>
</tr>
<tr>
<td>Random</td>
<td>ESA</td>
<td>0.6147</td>
<td>0.4599</td>
</tr>
<tr>
<td></td>
<td>Context-VSM</td>
<td>0.6165</td>
<td>0.4588</td>
</tr>
<tr>
<td></td>
<td>DiSER</td>
<td><strong>0.6271</strong></td>
<td><strong>0.4703</strong></td>
</tr>
</tbody>
</table>

Table 4.6 Top N books recommendation

### 4.4.2 Experiment

The user profile is defined by some liked and disliked books. The task is to find out a ranked list of N other books that the user might like. We compute the relatedness score of a given book with the liked and disliked books. We put a book in the recommendation list if its relatedness score with the liked books is greater than the disliked ones. Since the user can like or dislike more than one book, we need to aggregate the relatedness scores to obtain the confidence score.

---

8http://challenges.2014.eswc-conferences.org/index.php/RecSys
for like and dislike prediction. Therefore, we present following three methods to aggregate the relatedness scores.

1. **Average**: We calculate relatedness scores of the given book with all the books liked by corresponding user, and obtained a confidence score by taking an average of these scores. Similarly, we calculate the confidence score for disliked items.

2. **Maximum**: We compute the relatedness scores of the given book with all the books liked by corresponding user, but choose the confidence score as the relatedness score of the most related pair rather than taking the average. Similarly, we calculate the confidence score for disliked items.

3. **Random**: We randomly select one book from all the books liked by the corresponding user, and one from all the disliked ones. We calculate relatedness scores of the given book with a randomly selected liked book and randomly selected disliked book. The obtained relatedness scores are considered as confidence scores for the corresponding classes.

In order to compare DiSER with ESA and Context-VSM, the experiments are performed with these three relatedness measures for the above mentioned aggregation methods: Average, Maximum and Random.

### 4.4.3 Results and Discussion

Table 4.6 shows the results for top N recommendation. The top 5 recommendations are considered by computing Precision@5, Recall@5 and F1@5. Results show that DiSER outperforms the other two relatedness measures. All three approaches of finding top N recommendations are comparable and do not show a major difference in scores. It demonstrates that our approach “Random” can work with sparse data as it requires just one liked and disliked item. Figure 4.2 illustrates that our approach “UNLP-Insight@NUIG” performs better than other 15 approaches submitted to the ESWC Top-N recommendation challenge⁹.

⁹We could not participate in the task, we evaluated our approach by using the evaluation service provided by the organizers.
4.4 Evaluation in Top-N books recommendation

Fig. 4.2 F1 scores of different systems submitted to ESWC books recommendation challenge

Although, we experiment with Top N recommendation to compare our entity relatedness measures, it is interesting that the results show a simple unsupervised approach which relies only on entity relatedness, performs better than complicated supervised approaches like SPrank [106] and LODify [97]. SPrank extracts several features from DBpedia and combined them using learning to rank. On the other hand, we can see in figure 4.2 that other supervised approaches [17, 112, 127] achieve higher scores than our approach, which shows the importance of relevant features selection in learning algorithms. Basile et al. [17] obtains the highest scores in the task. This method aggregates the different relevancy scores obtained by multiple trained models based on popularity, random forests, logistic regression, and Pagerank. Ristoski et al. [127] also trained the model by using random decision trees on different features extracted from DBpedia such as book types, book categories, author types, author categories, other books by the same authors. It can be concluded that trained model in general achieved higher scores in the task, however, our unsupervised method also performs reasonably well.
4.5 Efficiency

To evaluate the efficiency of DiSER, we compare its running time with Context-VSM and WLM. The efficiency of Context-VSM is very similar to state of the art method KORE as Context-VSM and KORE both compute the relatedness scores by taking an overlap between entities’ contexts. We tested the running time of each method on above mentioned example of finding entities for the mentions “Desire”, “Harris” and “Joey” in a sentence “Desire contains a duet with Harris in the song Joey”. We obtain 35 candidates for “Desire”, 267 candidates for “Joey” and 1043 candidates for “Harris”. To obtain the best set of candidates, we compute 9.7 million (35x267x1043) confidence scores. Table 4.7 shows the time taken by each method. We can see Context-VSM is very fast, as it only needs to compute the overlap between corresponding entities context, which is Wikipedia article in this experiment. On the other hand, DiSER takes more time in generating the distributional vectors of entities and computing the vector cosine between them. WLM is a little faster than DiSER, as WLM does not generate the vectors and only counts the overlapping incoming links. We further investigate the DiSER performance and found that obtaining a vector from Lucene index is a time consuming process. Therefore, we perform another experiment by uploading all the pre-trained vectors in memory and run the same experiment again. We can see in the results table 4.7 that in memory DiSER runs remarkably faster. Although, we do not have a running time of KORE on this experiment, we can get an estimation of KORE for performing 9.3M comparison. Hoffart et al. [62] reported an average running time of KORE for performing 8.98K comparisons in 1.884 sec, which means KORE will take around 19.5 sec to perform 9.3M comparisons. It shows that KORE is a little slower than Context-VSM, which could be because

<table>
<thead>
<tr>
<th>Entity Relatedness Measures</th>
<th>Running Time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Context-VSM</td>
<td>16.258</td>
</tr>
<tr>
<td>WLM</td>
<td>52.142</td>
</tr>
<tr>
<td>DiSER</td>
<td>58.332</td>
</tr>
<tr>
<td>DiSER (in-memory)</td>
<td>23.439</td>
</tr>
</tbody>
</table>

Table 4.7 Results of efficiency experiments
KORE also performs key-phrase extraction to calculate the overlap. We perform all these experiments on a machine consisting of 2.3 Ghz core i5 processor with 8GB of RAM.

4.6 Summary

This chapter presented the DiSER approach for computing semantic relatedness between entities. We used Wikipedia as it consists of the world knowledge about millions of entities. DiSER builds distributional vectors by considering only the manually annotated entities appearing in Wikipedia articles. Therefore, it can build unambiguous distributional vectors for ambiguous surface forms of the given entities. In our experiments, DiSER outperforms state of the art methods and achieves a significant improvement in entity ranking and disambiguation tasks over other methods. We also proposed an alternative Context-DiSER approach to generate the DiSER vectors for long tail and non-popular entities, which do not have a Wikipedia page. Moreover, we evaluated DiSER in top-N book recommendation and showed that it is an effective solution in cold-start situations.

DiSER relies on collection of disambiguated entities which requires manual annotations to prepare a corpus. Although, Wikipedia contains millions of popular entities annotated by volunteers, it may not have much information about very specific diseases and symptoms. Therefore, we need to generate a similar high quality annotated corpus for domain specific applications. Context-DiSER can deal with non-Wikipedia entities like specific diseases, however, the overall performance may suffer from the error generated by the entity linking tool in obtaining the appropriate context. We showed that DiSER outperforms other existing methods of computing entity relatedness. However, evaluation is performed on KORE dataset, which mainly consists of popular entities. Therefore, it is hard to make any conclusions about the performance of the methods for long tail entities. Moreover, preparing a dataset about the relatedness of long tail entities is relatively difficult, as due to limited common sense knowledge, it may
lead to very low inter annotator agreement in judging the relatedness between them. Further, the degree of relatedness for entities may change over time. For instance, a Hollywood actor or director might be very related to their current movie but after a while their relatedness can completely change due to another upcoming movie. Therefore, current evaluations can not satisfy all the characteristics of entity relatedness. In order to address the performance over long tail entities, we also evaluate our approach in recommending specific locations like museums and monuments, in chapter 5.
Chapter 5

Leveraging Wikipedia Knowledge for Entity recommendation

With the advent of large knowledge bases like DBpedia [13], YAGO [136] and Freebase [24], search engines have started recommending entities related to web search queries. Pound et al. [120] reported that more than 50% of web search queries pivot around a single entity and can be linked to an entity in a knowledge base. Consequently, the task of entity recommendation in the context of web search can be defined as finding entities related to the entity appearing in a web search query. It is very intuitive to get related entities by obtaining all the explicitly linked entities to a given entity in a knowledge base. However, most of the popular entities can easily have more than 1,000 directly connected entities, and a knowledge base mainly tends to cover some specific types of relations. For instance, “Tom Cruise” and “Brad Pitt” are not directly connected in the DBpedia graph through any relation, however, they can be considered related to each other as they both are popular Hollywood actors and co-starred in movies. Therefore, to build a system for entity recommendation, there is a need to discover related entities beyond the relations explicitly defined in a knowledge base. Furthermore, these related entities require a ranking method to select the most related ones.

Blanco et al. [20] described the “Spark” system for related entity recommendation and suggested that such recommendations are successful at extending
users’ search sessions in Yahoo! search. Microsoft also published a similar system [151] that performs personalized entity recommendation by analyzing user click logs. In this chapter, we describe our approaches of entity recommendation. We present two methods: Entity Relatedness Graph (EnRG) and Wikipedia Features for Entity Recommendation (WiFER). EnRG is a very large graph that provides lists of related entities. EnRG is built by using DiSER only, therefore, it is an unsupervised approach and does not require any training data. However, WiFER is a supervised approach that combines different features extracted from Wikipedia, including DiSER. It uses learning to rank to combine all the features. WiFER is inspired by an existing commercial entity recommendation system called Spark [20] used in Yahoo! search. However, Spark utilizes proprietary data like query logs and search session which are not available publicly. Therefore, we focus on exploring different features in entity recommendation system and investigate their effectiveness. Moreover, we present an extensive study about the importance of different features and their comparisons. In order to evaluate if WiFER can complement Spark, we combine all the features from WiFER and Spark and name it SparkWiki [7].

5.1 Entity recommendation

This section provides the detailed overview of all the entity recommendation systems that will be discussed in this chapter, EnRG, WiFER, Spark, and SparkWiki.

5.1.1 Entity Relatedness Graph (EnRG)

EnRG is constructed by calculating the entity relatedness scores between every entity pair by using DiSER. Wikipedia\footnote{Snapshot of the English Wikipedia from October 2013} contains more than 4.1 million entities. Therefore, to build the EnRG graph, we calculate relatedness scores between 16.8 trillions (4.1 million \times 4.1 million) of entity-pairs. It can be seen as a sparse square matrix of order 4.1 million. To reduce the number of computations, we keep only the top 1000 dimensions in the DiSER vector according to
their associativity scores with the entity, which converges the remaining dimensions’ scores to zero.

In order to produce 16.8 trillion scores, a very fast and efficient computing is required. We applied a pruning technique which only calculates the DiSER score if it would be a non-zero value. We collect all the possible related entities with non-zero scores for a given entity. Since we take only the top 1,000 articles to build the vector, the entities not appearing in the content of the top 1,000 articles for a given entity would produce a zero relatedness score with that entity. For instance, if DiSER takes only the top 2 articles to calculate the relatedness score, and we want to retrieve all the entities having a non-zero relatedness score with “Apple Inc.”, we would obtain entities such as “Steve Jobs”, “iPad” and “OS X” as they appear in the content of the top 2 articles of “Apple Inc.”, while we would miss entities like “Samsung” and “Motorola” as they do not appear in the top 2 articles. We obtain around 10K related entities for every individual entity in EnRG. Therefore, we calculate DiSER scores for only 4.1 billion entity-pairs, and this reduces the comparisons by 99.8%. It takes around 48 hours to build the EnRG graph with 25K comparisons per second.

### 5.1.2 Wikipedia Features for Entity Recommendation (WiFER)

Wikipedia Features for Entity Recommendation (WiFER) combines the different features extracted from Wikipedia that provide evidence of the relevance of an entity. The final relevance scores are calculated by combining the different features using a state of the art learning to rank approach. The features are extracted from Wikipedia by considering two different types of data source: the collection of textual content and the collection of Wikipedia anchors. These features are derived from the hypothesis that the entities, which occur often in the same event or context, are more likely to be related to each other. Let $E_1$ and $E_2$ be two entities and $S$ is the set of events, where $S = \{s_1, s_2, ..., s_n\}$ and $s_n$ is the $n^{th}$ event. The event is defined as one observation under consideration for measuring the occurrence. For instance, every Wikipedia article is an event to measure the occurrence of an entity $E$. We use 7 different types of features: Probability
(\(P_1, P_2\)), Joint Probability (JPSYM), Conditional Probability (CPASYM), Cosine Similarity (CSSYM), PMI (SISYM), Reverse Conditional Probability (RCPASYM) and DSM.

1. **Probability** \((P_1, P_2)\) is calculated by taking the ratio of the number of events that contain the given entity to the total number of events. \(P_1\) is the probability of an entity \(E_1\).

\[
P_1 = \frac{\sum_{i=0}^{N} o_i}{N}
\]

where \(o_i = 1\), if an event \(s_i\) contains the entity \(E\) otherwise \(o_i = 0\). \(N\) is the total number of events. The value of \(P\) of an entity is independent of the other entities, therefore it gives two values \(P_1\) and \(P_2\) for an entity pair consisting of \(E_1\) and \(E_2\).

2. **Joint probability (JPSYM)** is obtained by taking the ratio of the number of events that contain both the given entities to the total number of events.

\[
JPSYM = \frac{\sum_{i=0}^{N} co_i}{N}
\]

where \(co_i = 1\) if an event \(s_i\) contains both the entities \(E_1\) and \(E_2\), otherwise \(o_i = 0\).

3. **PMI (SISYM)** computes point wise mutual information (PMI).

\[
PMI(E_1, E_2) = \frac{\log(P(E_1, E_2))}{P(E_1) \ast P(E_2)}
\]

where \(P(E_1)\) and \(P(E_2)\) are the prior probabilities as described above. \(P(E_1, E_2)\) is computed by taking the ratio of number of events that contain both the entities \(E_1\) and \(E_2\), to the total number of events.

4. **Cosine similarity (CSSYM)** is calculated as

\[
\text{Cosine}(E_1, E_2) = \frac{P(E_1, E_2)}{P(E_1) \ast P(E_2)}
\]
5.1 Entity recommendation

5. **Conditional probability (CPASYM)** is calculated as the ratio of the total number of events that contain $E_1$ and $E_2$, to the total number of events that contain $E_1$.

$$CPASYM(E_1, E_2) = \frac{\sum_{i=0}^{N} co_i}{\sum_{i=0}^{N} oe_{1i}}$$  \hspace{1cm} (5.5)

where $oe_{1i} = 1$ if an event $s_i$ contains the entity $E_1$, otherwise $oe_{1i} = 0$.

6. **Reverse conditional probability (RCPASYM)** is the reverse of CPASYM.

$$RCPASYM(E_1, E_2) = \frac{\sum_{i=0}^{N} co_i}{\sum_{i=0}^{N} oe_{2i}}$$  \hspace{1cm} (5.6)

where $oe_{2i} = 1$ if an event $s_i$ contains the entity $E_2$, otherwise $oe_{1i} = 0$.

7. **Distributional Semantic Model (DSM)** builds distributional vector over all the events as described in Chapter 4. DSM computes the values by taking a cosine score between the distributional vectors. Therefore, similar to the above described features, it relies on co-occurrence information. However, other features only consider the presence of an entity in the events whereas DSM measures the importance of an entity to a given event in addition to its presence.

Since we mentioned that Wikipedia is used as two different data sources, WiFER generates 16 different feature values. In order to generate the feature values from the text collection, we consider only the surface form of an entity to obtain the occurrence value. However, we count the occurrence of an entity in the collection of Wikipedia anchors, only if the corresponding anchor appears in an article (event). The Probability feature generates two values for an entity pair, therefore, each data source provides 8 different feature values and we obtain total 16 values.

5.1.3 Spark

Spark is a commercial entity recommendation system that is currently powering Yahoo! search. Spark utilizes more than 100 different features extracted from
different data sources. Similar to WiFER, the final relevance scores are calculated by combining different features using a learning to rank approach. The features used by the Spark system can be divided into five types: co-occurrence based, linear combination of co-occurrence based features, graph-based, popularity-based, and type-based features. Co-occurrence based features make use of four different data sources: query term, user-specific query sessions, Flickr tags, and tweets. Since WiFER takes inspiration from Spark, many features are computed in the same way. The major difference is that WiFER uses only publicly available data sources, however, Spark utilizes the proprietary data sources such as query logs and query sessions. Moreover, WiFER contains only co-occurrence based features whereas Spark has five different types of features from four different data sources.

Section 5.1.3.1 describes the construction of the Yahoo! knowledge graph, which is used to obtain the potential entity candidates. Section 5.1.3.2 explains different types of features and how they are extracted from different data sources. Spark and SparkWiki combine the values obtained from different features, by using a learning to rank approach, which is explained in Section 5.1.4.

5.1.3.1 Yahoo! knowledge graph

In order to retrieve a ranked list of the entities, the system requires a list of potential entity candidates that can be considered related with the given entity. These candidates can be obtained from existing knowledge bases like DBpedia or YAGO. However, such existing knowledge bases may not cover all the relations that can be defined between the related entities. For instance, “Tom Cruise” can be considered highly related to “Brad Pitt”, but they are not connected by any relation in the DBpedia graph. Therefore, Spark uses an entity graph extracted from different structured and unstructured data sources. The entity graph uses DBpedia, Freebase and YAGO as the structured data sources. It also uses a manually constructed ontology that defines the types of an entity extracted from different resources. In order to extend the coverage of the defined relations in entity graph, it performs information extraction over various unstructured data
sources in different domains like movies, music, TV shows and sports. This entity graph consists of over 3.5 M entities and 1.4 B relations (see for more detail [20]). Thus, Spark retrieves a much bigger set of potential related entities for a given entity by using this entity graph.

5.1.3.2 Feature extraction

Co-occurrence features are derived from the hypothesis that entities, which occur often in the same event or context, are more likely to be related to each other. Spark uses 11 different types of features which are obtained by using different co-occurrence measures. Out of these 11 features, 6 are described above in section 5.1.2. Spark does not use DSM features which are used in WiFER.

1. **Entropy** ($Ent_1, Ent_2$) This is the standard entropy of an entity that is defined by

   \[ Ent_1 = -P_1 \ast \log(P_1) \]  

   and $P$ is the probability defined in feature 1 in section 5.1.2. Similar to the probability feature, it gives two values $Ent_1$ and $Ent_2$ for an entity pair.

2. **KL** ($KL_1, KL_2$) It is KL divergence of an entity $E$. Similar to the above features, it also gives two values $KL_1$ and $KL_2$ for an entity pair.

3. **Joint user probability** (JPUSYM) This is similar to the Joint probability described in WiFER, however, it calculates the co-occurrence over users rather than events.

   \[ JPUSYM = \sum_{i=0}^{U} \frac{cou_i}{U} \]  

   where $U$ is the total number of users and $cou_i = 1$ if a user $u_i$ contains both the entities $E_1$ and $E_2$, otherwise $cou_i = 0$.

4. **Conditional user probability** (CUPASYM) This is similar to the conditional probability described in WiFER, except it computes the score over
Leveraging Wikipedia Knowledge for Entity recommendation

the users.

\[ CUPASYM(E_1, E_2) = \frac{\sum_{i=0}^{U} con_i}{\sum_{i=0}^{U} oue_{1i}} \]  

(5.9)

where \( oue_{1i} = 1 \) if an user \( u_i \) contains the entity \( E_1 \), otherwise \( oue_{1i} = 0 \).

5. **Reverse conditional user probability (RCUPASYM)** is the reverse of the Conditional user probability (CUPASYM).

\[ RCUPASYM(E_1, E_2) = \frac{\sum_{i=0}^{U} con_i}{\sum_{i=0}^{U} oue_{2i}} \]  

(5.10)

where \( oue_{2i} = 1 \) if an user \( u_i \) contains the entity \( E_2 \), otherwise \( oue_{1i} = 0 \).

Combined features are the combination of co-occurrence features. The Spark system uses 8 different types of combined features from every data source. Therefore it generates a total of 32 different features. These are the following 8 features:

1. **CF1** is the combination of conditional user probability and prior probability of a target entity defined by:

\[ CF1 = CUPASYM \times P_2 \]  

(5.11)

2. **CF2** is the combination of conditional user probability and prior probability of a target entity defined by:

\[ CF2 = \frac{CUPASYM}{P_2} \]  

(5.12)

3. **CF3** is the combination of reverse conditional probability and prior probability of a target entity defined by:

\[ CF3 = RCUPASYM \times P_2 \]  

(5.13)

4. **CF4** is the combination of reverse conditional probability and entropy of a
target entity defined by:

\[ CF_4 = RCPASYM \ast \text{Ent}_2 \] (5.14)

5. **CF5** is the combination of joint user probability and prior probability of a target entity defined by:

\[ CF_5 = JPUSYM \ast P_2 \] (5.15)

6. **CF6** is the combination of joint user probability and prior probability of a target entity defined by:

\[ CF_6 = \frac{JPUSYM}{P_2} \] (5.16)

7. **CF7** is the combination of joint user probability and entropy of a target entity defined by:

\[ CF_7 = JPUSYM \ast E_2 \] (5.17)

8. **CF8** is the combination of joint user probability and entropy of a target entity defined by:

\[ CF_8 = \frac{JPUSYM}{E_2} \] (5.18)

**Graph-based features** use knowledge graphs like DBpedia and Freebase. Spark computes 5 different features by using knowledge graphs.

1. **Graph similarity (GSCEG)** This feature computes the total shared connections between two given entities in the Yahoo! knowledge graph.

2. **Entity popularity in movies (EPOPU MOVIE)** This feature counts the total number of directly connected nodes in a movie specific knowledge graph, to compute the entity popularity rank.

3. **Facet popularity in movies (FPOPU MOVIE)** This is facet popularity rank in a movie specific knowledge graph.
4. **Entity popularity in all (EPOPUALL)** Similar to \textit{EPOPUMOVIE} it counts the total number of directly connected nodes in the complete Yahoo! knowledge graph.

5. **Facet popularity in all (FPOPUALL)** This is facet popularity rank in the complete knowledge graph.

### Popularity-based features

1. **Web search citation (WCTHWEB)** counts the total hits in web search results of Yahoo!.

2. **Web deep citation (WCDHWEB)** counts the total number of user clicks in web search results of Yahoo!.

3. **Entity Volume in query (COVQ)** counts the total number of occurrence of a given entity in query logs.

4. **Entity Volume in facet (COVF)** Facet volume in query logs.

5. **Entity view volume in query \((WPOP_1, WPOP_2)\)** computes the total number of user clicks for a given entity while the entity occurs in a query.

### Entity type features

Entity type features reflect the entity types and relation types present in the knowledge base used. Spark uses two different entity type features:

1. **Entity class type \((ET_1, ET_2)\)** This is the type of an entity defined in the knowledge base. It provides two different feature values \(ET_1\) and \(ET_2\) for an entity pair of the entities \(E_1\) and \(E_2\).

2. **Relation type (RT)** This feature defines the relation type between two given entities. For instance, “Brad Pitt” and “Angelina Jolie” are defined by the relation type “Partner” in DBpedia.

Spark contains 112 features in total where 56 features are co-occurrence based, 32 features are the linear combination of co-occurrence based features, 5 features are graph-based, 6 features are popularity-based, 3 features are type-based, and the remaining 10 features are of types such as string length and Wikipedia
clicks. These 56 co-occurrence based features are built over 4 different data sources: query term (QT), query session (QS), Flickr tags (FL), and tweets (TW). It means that there are 14 co-occurrence based features generated from each data source.

5.1.3.3 SparkWiki

SparkWiki is the combination of WiFER and Spark. Therefore, it contains all the features that we described above. It contains 128 feature values of which 112 are coming from Spark and 16 from WiFER. With SparkWiki, we can investigate which features and sources are the most influential to achieve the best performance.

5.1.4 Ranking

In order to predict the ranking by combining all the features, we use a learning to rank approach by considering all the scores obtained from different features. As all learning algorithms require training data, Blanco et al. [20] built a dataset that contains more than four thousand web search queries. Every query refers to an entity defined in the knowledge graph, and contains a list of entity candidates with their appropriate label of relevance. The ranking can be defined by learning a ranking function \( f(.) \) that generates a score for an input query entity \( q_i \) and an entity candidate \( e_j \). We use Stochastic Gradient Boosted Decision Trees (GBDT) to obtain the ranking score to decide the appropriate label for given pairs.

5.2 Evaluation

This section describes the evaluation of the above described entity recommendation systems. Since, this chapter presents an unsupervised method EnRG and a supervised method WiFER, we perform experiments for both the scenarios. We compare EnRG against two other unsupervised approaches: GraphSim and ESASim. Similar to WLM [144], GraphSim computes the relatedness scores by using Normalized Google Distance [28] on incoming and outgoing links in DB-
Leveraging Wikipedia Knowledge for Entity recommendation

pedia. ESASim calculates the relatedness by using ESA model [52]. To evaluate our supervised approach, we compare WiFER against Spark and SparkWiki. We use a dataset that consists of 47,623 query-entity pairs. As GBDT has different parameters, we tune the parameters by using a small set of instances as development dataset which consists of 50 entities and their potential candidates. Finally, we set number of regression trees, number of nodes, shrinkage rate, and sampling rate 60, 25, 0.1 and 1 respectively. Due to variations in the number of retrieved related entities for a query, we use Normalized Discounted Cumulative Gain (nDCG) for the performance metric. nDCG is defined by the ratio of DCG to the maximum or ideal DCG.

\[
nDCG_p = \frac{DCG_p}{IDCG_p},
\]

(5.19)

DCG is defined by:

\[
DCG_p = \sum_{i=1}^{p} \frac{2g(l_i) - 1}{\log_2(g(l_i)) + 1}
\]

(5.20)

\(g(l_i)\) is the gain for the label \(l_i\). nDCG gives different scores on different values of \(p\), therefore, we reported the nDCG scores for 1, 5, and 10.

### 5.2.1 Datasets

Blanco et al. [20] reported the Spark performance on a dataset that consists of 4,797 search queries obtained from commercial search engines. We use this data for our evaluation. Every query refers to an entity in DBpedia, and contains a list of entity candidates. The entity candidates are tagged by professional editors on 5 label scales: Excellent, Prefer, Good, Fair, and Bad. The dataset contains different types of entity candidates such as person, location, movie, and TV show. Table 5.1 provides the details about different types of instances in the dataset. It shows that most of the entities are of type “location” or “person”. Section 5.2.3 reports the performance for these specific types in addition to the overall dataset.
5.2 Evaluation

<table>
<thead>
<tr>
<th>Type</th>
<th>Total instance</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Locations</td>
<td>22,062</td>
<td>46.32</td>
</tr>
<tr>
<td>People</td>
<td>21,626</td>
<td>45.41</td>
</tr>
<tr>
<td>Movies</td>
<td>3,031</td>
<td>6.36</td>
</tr>
<tr>
<td>TV shows</td>
<td>280</td>
<td>0.58</td>
</tr>
<tr>
<td>Album</td>
<td>563</td>
<td>1.18</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>47,623</strong></td>
<td><strong>100.00</strong></td>
</tr>
</tbody>
</table>

Table 5.1 Dataset details

5.2.2 Experiments

We performed several experiments with our proposed entity recommendation systems: EnRG and WiFER. EnRG is an unsupervised approach that computes the relevancy scores by using DiSER. Therefore, to compare the performance of EnRG, we generate results by using two other unsupervised methods: GraphSim and ESASim. Since, the evaluation dataset provides a DBpedia entity and a list of potential entity candidates, we need to calculate relevancy scores and rank them accordingly. We calculate relevancy scores by using GraphSim, ESASim, and DiSER. The results obtained by using DiSER are referred to as EnRG here. We use nDCG@10, nDCG@5, and nDCG@1 as the evaluation metrics. As these methods do not require any training, table 5.2 shows the results on full dataset.

In order to inspect whether the additional features obtained from WiFER can complement the Spark performance, we perform the experiments with Spark and SparkWiki. In addition to performing experiments on the dataset with all types of entities, we also evaluate the proposed features for the datasets including only person type or location type entities. WiFER has 7 features that generate 16 feature values. However, Spark contains 112 feature values from 5 types of categories of features, where 56 values are obtained from co-occurrence based features built over 4 different data sources: query term (QT), query session (QS), Flickr tags (FL), and tweets (TW). It means that there are 14 co-occurrence based features generated from each data source. SparkWiki has additional co-occurrence based feature values obtained from WiFER. Therefore, SparkWiki uses 8 co-occurrence based features: Probability \( (P_1, P_2) \), Joint prob-
ability (JPSYM), PMI (SYSYM), Cosine similarity (CSSYM), Conditional probability (CPASYM), Reverse conditional probability (RCPASYM), and Distributional Semantic Model (DSM) vector. The DSM feature was not available in Spark as the data sources used in Spark have small documents (query or tweet). However, Wikipedia characteristics allow us to build the DSM vector over Wikipedia concepts [6, 52]. As a result, SparkWiki consists of 128 features where 16 features are additional to Spark from WiFER. We combine all the features by using GBDT, which is trained by performing 10 fold cross-validation. Therefore, results shown in table 5.3, 5.4, and 5.5 are obtained by 10 fold cross validation.

In order to investigate the importance of the features, we build the ranking model by taking the features from one category at a time. Therefore, we examine the performance of all five models: co-occurrence based, linear combination of co-occurrence based features, graph-based, popularity-based, and type-based. Furthermore, we perform experiments with only co-occurrence based features as they turn out to be the most significant features in the system. We calculate the scores by taking co-occurrence based features and compare the importance of each data source separately.

<table>
<thead>
<tr>
<th>Features</th>
<th>ndcg@10</th>
<th>ndcg@5</th>
<th>ndcg@1</th>
</tr>
</thead>
<tbody>
<tr>
<td>WiFER</td>
<td>0.9173</td>
<td>0.8878</td>
<td>0.8415</td>
</tr>
<tr>
<td>Spark</td>
<td>0.9276</td>
<td>0.9038</td>
<td>0.8698</td>
</tr>
<tr>
<td>SparkWiki</td>
<td>0.9325</td>
<td>0.9089</td>
<td>0.8747</td>
</tr>
</tbody>
</table>

Table 5.3 Retrieval performance of supervised entity recommendation on all entity types
## 5.2 Evaluation

<table>
<thead>
<tr>
<th>Features</th>
<th>ndcg@10</th>
<th>ndcg@5</th>
<th>ndcg@1</th>
</tr>
</thead>
<tbody>
<tr>
<td>WiFER</td>
<td>0.9432</td>
<td>0.9271</td>
<td>0.8857</td>
</tr>
<tr>
<td>Spark</td>
<td>0.9479</td>
<td>0.9337</td>
<td>0.8990</td>
</tr>
<tr>
<td>SparkWiki</td>
<td><strong>0.9505</strong></td>
<td><strong>0.9361</strong></td>
<td><strong>0.9032</strong></td>
</tr>
</tbody>
</table>

Table 5.4 Retrieval performance of supervised entity recommendation on the person entity types

<table>
<thead>
<tr>
<th>Features</th>
<th>ndcg@10</th>
<th>ndcg@5</th>
<th>ndcg@1</th>
</tr>
</thead>
<tbody>
<tr>
<td>WiFER</td>
<td>0.8795</td>
<td>0.8359</td>
<td>0.7773</td>
</tr>
<tr>
<td>Spark</td>
<td>0.8882</td>
<td>0.8507</td>
<td>0.8120</td>
</tr>
<tr>
<td>SparkWiki</td>
<td><strong>0.8987</strong></td>
<td><strong>0.8620</strong></td>
<td><strong>0.8253</strong></td>
</tr>
</tbody>
</table>

Table 5.5 Retrieval performance of supervised entity recommendation on the location entity types

### 5.2.3 Result and Discussion

This section presents the results obtained from the above described experiments. Table 5.2 shows that the relevancy scores computed using DiSER and ESA achieved remarkably higher accuracy than scores based on DBpedia graph. This is because many of the instances in the dataset are very specific type of entities, and DBpedia may not have much information about them yet. However, textual content available in Wikipedia articles are more up to date, which leads to higher accuracy using distributional representation over Wikipedia articles. We also can see that EnRG performed better that ESASim, which shows that entity specific information can be captured better by using DiSER. However, it is hard to make clear conclusion here as ESASim achieved little higher nDCG@1. It is because DiSER builds the vector over annotated entities only, and many new entities might not be annotated in the Wikipedia. However, ESA can capture their distributional information as it only requires the surface forms of entities. Therefore, ESASim generated better top most entity and achieved higher nDCG@1.

Table 5.3 shows the retrieval performance of WiFER, Spark and SparkWiki. It shows that WiFER achieved comparable results for all entity types. Table 5.4 shows that WiFER is a promising approach for person type entities. However, Table 5.5 shows that it could not cope well with location type entities. The possible reason behind this could be that most of the locations are too specific
which do not have enough information on Wikipedia. Although WiFER could not outperform Spark, the combination of both i.e. SparkWiki achieved higher scores for all the test cases. WiFER obtained relatively lower scores for location type entities, however, it is able to compliment the Spark performance in all the cases. Moreover, we can see that unsupervised approaches could not perform comparable to supervised ones. However, the results obtained by ESASim and EnRG are very satisfactory considering that they do not require any training.

In order to inspect the effectiveness of different features, we compute the importance of different features in our learning algorithm. We calculate the reduction in the loss function for every split of feature variable and then compute the total reduction in loss function. It provides how many times the given feature was used in making the final decision by the learning algorithm. For instance, table 5.6 shows that DiSER is used for obtaining the ranking scores of 87% instances.

Table 5.6 lists all the features of WiFER according to their importance, in all type of entities. It shows that the probability (P2) obtained from the textual content of Wikipedia is the most significant feature as it captures specificity of the entities. However, the DSM vectors over Wikipedia anchors (i.e. DiSER)
5.2 Evaluation

<table>
<thead>
<tr>
<th>Rank</th>
<th>Feature</th>
<th>Importance</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Conditional probability over Wikipedia link corpus</td>
<td>100</td>
</tr>
<tr>
<td>2</td>
<td>DSM over Wikipedia link corpus (DiSER)</td>
<td>88.5308</td>
</tr>
<tr>
<td>3</td>
<td>DSM over Wikipedia Text corpus (ESA)</td>
<td>82.6514</td>
</tr>
<tr>
<td>4</td>
<td>Probability of target entity over Wikipedia link corpus</td>
<td>67.7069</td>
</tr>
<tr>
<td>5</td>
<td>Probability of target entity over Wikipedia text corpus</td>
<td>52.8873</td>
</tr>
<tr>
<td>6</td>
<td>Probability of source entity over Wikipedia link corpus</td>
<td>52.6258</td>
</tr>
<tr>
<td>7</td>
<td>Cosine similarity over Wikipedia text corpus</td>
<td>52.2352</td>
</tr>
<tr>
<td>8</td>
<td>Joint probability over Wikipedia link corpus</td>
<td>51.948</td>
</tr>
<tr>
<td>9</td>
<td>Reverse conditional probability over Wikipedia link corpus</td>
<td>49.7912</td>
</tr>
<tr>
<td>10</td>
<td>Reverse conditional probability over Wikipedia text corpus</td>
<td>48.1982</td>
</tr>
<tr>
<td>11</td>
<td>Joint probability over Wikipedia text corpus</td>
<td>42.7255</td>
</tr>
<tr>
<td>12</td>
<td>Probability of source entity over Wikipedia text corpus</td>
<td>41.8859</td>
</tr>
<tr>
<td>13</td>
<td>Conditional probability over Wikipedia text corpus</td>
<td>41.8081</td>
</tr>
<tr>
<td>14</td>
<td>Cosine similarity over Wikipedia link corpus</td>
<td>19.7525</td>
</tr>
<tr>
<td>15</td>
<td>PMI over Wikipedia text corpus</td>
<td>9.12401</td>
</tr>
</tbody>
</table>

Table 5.7 Top 15 features sorted by rank according to their importance in WiFER for person types

<table>
<thead>
<tr>
<th>Rank</th>
<th>Feature</th>
<th>Importance</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Probability of target entity over Wikipedia text corpus</td>
<td>100</td>
</tr>
<tr>
<td>2</td>
<td>Probability of target entity over Wikipedia link corpus</td>
<td>69.9444</td>
</tr>
<tr>
<td>3</td>
<td>DSM over Wikipedia Text corpus (ESA)</td>
<td>63.4468</td>
</tr>
<tr>
<td>4</td>
<td>Probability of source entity over Wikipedia link corpus</td>
<td>53.1977</td>
</tr>
<tr>
<td>5</td>
<td>Cosine similarity over Wikipedia text corpus</td>
<td>49.4132</td>
</tr>
<tr>
<td>6</td>
<td>Reverse conditional probability over Wikipedia text corpus</td>
<td>43.3197</td>
</tr>
<tr>
<td>7</td>
<td>Conditional probability over Wikipedia text corpus</td>
<td>43.0521</td>
</tr>
<tr>
<td>8</td>
<td>Joint probability over Wikipedia text corpus</td>
<td>39.9403</td>
</tr>
<tr>
<td>9</td>
<td>DSM over Wikipedia link corpus (DiSER)</td>
<td>38.4436</td>
</tr>
<tr>
<td>10</td>
<td>Probability of source entity over Wikipedia text corpus</td>
<td>36.5881</td>
</tr>
<tr>
<td>11</td>
<td>Conditional probability over Wikipedia link corpus</td>
<td>34.8135</td>
</tr>
<tr>
<td>12</td>
<td>Reverse conditional probability over Wikipedia link corpus</td>
<td>34.0882</td>
</tr>
<tr>
<td>13</td>
<td>Joint probability over Wikipedia link corpus</td>
<td>33.7888</td>
</tr>
<tr>
<td>14</td>
<td>PMI over Wikipedia text corpus</td>
<td>7.76604</td>
</tr>
<tr>
<td>15</td>
<td>Cosine similarity over Wikipedia link corpus</td>
<td>2.48415</td>
</tr>
</tbody>
</table>

Table 5.8 Top 15 features sorted by rank according to their importance in WiFER for location types

turned out to be more important than vectors over textual content (i.e. ESA) and other features in the model. Further, table 5.7 reports that WiFER features have similar behavior for person type entities. However, feature importance is quite different in case of locations (table 5.8). DSM vectors over anchors (DiSER) falls down from 2<sup>nd</sup> position to 9<sup>th</sup>. It provides further insight that DiSER could not perform well with long tail entities. Moreover, in all the experiments the DSM
Leveraging Wikipedia Knowledge for Entity recommendation

feature shows a good relevance for the model over other features.

In order to investigate if WiFER features play an important role in the com-

<table>
<thead>
<tr>
<th>Rank</th>
<th>Feature</th>
<th>Importance</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Relation type</td>
<td>100</td>
</tr>
<tr>
<td>2</td>
<td>Cosine similarity over Flickr tags</td>
<td>63.3224</td>
</tr>
<tr>
<td>3</td>
<td>Probability of target entity over Wikipedia text corpus</td>
<td>55.9451</td>
</tr>
<tr>
<td>4</td>
<td>CF7 over Flickr tags</td>
<td>54.7444</td>
</tr>
<tr>
<td>5</td>
<td>DSM over Wikipedia Link corpus (DiSER)</td>
<td>54.0078</td>
</tr>
<tr>
<td>6</td>
<td>Conditional user probability over query terms</td>
<td>45.7274</td>
</tr>
<tr>
<td>7</td>
<td>DSM over Wikipedia Text corpus (ESA)</td>
<td>42.2918</td>
</tr>
<tr>
<td>8</td>
<td>Probability of source entity over Wikipedia link corpus</td>
<td>39.9875</td>
</tr>
<tr>
<td>9</td>
<td>Probability of target entity over Flickr tags</td>
<td>38.6405</td>
</tr>
<tr>
<td>10</td>
<td>Probability of target entity over Wikipedia link corpus</td>
<td>36.2818</td>
</tr>
<tr>
<td>11</td>
<td>Conditional user probability over query sessions</td>
<td>34.3559</td>
</tr>
<tr>
<td>12</td>
<td>KL divergence of target entity over Flickr tags</td>
<td>33.945</td>
</tr>
<tr>
<td>13</td>
<td>Conditional probability over Flickr tags</td>
<td>33.062</td>
</tr>
<tr>
<td>14</td>
<td>Entity Facet popularity in knowledge graph</td>
<td>30.1997</td>
</tr>
<tr>
<td>15</td>
<td>CF6 over Flickr tags</td>
<td>29.4447</td>
</tr>
<tr>
<td>16</td>
<td>CF1 over Flickr tags</td>
<td>28.6009</td>
</tr>
<tr>
<td>17</td>
<td>Entropy of target entity in Flickr tags</td>
<td>27.679</td>
</tr>
<tr>
<td>18</td>
<td>Conditional user probability over Flickr tags</td>
<td>27.2086</td>
</tr>
<tr>
<td>19</td>
<td>Conditional probability over query sessions</td>
<td>27.1851</td>
</tr>
<tr>
<td>20</td>
<td>CF8 over Flickr tags</td>
<td>26.9402</td>
</tr>
</tbody>
</table>

Table 5.9 Top 20 features sorted by rank according to their importance in SparkWiki for all types

combined system, we compute the importance of different features in SparkWiki. Table 5.9 shows the importance of the top 20 features for all types of entities. It shows that the relation type (RT$) is the most important feature in SparkWiki which is the same as reported by Blanco et al. [20]. Further, this table reports the effectiveness of the WiFER features as there are 5 WiFER features in the top 10 of most effective ones for the full dataset. It also shows the advantage of using additional DSM features. In particular, table 5.10 shows that the Wikipedia-based DSM feature achieved remarkable importance for person type entities. Moreover, Wikipedia turned out to be a useful data source to obtain background information about location type entities (table 5.11). The Wikipedia document collection created by keeping only anchors (DiSER), shows more effectiveness than taking all the textual content for building the DSM model (ESA). Table 5.11 also reports that probability features (P1 and P2) completely replaced the relation type
5.2 Evaluation

<table>
<thead>
<tr>
<th>Rank</th>
<th>Feature</th>
<th>Importance</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Conditional user probability over query sessions</td>
<td>100</td>
</tr>
<tr>
<td>2</td>
<td>DSM over Wikipedia Link corpus (DiSER)</td>
<td>89.6374</td>
</tr>
<tr>
<td>3</td>
<td>DSM over Wikipedia Text corpus (ESA)</td>
<td>88.5268</td>
</tr>
<tr>
<td>4</td>
<td>Relation type</td>
<td>87.6937</td>
</tr>
<tr>
<td>5</td>
<td>Conditional probability over Wikipedia link corpus</td>
<td>83.381</td>
</tr>
<tr>
<td>6</td>
<td>Conditional probability over query terms</td>
<td>76.5171</td>
</tr>
<tr>
<td>7</td>
<td>Conditional user probability over Flickr tags</td>
<td>64.0326</td>
</tr>
<tr>
<td>8</td>
<td>CF7 over Flickr tags</td>
<td>60.437</td>
</tr>
<tr>
<td>9</td>
<td>Conditional probability over query sessions</td>
<td>53.7163</td>
</tr>
<tr>
<td>10</td>
<td>Entropy of target entity in Flickr tags</td>
<td>53.6132</td>
</tr>
<tr>
<td>11</td>
<td>Entity Facet popularity in knowledge graph</td>
<td>53.4569</td>
</tr>
<tr>
<td>12</td>
<td>Number of search results for target entity</td>
<td>52.9191</td>
</tr>
<tr>
<td>13</td>
<td>Joint probability over Wikipedia link corpus</td>
<td>52.5841</td>
</tr>
<tr>
<td>14</td>
<td>Probability of target entity over Wikipedia link corpus</td>
<td>49.6414</td>
</tr>
<tr>
<td>15</td>
<td>CF8 over Flickr tags</td>
<td>48.7009</td>
</tr>
<tr>
<td>16</td>
<td>Entity popularity in knowledge graph</td>
<td>48.1824</td>
</tr>
<tr>
<td>17</td>
<td>Total counts of target entity in Wikipedia</td>
<td>47.3189</td>
</tr>
<tr>
<td>18</td>
<td>View counts of source entity in query logs</td>
<td>47.1626</td>
</tr>
<tr>
<td>19</td>
<td>Graph similarity using shared nodes</td>
<td>46.3486</td>
</tr>
<tr>
<td>20</td>
<td>CF5 over query session</td>
<td>45.8869</td>
</tr>
</tbody>
</table>

Table 5.10 Top 20 features sorted by rank according to their importance in Spark-Wiki for person types

(RT$) feature as the most significant feature in Spark and SparkWiki for the full dataset. We may conclude that Wikipedia is also a good source to obtain the importance of entities and provides implicit relation type information required in the entity recommendation task.

As we performed experiments by categorizing the features based on their types, we also evaluate models which are built over the subset of features coming from the same category. Table 5.12 shows the scores obtained from five different models based on the feature categories: co-occurrence features, linear combination of co-occurrence features, graph-based features, popularity-based features, and type-based features. It shows that co-occurrence based features are very effective. Although the relation-type feature turned out to be the most important feature (see table 5.9), the type-based features are not very effective without other features. The co-occurrence based features are built by using 5 data sources: query terms, query sessions, Flickr tags, tweets, and Wikipedia. Therefore, we reported the scores generated by co-occurrence based features over different
Leveraging Wikipedia Knowledge for Entity recommendation

<table>
<thead>
<tr>
<th>Rank</th>
<th>Feature</th>
<th>Importance</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Probability of target entity over Wikipedia text corpus</td>
<td>100</td>
</tr>
<tr>
<td>2</td>
<td>Probability of target entity over Wikipedia link corpus (DiSER)</td>
<td>56.6901</td>
</tr>
<tr>
<td>3</td>
<td>Probability of source entity over Wikipedia link corpus</td>
<td>56.3085</td>
</tr>
<tr>
<td>4</td>
<td>Probability of target entity over Flickr tags</td>
<td>55.8144</td>
</tr>
<tr>
<td>5</td>
<td>Cosine similarity over Flickr tags</td>
<td>51.6135</td>
</tr>
<tr>
<td>6</td>
<td>Conditional probability over Flickr tags</td>
<td>51.3306</td>
</tr>
<tr>
<td>7</td>
<td>Conditional user probability over Flickr tags</td>
<td>48.8751</td>
</tr>
<tr>
<td>8</td>
<td>Entity popularity in knowledge graph</td>
<td>44.535</td>
</tr>
<tr>
<td>9</td>
<td>KL divergence of target entity over Flickr tags</td>
<td>44.3557</td>
</tr>
<tr>
<td>10</td>
<td>CF7 over Flickr tags</td>
<td>42.329</td>
</tr>
<tr>
<td>11</td>
<td>CF4 over Flickr tags</td>
<td>41.5688</td>
</tr>
<tr>
<td>12</td>
<td>Entropy of target entity in query terms</td>
<td>39.6188</td>
</tr>
<tr>
<td>13</td>
<td>Entity Facet popularity in knowledge graph</td>
<td>38.6026</td>
</tr>
<tr>
<td>14</td>
<td>Entropy of target entity in Flickr tags</td>
<td>38.0442</td>
</tr>
<tr>
<td>15</td>
<td>CF1 over query session</td>
<td>35.3637</td>
</tr>
<tr>
<td>16</td>
<td>Conditional probability over Wikipedia text corpus</td>
<td>34.6719</td>
</tr>
<tr>
<td>17</td>
<td>length of target entity</td>
<td>34.3997</td>
</tr>
<tr>
<td>18</td>
<td>CF8 over Flickr tags</td>
<td>34.3575</td>
</tr>
<tr>
<td>19</td>
<td>CF1 over Flickr tags</td>
<td>34.0841</td>
</tr>
</tbody>
</table>

Table 5.11 Top 20 features sorted by rank according to their importance in SparkWiki for location types

<table>
<thead>
<tr>
<th>Features types</th>
<th>All</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>ndcg@10</td>
</tr>
<tr>
<td>Co-occurrence</td>
<td>0.9305</td>
</tr>
<tr>
<td>Combined</td>
<td>0.9185</td>
</tr>
<tr>
<td>Graph</td>
<td>0.8953</td>
</tr>
<tr>
<td>Popularity</td>
<td>0.8892</td>
</tr>
<tr>
<td>Type</td>
<td>0.8918</td>
</tr>
<tr>
<td>Person</td>
<td></td>
</tr>
<tr>
<td>Co-occurrence</td>
<td>0.9493</td>
</tr>
<tr>
<td>Combined</td>
<td>0.9420</td>
</tr>
<tr>
<td>Graph</td>
<td>0.9284</td>
</tr>
<tr>
<td>Popularity</td>
<td>0.9269</td>
</tr>
<tr>
<td>Type</td>
<td>0.9229</td>
</tr>
<tr>
<td>Location</td>
<td></td>
</tr>
<tr>
<td>Co-occurrence</td>
<td>0.9493</td>
</tr>
<tr>
<td>Combined</td>
<td>0.9420</td>
</tr>
<tr>
<td>Graph</td>
<td>0.9284</td>
</tr>
<tr>
<td>Popularity</td>
<td>0.9269</td>
</tr>
<tr>
<td>Type</td>
<td>0.9229</td>
</tr>
</tbody>
</table>

Table 5.12 Retrieval performance per feature type

Data sources in table 5.13. It shows that Wikipedia is the most effective resource for all types of entities. However, for location type entities, Flickr tags perform
better than Wikipedia. This shows the usefulness of the Flickr data to capture long tail entities i.e. specific and non-popular place names. Table 5.13 shows that Wikipedia-based features are the most effective ones for building the co-occurrence based model.

### 5.3 Summary

In this chapter, we presented our two approaches for entity recommendation i.e. EnRG and WiFER. EnRG is an unsupervised method that uses DiSER to build an entity relatedness graph. WiFER combines different signals from Wikipedia to obtain features for building a supervised entity recommendation system. Further, we performed an extensive evaluation of our entity recommendation system WiFER and compare it with a commercial entity recommendation system named “Spark”. Spark uses more than 100 features, and produces the final scores by combining these features. These features are built over varying data sources: query term, query session, Flickr tags, and tweets. Therefore, we investigated the performance of these features individually and by combining them based on their data source. Most of the data sources used in Spark such as users’ query
logs, are not publicly available. However, Wikipedia is a continuously growing encyclopedia that is publicly available. Therefore, we showed that the model built only over Wikipedia i.e. WiFER achieved a comparable accuracy to Spark. Moreover, Spark does not utilize Wikipedia to build its features, thus, we also analyzed the effect of using Wikipedia as an additional resource. We showed that WiFER complements the overall performance of Spark.
Significance of quantifying relatedness between two natural language texts has been shown in various tasks which deal with information retrieval (IR), natural language processing (NLP), or other related fields. The semantics of a word can be obtained from existing lexical resources like WordNet and FrameNet. However, such lexical resources require domain expertise for defining the hierarchical structure, which makes their creation very expensive. Therefore, distributional semantic models (DSMs) have achieved much attention as they utilize available document collections like Wikipedia, and do not depend upon human expertise [59]. DSMs represent the semantics of a word by transforming it to a high dimensional distributional vector in a predefined concept space. Many models have been proposed that derive this concept space by using explicit concepts or implicit concepts. Explicit Semantic Analysis (ESA) [52] utilizes the concepts which are explicitly derived under human cognition like Wikipedia concepts (articles). However, Latent Semantic Analysis (LSA) derives a latent concept space by performing dimensionality reduction [85].

Gabrilovich and Markovitch [52] introduced the ESA model in which Wikipedia and Open Directory Project were used to obtain the explicit concepts, however,
Wikipedia has been a popular choice in further ESA implementations [55, 60, 117]. ESA represents the semantics of a word with a high dimensional vector over the Wikipedia concepts. The tf-idf weight of the word with the textual content under a Wikipedia concept can reflect the magnitude of the corresponding vector dimension. To obtain the semantic relatedness between two words, it computes the vector dot product between their vectors. ESA considers the dimensions as orthogonal to each other. For instance, synonyms like “soccer” and “football” are highly related but they may not co-occur in many Wikipedia articles. Table 6.1 shows that the top 5 Wikipedia concepts retrieved for “football” and “soccer” are not shared, whereas the concepts may exhibit relatedness to each other. Consequently, the ESA model assumes that words can be related only if they co-occur in the same articles. Nevertheless two words can also be related even if they do not share the same articles at all, but appear in related ones. LSA resolves the orthogonality issue to some extent by building a latent concept space in an unsupervised way [85]. However, the resulting latent concepts are not as clearly interpretable as the human-labeled concepts in the ESA model. Previous studies [29, 52, 60] show that ESA performs better than LSA for computing text relatedness. Therefore, it is important to consider the relatedness between dimensions in the ESA model, rather than considering them orthogonal, and also without losing the explicit nature of the ESA model at the same time.

In this paper, we present the Non-Orthogonal ESA (NESA) model, an extension of ESA, which also uses relatedness between explicit concepts for computing semantic relatedness between texts. The concepts in the ESA model are clearly
interpretable and they refer to the titles of Wikipedia articles. This characteristic provides an opportunity to investigate different concept relatedness measures, such as relatedness between article content (document relatedness) or relatedness between corresponding Wikipedia titles. In order to investigate the performance of these concept relatedness measures, we presented several experiments on an entity relatedness benchmark KORE [62], in chapter 4.

We then apply different approaches for computing concept relatedness in our model NESA to compute text relatedness. We evaluate NESA on several word relatedness benchmarks to verify whether considering non-orthogonality in the ESA model improves its performance.

### 6.1 Non-Orthogonal Explicit Semantic Analysis

To compute text relatedness, NESA uses relatedness between the dimensions of distributional vectors to overcome the orthogonality in the ESA model. In addition to representing the words as distributional vectors, where each dimension is associated with a Wikipedia concept as in the ESA model, NESA also utilizes a square matrix $C_{n,n}$ ($n$ is the total number of dimensions) containing the correlation weights between the dimensions. Thus, to obtain the relatedness score between the words $w_1$ and $w_2$, NESA formulates the measure as follows:

$$\text{rel}_{\text{NESA}}(w_1, w_2) = w_1^T C_{n,n} w_2$$  \hspace{1cm} (6.1)

where $w_{1,1}$ and $w_{2,1}$ are the corresponding distributional vectors consisting of $n$ dimensions. Every concept dimension can be further semantically interpreted as a distributional vector in some other vector space of $m$ dimensions. This transformation allows the computation of the correlation weights between the concept dimensions. Thus, a transformation matrix $E_{m,n}$ can be built, where each column corresponds to a transformation vector for each concept dimension. Using the matrix $E_{m,n}$, we can compute the matrix $C_{n,n}$ by multiplying $E_{m,n}$ with its transpose as in equation 6.2. In the next section, we discuss the different approaches used for computing $C_{n,n}$ containing the relatedness between the
Using entity relatedness for non-orthogonal explicit semantic analysis

\[ C_{n,n} = E_{n,m}^T \cdot E_{m,n} \]  

\section{6.2 Computing Concept Relatedness}

NESA requires the relatedness scores between Wikipedia concepts (articles), therefore we present different approaches for computing \( C_{n,n} \) matrix using \( E_{m,n} \). Every Wikipedia article consists of different fields to represent the semantics of the concept dimensions, such as Wikipedia title, textual description and entities referred by Wikipedia anchors. We utilize this information to implement four different concept relatedness measures: VSM-Text, VSM-Anchors, ESA-WikiTitle, and DiSER. These approaches represent the semantics of a concept with a distributional vector of \( m \) dimensions. All such vectors combined as column vectors for \( n \) concept dimensions form the matrix \( E_{m,n} \).

\subsection{6.2.1 VSM-Text}

This approach is based on a plain Vector Space Model (VSM) for text. It calculates the relatedness scores between concepts by taking word overlap between corresponding Wikipedia article content. The concept is transformed to a column vector \( m \times 1 \), where \( m \) is the total number of unique words in the Wikipedia corpus. The magnitude of each dimension is calculated on the basis of the number of occurrences of the different words in the associated Wikipedia article content.

\subsection{6.2.2 VSM-Anchors}

Similar to VSM-Text, this approach calculates concept relatedness by taking the overlap between Wikipedia anchors present in corresponding Wikipedia articles’ content. The concept is transformed to a column vector \( m \times 1 \), where \( m \) is the total number of unique anchors in the whole Wikipedia. The magnitude of each dimension is calculated on the basis of the number of occurrences of the different anchors in the associated Wikipedia article content.
6.2.3 ESA-WikiTitle

One intuitive way of obtaining concept relatedness scores is by using ESA itself for calculating the relatedness between concepts. We use the associated Wikipedia article title for this purpose. ESA represents the semantics of a word with a high dimensional vector over the Wikipedia concepts. Therefore, each concept dimension is transformed into a column vector of $m \times 1$, where $m$ is the total number of Wikipedia concepts. The magnitude of each dimension is computed by using term frequency (tf) and inverse document frequency (idf) for the terms appearing in the Wikipedia article title over the Wikipedia corpus [52].

6.2.4 DiSER

Distributional Semantics for Entity Relatedness (DiSER) is our proposed approach that is described in section 4.1.1. DiSER computes the relatedness between two Wikipedia articles by taking the DiSER scores between corresponding titles. Therefore, similar to ESA-WikiTitle, this method uses the Wikipedia titles only.

6.3 Evaluation

For our evaluation, we use the same snapshot of English Wikipedia from 1st October, 2013 which is used in Chapter 4. Section 4.2 explained that we preprocess the Wikipedia corpus by filtering out the noisy articles. We obtain a total of 3,635,833 Wikipedia articles from a total of 13,872,614 articles, for our experiment. All the concept relatedness measures are implemented by using these obtained Wikipedia articles. VSM-Text represents the semantics of a concept with a column vector of $m \times 1$, where $m$ is the total number of unique words that appear in Wikipedia. Wikipedia contains more than 2.5 billion unique words, therefore, to reduce the matrix size, we use only the 5 million most frequent words. ESA-WikiTitle represents the semantics of a concept with a column vector of $m \times 1$, where $m$ is 3,635,833 in our implementation.
NESA is evaluated on different word relatedness and text relatedness benchmarks. We experiment by using different concept relatedness measures as explained in section 6.2 for building the $C_{n,n}$ in NESA model as shown in equations 6.1 and 6.2. We use the filtered Wikipedia articles that are obtained after pre-processing.

### 6.3.1 Word Relatedness Dataset

We use six different word relatedness benchmarks to evaluate NESA.

**WN353**

consists of 353 word pairs annotated by 13-15 human experts on a scale of 0-10. 0 refers to un-related and 10 stands for highly related or identical. This dataset mainly contains generic words like “money”, “drink”, “movie”, etc.. It also contains named entities such as “Jerusalem”, “Palestinian” and “Israel”, which makes this dataset more challenging for approaches that use only lexical resources.

**WN353Rel and WN353Sim**

datasets are subsets of WN353. As WN353 contains similar and related word pairs, Agirre at el. [9] refine the WN353 gold standard by splitting it in two parts: related word pairs and similar word pairs. The notion of similarity and relatedness are defined as follow: two words are similar if they are connected through taxonomic relations like synonym or hyponym in lexical resources, while two words can be considered related if they are connected through other relations such as meronym and holonym. For instance, “football” and “soccer” are two similar words while “computer” and “software” can be considered as related. Finally, WN353Rel and WN353Sim contain 252 and 203 word pairs respectively.

**MC30**

is the dataset build by Miller and Charles [101] that contains selected word pairs of WN353. The relatedness scores of these words are provided by 38 human experts on a scale of 0-4.
RG65
is a collection of 65 non-technical word pairs. These word pairs are annotated by 51 human experts (see for more detail [129]).

MT287
is a relatively new dataset that contains 287 word pairs. This dataset is prepared mainly to study the temporal distribution [122] of a word over several years. The relatedness scores of the word pairs are obtained from 15-20 mechanical turkers.

6.3.2 Text Relatedness Dataset
We use four text relatedness gold standard datasets which are obtained from the SemEval semantic relatedness challenges [11]. These datasets consist of short text pairs, and their relatedness scores are given by human experts.

OnWN
This dataset contains pairs of short phrases and words obtained from OntoNotes and WordNet. SemEval created one version of the OnWN dataset every year from 2012 to 2014. Every dataset consists of 750 pairs and their relatedness scores annotated by 8-10 human experts.

Tweets
This datasets contains 750 pairs of tweets and their relatedness scores. Similar to the OnWN dataset, these tweets are annotated by 8-10 human experts.

6.3.3 Experiment
We compare the NESA model with other state of the art methods for calculating word relatedness: Explicit Semantic Analysis (ESA), Salient Semantic Analysis (SSA), Word2Vec, Latent Semantic Analysis (LSA), and several WordNet-based similarity measures. We implemented the ESA model as described here [52] by considering top 1,000 articles based on their tf-idf scores. We built two LSA
<table>
<thead>
<tr>
<th></th>
<th>WN353</th>
<th>WN353Rel</th>
<th>WN353Sim</th>
<th>MC30</th>
<th>RG65</th>
<th>MT287</th>
</tr>
</thead>
<tbody>
<tr>
<td>H&amp;S</td>
<td>0.347</td>
<td>0.142</td>
<td>0.497</td>
<td>0.811</td>
<td>0.813</td>
<td>0.278</td>
</tr>
<tr>
<td>L&amp;C</td>
<td>0.302</td>
<td>0.172</td>
<td>0.412</td>
<td>0.793</td>
<td>0.823</td>
<td>0.284</td>
</tr>
<tr>
<td>Lesk</td>
<td>0.337</td>
<td>0.125</td>
<td>0.511</td>
<td>0.583</td>
<td>0.546</td>
<td>0.271</td>
</tr>
<tr>
<td>W&amp;P</td>
<td>0.316</td>
<td>0.131</td>
<td>0.461</td>
<td>0.784</td>
<td>0.807</td>
<td>0.331</td>
</tr>
<tr>
<td>Resnik</td>
<td>0.353</td>
<td>0.184</td>
<td>0.535</td>
<td>0.693</td>
<td>0.731</td>
<td>0.234</td>
</tr>
<tr>
<td>J&amp;C</td>
<td>0.317</td>
<td>0.089</td>
<td>0.442</td>
<td>0.820</td>
<td>0.804</td>
<td>0.296</td>
</tr>
<tr>
<td>Lin</td>
<td>0.348</td>
<td>0.154</td>
<td>0.483</td>
<td>0.750</td>
<td>0.788</td>
<td>0.286</td>
</tr>
<tr>
<td>Roget</td>
<td>0.415</td>
<td>-</td>
<td>-</td>
<td>0.856</td>
<td>0.804</td>
<td>-</td>
</tr>
<tr>
<td>LSA</td>
<td>0.579</td>
<td>0.521</td>
<td>0.662</td>
<td>0.667</td>
<td>0.616</td>
<td>0.555</td>
</tr>
<tr>
<td>LSA (Wiki)</td>
<td>0.538</td>
<td>0.506</td>
<td>0.559</td>
<td>0.744</td>
<td>0.697</td>
<td>0.353</td>
</tr>
<tr>
<td>Word2Vec</td>
<td>0.673</td>
<td>0.601</td>
<td><strong>0.741</strong></td>
<td>0.824</td>
<td>0.751</td>
<td>0.560</td>
</tr>
<tr>
<td>SSA</td>
<td>0.629</td>
<td>-</td>
<td>-</td>
<td>0.810</td>
<td>0.830</td>
<td>-</td>
</tr>
<tr>
<td>ESA</td>
<td>0.660</td>
<td>0.643</td>
<td>0.663</td>
<td>0.755</td>
<td>0.826</td>
<td>0.507</td>
</tr>
<tr>
<td>NESA</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>VSM-Text</td>
<td>0.661</td>
<td>0.644</td>
<td>0.679</td>
<td>0.768</td>
<td>0.821</td>
<td>0.506</td>
</tr>
<tr>
<td>VSM-Anchors</td>
<td>0.665</td>
<td>0.643</td>
<td>0.678</td>
<td>0.768</td>
<td>0.821</td>
<td>0.516</td>
</tr>
<tr>
<td>ESA-WikiTitle</td>
<td>0.681</td>
<td>0.654</td>
<td>0.691</td>
<td>0.764</td>
<td>0.827</td>
<td>0.541</td>
</tr>
<tr>
<td>DiSER</td>
<td><strong>0.696</strong></td>
<td><strong>0.663</strong></td>
<td>0.719</td>
<td>0.784</td>
<td>0.839</td>
<td><strong>0.572</strong></td>
</tr>
</tbody>
</table>

Table 6.2 Spearman rank correlation of relatedness measures with word relatedness gold standard datasets

models: using Gutenberg ebooks collection as described in the original article [85], and using Wikipedia. 300 dimensions are used to generate the LSA space. Word2Vec is also built over Wikipedia by taking 200 dimensions with the default parameters described by Mikolov et al. [100]. Henceforth, he results are computed for ESA, SSA, LSA, LSA (Wiki), and Word2Vec on all the popular word relatedness datasets. Hassan and Mihalcea [60] reported SSA performance only on WN353, MC30 and RG65 datasets as shown in table 6.2. The WordNet-based similarity measures are implemented using the WS4J (WordNet Similarity for Java)\(^1\) library built on WordNet 3.0.

### 6.3.4 Results and Discussion

Table 6.2 shows the results of the NESA model with different concept relatedness approaches and other state of the art methods for calculating word relatedness. The knowledge-based methods that use lexical resources like WordNet or the Roget thesaurus [71], achieve higher accuracy if the words in the benchmark datasets are available in the knowledge bases. For instance, WordNet-

\(^1\)https://code.google.com/p/ws4j/
based measures (H&S [61], L&C [86], Lesk [15], W&P [146], Resnik [124] J&C [77], Lin [91]) and the Roget thesaurus-based measure [71] achieve higher accuracy on MC30 and RG65 datasets. However, these approaches may not fit well with the datasets that contain non-dictionary words, therefore, the accuracy of knowledge-based measures decreases significantly on other datasets. Corpus-based measures achieve higher scores than the knowledge-based methods on WN353, WN353Rel, WN353Sim and MT287 datasets. Moreover, corpus-based methods perform comparable to knowledge-based methods on MC30 and RG65. Most of the knowledge-based measures use taxonomic relations for computing word relatedness. Therefore, these measures obtain poor results on WN353Rel in contrast to the WN353Sim dataset. However, corpus-based measures perform well for both types of relations i.e. similarity and relatedness.

All the knowledge-based measures are based on WordNet except Roget, which uses Roget’s thesaurus. Although, WordNet is more popular to perform NLP oriented tasks, Roget’s thesaurus turns out to be very effective in computing word relatedness. Hale [58] showed that traditional edge counting using Roget’s thesaurus achieved remarkably higher scores than using WordNet, which could be because Roget’s thesaurus has less complex structure than WordNet. Moreover, several measures have been proposed using WordNet but it is hard to make clear conclusions about the best performing approach. We can see Word2Vec achieves high scores compare to other corpus-based methods, which shows that neural network based models are very effective for obtaining semantic representations of a word. LSA built over default book corpus performed better than the model generated using Wikipedia, which shows that a further parameter tuning is required to generate LSA space over Wikipedia. As reported by Gabrilovich and Markovitch [52], ESA outperforms LSA for all the datasets. It is interesting that Word2Vec achieves best scores on WN353Sim but it could not perform well on WN353Rel, which is because Word2Vec considers a small context window to encode the co-occurrence of words which might not allow it to capture the weak relatedness between words. Therefore, ESA and NESA perform better than Word2Vec on WN353Rel but not on WN353Sim.
Table 6.2 shows that the NESA model combined with any concept relatedness measure outperforms ESA for all the word relatedness benchmark datasets. It shows that considering non-orthogonality between explicit concepts in the ESA model improves accuracy. In most cases, NESA-VSM-Anchors performs better than NESA-VSM-Text, implying that considering only the anchors from the article content works better than taking the overlap of the whole content. NESA-ESA-WikiTitle and NESA-DiSER achieve higher scores than both NESA-VSM-Text and NESA-VSM-Anchors. It shows that the distributional representation of the article title captures the semantic information better than considering only the corresponding article content. NESA-DiSER achieves the highest correlation scores in all the word relatedness benchmark datasets. NESA outperforms the LSA and ESA models. Another interesting thing to note here is that the correlation scores obtained by the NESA model with the four concept relatedness measures follow the same order in table 6.2 as the correlation scores obtained in evaluating concept relatedness shown in table 4.3 in Chapter 4. It shows the consistency of proposed concept relatedness measures in two different experiment settings.

In order to investigate on which cases NESA is more effective, we further

<table>
<thead>
<tr>
<th>word1</th>
<th>word2</th>
<th>Human Rank</th>
<th>ESA Rank</th>
<th>NESA Rank</th>
</tr>
</thead>
<tbody>
<tr>
<td>football</td>
<td>soccer</td>
<td>344</td>
<td>181</td>
<td>290</td>
</tr>
<tr>
<td>car</td>
<td>automobile</td>
<td>340</td>
<td>324</td>
<td>343</td>
</tr>
<tr>
<td>money</td>
<td>currency</td>
<td>345</td>
<td>326</td>
<td>340</td>
</tr>
</tbody>
</table>

Table 6.3 Analysis over synonyms

<table>
<thead>
<tr>
<th>word1</th>
<th>word2</th>
<th>Human Rank</th>
<th>ESA Rank</th>
<th>NESA Rank</th>
</tr>
</thead>
<tbody>
<tr>
<td>tiger</td>
<td>cat</td>
<td>244</td>
<td>61</td>
<td>140</td>
</tr>
<tr>
<td>lobster</td>
<td>food</td>
<td>285</td>
<td>190</td>
<td>247</td>
</tr>
</tbody>
</table>

Table 6.4 Analysis over hyponyms

analyze different examples from WN353 dataset. We use Spearman rank correlation for our evaluation, which measures the performance by computing the
difference in rankings generated by the measure with the gold standard rankings given by human. We discussed above that the vectors of “football” and “soccer” do not have many shared articles, therefore, ESA generates a low score about their relatedness. However, considering relatedness between the articles or explicit topics in NESA can reduce this issue. Table 6.3 shows the word pairs that can be considered as synonyms. It shows that ESA places “football” and “soccer” at 181\textsuperscript{th} rank. However, NESA boosts the ESA rank from 181 to 290, which is very close to the gold standard ranking. Similarly, NESA pushes the ranking of “car” and “automobile”, and “money” and “currency” close to the ground truth. It can be concluded that NESA improves the ESA accuracy for synonyms. Table 6.4 shows that NESA also boosts the ESA ranking for hyponyms, which could be the same reason that hyponyms generally appear in similar context rather in the same articles. Moreover, NESA improves the ESA ranking for the word pairs which can also be seen as word phrases, e.g. “credit” and “card” (table 6.5). It is very likely that word phrases will appear together in the same articles, but the results show that considering relatedness between articles can complement the ESA performance. To further analyze the related words, we compare the rankings of the word pairs which have implicit relations, e.g. “cup” and “liquid”. Table 6.6 shows that considering relatedness between articles could be harmful for such word pairs. ESA generates better ranking for these word pairs. However, it is difficult to draw a clear conclusion for ranking word pairs with implicit relations as NESA gives better ranking for “profit” and “loss”.

Table 6.7 shows the results for LSA, ESA and NESA on different text relatedness datasets. Similar to word relatedness task, ESA performed better than LSA for all the text relatedness datasets. Although, NESA outperforms ESA, the improvement is insignificant. It shows that the word relatedness task takes more
Using entity relatedness for non-orthogonal explicit semantic analysis

<table>
<thead>
<tr>
<th>word1</th>
<th>word2</th>
<th>Human Rank</th>
<th>ESA Rank</th>
<th>NESA Rank</th>
</tr>
</thead>
<tbody>
<tr>
<td>closet</td>
<td>clothes</td>
<td>296</td>
<td>275</td>
<td>217</td>
</tr>
<tr>
<td>cup</td>
<td>liquid</td>
<td>146</td>
<td>136</td>
<td>46</td>
</tr>
<tr>
<td>profit</td>
<td>loss</td>
<td>274</td>
<td>224</td>
<td>288</td>
</tr>
</tbody>
</table>

Table 6.6 Analysis over implicit relations

benefit of considering relatedness between articles. This is because a word could be more ambiguous than a phrase or sentence. Thus, ESA may fail to generate an unambiguous vector for a word, which leads to obtaining irrelevant articles in the vector. However, a phrase carries more context which can help in generating a better vector representation. Therefore, NESA does not have much effect on ESA for text relatedness.

<table>
<thead>
<tr>
<th></th>
<th>OnWN-12</th>
<th>OnWN-13</th>
<th>OnWN-14</th>
<th>Tweets-14</th>
</tr>
</thead>
<tbody>
<tr>
<td>LSA</td>
<td>0.566</td>
<td>0.648</td>
<td>0.669</td>
<td>0.568</td>
</tr>
<tr>
<td>ESA</td>
<td>0.602</td>
<td>0.688</td>
<td>0.768</td>
<td>0.609</td>
</tr>
<tr>
<td>VSM-Text</td>
<td>0.598</td>
<td>0.688</td>
<td>0.769</td>
<td>0.607</td>
</tr>
<tr>
<td>VSM-Anchors</td>
<td>0.603</td>
<td>0.686</td>
<td>0.770</td>
<td>0.609</td>
</tr>
<tr>
<td>ESA-WikiTitle</td>
<td>0.611</td>
<td>0.701</td>
<td>0.775</td>
<td>0.614</td>
</tr>
<tr>
<td>DISER</td>
<td><strong>0.617</strong></td>
<td><strong>0.707</strong></td>
<td><strong>0.787</strong></td>
<td><strong>0.616</strong></td>
</tr>
</tbody>
</table>

Table 6.7 Spearman rank correlation of relatedness measures with text relatedness gold standard datasets

### 6.4 Summary

This chapter presented Non-Orthogonal Explicit Semantic Analysis (NESA) which introduces the relatedness between explicit concepts in the ESA model for computing semantic relatedness, without compromising the explicit nature of the ESA concept space. We showed that the word relatedness results vary with the different concept relatedness measures. NESA outperformed all state of the art methods, in particular, NESA-DISER achieved the highest correlation with the gold standard datasets. We can conclude that considering correlation between concepts in ESA improves the overall accuracy. Further, we found that NESA is
more effective for word relatedness than text relatedness. Our empirical eval-
uation showed that NESA improves the ESA performance in different type of
word pairs like synonyms, hyponyms and word phrases. Moreover, NESA im-
proves ESA significantly for word similarity dataset but it could outperformed
Word2Vec, which shows that Word2Vec captures the better contextual semantics
around the given word. Since, NESA and Word2Vec have different parameters,
a further investigation is required to obtain more concrete conclusion.
Chapter 7

Application

This chapter highlights several use cases of the work presented in this thesis, starting with an entity recommendation system “EnRG” that provides different functionalities to users for exploring related information about their favourite topics or entities. Section 7.1 describes EnRG and its functionalities provided through a web user interface. Section 7.2 presents Medical Concept Resolution (MCR) that uses our relatedness measure in medical concept disambiguation as part of the IBM Watson Question Answering System [45]. We also present Cross-Lingual Natural Language Querying (CroNL) that retrieves answers from the English DBpedia for a natural language query in German. CroNL uses a cross-lingual extension of the relatedness measure explained in chapter 6.

7.1 EnRG: Entity relatedness graph

As we discussed in chapter 5 that most of the entity recommendation approaches make use of several features and combine them using machine learning methods to recommend the relevant list of entities. On the contrary, many small scale enterprises (SMEs) dealing with online retailing often lack high volumes of data about their products (entities), which is required by the machine learning algorithms for providing accurate recommendations. Therefore, we demonstrate EnRG, which does not require any user purchase data or the item content from the online retailers dealing in the entertainment domain. Entertainment items
like books, movies, TV shows, music, computer games or related products can exhibit relations to many entities like authors, actors, musicians etc. EnRG is a precomputed graph that contains all the Wikipedia entities and their relevancy scores (described in section 5.1.1). We built a web user interface (UI) based on EnRG that we refer to as EnRG-UI in this chapter. EnRG-UI provides users an easy exploration over related entities beyond the explicitly defined relations in knowledge graphs like DBpedia and Freebase. It provides a ranking of the connections between the entities in order to select the top related entities for a better retrieval and recommendation. The types of entities suggested by the current search engines are generally abstract and limited to people, movies, and others [20, 151]. On the contrary, EnRG-UI allows an extensive retrieval on the basis of dynamic facets and filters using DBpedia. EnRG considers every Wikipedia article topic as an entity except the Wikipedia pages referring to lists, disambiguation pages, and redirects. Similar to DBpedia, every node in EnRG represents a Wikipedia based article. The edges between the nodes reflect the relevancy scores between the corresponding entities in EnRG. EnRG-UI can also be seen as a search assistance system that retrieves several ranked lists of related entities classified under different types, and provides a further exploration on the results to the search query.

### 7.1.1 Web User Interface

EnRG provides the user with a dynamic set of filters and facets for exploring related entities with the help of DBpedia and YAGO [136]. Every Wikipedia article has a corresponding DBpedia page that provides further exploration for different relations of the entity. DBpedia defines rdf:type of every Wikipedia entity, which allows us to get the ranked list for each type. These types include classes mainly from the DBpedia ontology and YAGO. The DBpedia ontology covers abstract types like Person, Company, Location, Movie and others, while YAGO also provides very specific types like American film actors, People from Manhattan, etc.

---

2. [http://www.w3.org/1999/02/22-rdf-syntax-ns#type](http://www.w3.org/1999/02/22-rdf-syntax-ns#type)
4. YagoClasses
Fig. 7.1 Related entities to Brad Pitt
Fig. 7.2 Ranked list of american vegans related to Brad Pitt

This information enables us to group the related entities under different types.

Figure 7.1 shows a snapshot of the EnRG-UI, which illustrates ranked lists of related entities to “Brad Pitt”, categorized under different abstract types. Just below the main heading “Brad Pitt”, it shows the different specific types for “Brad Pitt” such as “American film actors”, “21st century actors”, “American vegans” and others. The figure also shows different functionalities provided by EnRG-UI. On clicking any entity type, it would list all the entities related to “Brad Pitt”, listed under that type. For instance, using such filters for “Brad Pitt”, one can easily navigate to find the “American vegans” related to Brad Pitt as shown in Figure 7.2. To explore the related films to Brad Pitt, one can simply click the “Explore” link at the bottom of the list of related films, which is shown in Figure 7.3. These exploratory options with specific type can be seen as a search assistance facility for making a complex query over a knowledge graph. For instance, a user can
find “Irish Universities founded in 1845” that are closely related to “National University of Ireland Galway (NUIG)” (shown in figure 7.5). The application also presents two types of provenance information, one based on Wikipedia articles containing both the related entities, while the other performs a Google web search giving the provenance information from the web. It also shows the relatedness strength quantifying the entity relatedness. The application also gives query suggestions pointing to the entities that the query string might refer to. For instance, if someone searches just using the string “Apple”, then it shows the related entities for Apple fruit, but also gives other query suggestions like Apple Inc. and Apple Records. Selecting the query suggestion “Apple Inc.” would lead to the related entities of Apple Inc. through which we can easily navigate to “Computer Hardware companies” that are closely related to “Apple Inc.” (shown in figure 7.4).
Fig. 7.4 Hardware companies related to Apple Inc.

Fig. 7.5 Educational institutions established in 1845, related to NUIG
7.2 Medical concept recognition

Linking phrases in text to concepts in a Knowledge Base (KB) like the Unified Medical Language System (UMLS) \(^5\) becomes difficult with the increasing size of this resource, as it comprises of multiple ontologies which are manually curated. Although standard open domain entity linking (EL) (discussed in section 4.3) deals with finding concepts or entities in a KB that match to a given phrase (mention) in text, it generally assumes that there could only be one correct match at a time for a given mention. However, in domain specific EL, there could be more than one correct match for a given mention depending upon the context. Moreover, open domain entity linking finds the entries in a KB that match the mention text exactly, so there is no need to “discover” the candidate entries with partial matches.

Commercial systems like IBM Watson exploits the different knowledge for answering the medical questions. Therefore, such systems require a high quality medical concept linking tool which can provide background knowledge about different span of text present in given question. Therefore, we present Medical Concept Resolution (MCR), a system for finding a concept (more precisely, a Concept Unique Identifier (CUI)) in UMLS, for medical terms mentioned in text \([3]\). MCR has two major steps: candidate overgeneration and candidate ranking. We consider partially matched candidates to overgenerate the candidates for a given mention. In order to rank the candidates, we make use of the similarity measure described in this thesis. In particular, we calculate similarity scores between the context around a mention and the context of a candidate in the knowledge base. MCR has properties of its own that makes it more challenging. We describe the following properties with examples for mapping text spans to CUIs in UMLS, but we note that these properties would likely apply to any large concept repository, especially those comprising concepts from multiple different sources.

- **Discovery** In a large concept repository such as UMLS, the hierarchy of concepts and the labels supplied for them can be arbitrary. Frequently,

\(^5\)http://www.nlm.nih.gov/research/umls/
there is no concept whose label matches a text span exactly. Given the variability of medical language, the span detection problem for medical terms is significantly more difficult than the typical word sense disambiguation or entity linking tasks. For these reasons, it can be quite difficult to determine whether a concept exists in the repository or not. For example, there is no UMLS concept with a label that matches the text span “distended jugular vein”. Relaxing the term order (e.g., “jugular vein distended”) and substituting synonyms (e.g., “engorged jugular vein”), we still find no UMLS concepts with matching labels. There is, however, a concept in UMLS for jugular vein distention.

- **Multiplicity** In concept repositories that combine multiple sources, there are often multiple entries for the same domain concept. So even once a concept is discovered, there may still be some other appropriate concepts. In UMLS there is only one CUI with the label “jugular vein distention”, but there are two with the label “jugular vein distension”. Finding all appropriate concepts in the repository is important because some information (e.g., relations) may be asserted for one but not others. In our example, neither of the two CUIs for “jugular vein distension” have relations indicating the condition for these symptoms. But there is a third CUI in UMLS: jugular venous distension (none of whose labels include phrases containing “vein”) with the information that it is clinically associated with the CUIs for congestive heart failure and organic heart disease.

- **Granularity** Even when there are close superficial matches between concept labels and text spans, more specific concepts often exist that capture more of the semantics of the span, given a larger context. If our “distended jugular vein” example appeared in text in the context of it occurring on “inspiration”, then a more appropriate UMLS concept might be the CUI for paradoxical inspiratory filling of neck veins (aka, Kussmaul’s sign or jugular venous distention with inspiration). These properties of the problem make the traditional metrics for word sense disambiguation and entity linking inadequate for evaluating MCR algorithms. In our experiments, we have found many cases in which different CUIs that were judged by experts to
identify the same term correctly actually ranged in acceptability and have clear rankings among them.

### 7.2.1 Approach

Our approach for mapping text spans (“mentions”) to UMLS CUIs consists of two main steps: candidate overgeneration and candidate ranking. For obtaining the mentions from text, we use a CRF-based method of extracting medical terms [141].

#### 7.2.1.1 Candidate Overgeneration

The first step “overgenerates” candidate CUIs, on the intuition that there may be a mismatch between the mentions in text and the variant labels of target CUIs: not all of the tokens within a mention are necessarily relevant (present in CUI variant labels) and not all of the tokens in CUI variant labels are necessarily in the mention (though they may be in context well outside of the mention span). Overgeneration finds all the CUIs having any variant containing any of the tokens in the mention text. The resulting candidates include many irrelevant CUIs, but also relevant CUIs that are more general or specific than the mention. For example, candidates for the string “pupil miosis” include the CUIs for pupil, miosis, school pupil, congenital miosis, and pupillary miosis of the right eye. Candidate overgeneration may produce an enormous number of candidates. For efficiency, only candidates most similar to the mention are considered in the subsequent ranking step. The most similar candidates are determined by “IDF-ranked-weighted” similarity of their labels to the mention text. The n tokens in the original mention are ranked according to their inverse document frequency (IDF) in a medical corpus. The ranks are converted to weights \( w_i = r_i/n \), where \( r_i \) is the IDF rank of the \( i_{th} \) token. The least frequent (highest IDF) token has rank \( n \) while the most frequent token is of rank 1. For example, for the phrase “pupillary miosis of the right eye”, the weighted word vector would be: [“pupillary”:0.83, “miosis”:1.0, “of”:0.33, “the”:0.17, “right”:0.5, “eye”:0.67]. The weights are used in calculating weighted cosine similarity between the mention tokens and candidate CUI variants. All candidates (up to 100) having variants with similarity to
the mention text above a certain threshold are kept for the ranking step. Converting IDF values to rank weights serves to normalize and smooth over the IDF values.

### 7.2.1.2 Candidate Ranking

In the second step, the candidate CUIs are ranked by measuring the similarity between mention context and candidate context. The mention context is a relatively large window of text surrounding the mention. Inspection of patient description scenarios showed that information from distant parts of the text often informs about the interpretation of a clinical factor text span. We experimented with the full sentence containing the mention span ($S_m$) as the mention context for the experiments described below. On the candidate CUI side, we implemented the following three different context generators:

- **Gloss-Based Medical Concept Resolution (GBMCR)** Two contributing sources in UMLS are MeSH and NCI, which together contribute definitions for only roughly 3% of concepts. Nevertheless, for 86% of the clinical factor mentions in our experiments, at least one of the filtered candidate concepts had at least one MeSH or NCI definition. In GBMCR, candidates are ranked according to the cosine similarity between the words in the factor mention sentence ($S_m$) and the words in the MeSH definition of the candidate, if one exists, otherwise the words from the NCI definition. For the candidate CUI C0018965 (Hematuria), the GBMCR context is [“presence”, “of”, “blood”, “in”, “the”, “urine”].

- **Neighbor-Based Medical Concept Resolution (NBMCR)** In addition to taxonomic relations, UMLS contains semantic relations between concepts. In NBMCR, we consider a subset of “clinically relevant” relations based on the semantic type of the candidate CUI. For example, for candidates of type Disease or Syndrome, we consider semantic relations for symptoms, treatments, risk factors, etc. Candidates are ranked according to the similarity between the words in Sm and the words in variants of the neighbor CUIs semantically related to the candidate. For the candidate Hematuria, neighbors include CUIs for urinary tract bladder, urine screening for blood, flank
pain, and many more. The NMBCR context for Hematuria is [“bladder”, “urinary”, “tract”, “urine”, “screening”, “for”, “blood”, “flank”, “pain”].

- **Variants-Based Medical Concept Resolution (VBMCR)** The bag of words of all of the variants in UMLS of the candidate CUI make up the VBMCR candidate context. For Hematuria, the VBMCR context is [“blood”, “in”, “urine”, “hematuria”, “haematuria”, “urine”, “blood”]. By allowing candidate CUI variants to contain words not in the mention span and then preferring CUIs whose variants more closely match the words in the larger context surrounding the mention, MCR is often able to recover from poor span boundary detection.

### 7.2.2 Evaluation

In order to evaluate our MCR methods, we compare the performance to MetaMap (mmap)\(^6\) on a dataset that contains 1,570 clinical factor spans extracted from 100 short descriptions (averaging roughly 8 sentences, 100 words) of “patient scenarios”. The MCR algorithms can produce a ranked list of as many CUIs as there are filtered candidates (arbitrarily capped at 100). We included the top three ranked concepts for each factor in the evaluation. Five human judges were randomly assigned roughly 560 factor-CUI mappings each. The assignments did not overlap, but judges also rated 41 mappings in common. The average pairwise kappa for judge agreement was 0.6. For each CUI for a clinical factor, judges gave a score of: 0 (inappropriate for the factor), 1 (appropriate for the factor) and 2 (appropriate for the factor and “better” than concepts scoring 1 for the same factor). The purpose of distinguishing two “grades” of appropriate is to verify one of the original motivations for MCR: that even when a CUI appropriate to the exact span exists, there are often more specific CUIs that are more appropriate considering more context. For example, consider the sentence “She has pain in the epigastric region and sometimes on the right side of her abdomen”. For the factor “pain”, the CUI for pain is appropriate, but UMLS also has CUIs for abdominal pain and right sided abdominal pain.

\(^6\)Metamap
<table>
<thead>
<tr>
<th>Method</th>
<th>Best-is-correct</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Precision</td>
<td>Recall</td>
<td>F1</td>
<td>Precision</td>
<td>Recall</td>
<td>F1</td>
<td>Precision</td>
</tr>
<tr>
<td>mmap</td>
<td>0.614</td>
<td>0.608</td>
<td>0.611</td>
<td>0.794</td>
<td>0.787</td>
<td>0.790</td>
<td></td>
</tr>
<tr>
<td>GBMCR</td>
<td>0.450</td>
<td>0.427</td>
<td>0.438</td>
<td>0.628</td>
<td>0.596</td>
<td>0.611</td>
<td></td>
</tr>
<tr>
<td>NBMCR</td>
<td>0.567</td>
<td>0.538</td>
<td>0.552</td>
<td>0.624</td>
<td>0.592</td>
<td>0.608</td>
<td></td>
</tr>
<tr>
<td>VBMCR</td>
<td><strong>0.721</strong></td>
<td><strong>0.685</strong></td>
<td><strong>0.703</strong></td>
<td><strong>0.762</strong></td>
<td><strong>0.723</strong></td>
<td><strong>0.742</strong></td>
<td></td>
</tr>
</tbody>
</table>

Table 7.1 Results of Medical Concept Resolution on human annotated clinical factors

### 7.2.3 Results and Discussion

We calculate the precision, recall and F1 by considering two settings: best (annotated with score 2) is correct and appropriate (annotated with score 2 or 1) is correct. Therefore, we can compare the performance of different methods in a strict environment and in a relaxed one. Table 7.1 shows that our method VBMCR outperforms all other approaches in strict environment (Best-is-correct). Particularly, VBMCR achieved more than 13% improvement over a state of the art method of medical entity linking i.e. MetaMap. Although VBMCR achieved high accuracy, the other two methods i.e. NBMCR and GBMCR did not perform well. The possible reason behind this could be the lack of glosses associated with concepts in UMLS. MetaMap achieved the highest scores in the relaxed setting (Appropriate-is-correct). This shows that our approaches are more strict in finding out the best fit according to context and MetaMap is not able to differentiate between acceptable and best matched concepts.

### 7.3 Cross-lingual natural language querying

Non-Orthogonal Explicit Semantic Analysis (NESA) uses the English Wikipedia to build a distributional vector of a word. The articles in Wikipedia are linked together across different languages which facilitates a mapping of a vector in one language to the other. This cross-lingual mapping is used to build a cross-lingual extension of ESA called Cross-Lingual ESA (CLESA) [133]. Similar to CLESA, we extend NESA across multiple languages. We refer this extension as Cross-Lingual NESA (CLNESA). We use CLNESA to develop a pipeline to perform
natural language querying over structured knowledge like DBpedia across different languages.

Retrieval of structured data, in general, requires structured queries like SPARQL\(^7\); however, effective construction of such queries is a laborious process. In order to provide a flexible querying environment, an automatic construction of a structured query from a natural language query (NL-Query) is required. While there are several efforts ([5, 50, 139, 147]) to convert NL-Queries into structured queries in the monolingual scenario, the multilingual scenario offers further challenges [8]. For example, the problem of mapping the query vocabulary to the ontology vocabulary is exacerbated by poor quality of ontology term translation [12] and by the lack of multilingual structured resources. Therefore, to avoid relying on automatic translation, we present an approach for cross-lingual natural language querying (CroNL) which includes entity identification, linguistic analysis, and semantic relatedness measures. We used CLNESA to calculate relatedness scores between the query vocabulary and the properties in DBPedia. In particular, CroNL focuses on the interpretation of NL-Queries by traversal over the structured knowledge graph, and the construction of a corresponding SPARQL query. The translation based approaches for cross-lingual NL-Queries suffer from the poor quality of automatic translation. Therefore, we introduce an approach for performing cross-lingual NL-Queries over a structured knowledge base, without automatic translation. In order to evaluate our approach, we created a benchmark dataset of 50 NL-Queries in German. We discuss the results of a comparison of CLNESA with an automatic translation-based method.

### 7.3.1 Related work

Most of the proposed approaches that address the task of Cross-Lingual Information Retrieval (CLIR) reduce the problem into a monolingual scenario by translating the search query or documents in the corresponding language. Many of

\(^7\)http://www.w3.org/TR/rdf-sparql-query/
them perform query translation ([93], [113], [103], [78])) into the language of the documents. However, all these approaches suffer from the poor performance of machine translation on short texts like queries. Jones et al. [78] performs query translation by restricting the translation to the cultural heritage domain, while Nquyen et al. [103] makes use of the Wikipedia cross-lingual links structure.

Without relying on machine translation, some approaches ([92], [153], [133]) make use of distributional semantics. They calculate a cross-lingual semantic relatedness score between the query and the documents. However, none of these approaches interpret the query to retrieve the answers from large available structured knowledge bases. With the assumption that documents of different languages are already marked-up with the knowledge base (for instance, Wikipedia articles are annotated with DBpedia), the problem of CLIR can be converted into querying over structured data. There is still a language barrier, as queries can be in different languages, while most of the structured data is only available in English. Qall-Me [105] performs NL-Querying over structured information by using textual entailment to convert a natural language question into SPARQL. This system relies on availability of multilingual structured data. It can only retrieve the information that is available in the query language. Therefore, this system is not able to perform CLIR. Many small scale enterprises (SMEs) want to support the search functionality in different languages over their products like books and movies. However, the existing methods mainly focus on monolingual scenarios. Therefore, we focus on developing a system which can perform query in different languages over several entity types.

7.3.2 Approach

The key to our approach is the interpretation of NL-Queries in different languages by using a combination of entity identification, linguistic analysis and cross-lingual relatedness. Figure 7.6 shows these three components of CroNL
along with an example of a NL-Query in German\(^8\). The interpretation process
starts with the identification of possible entities appearing in a given NL-Query
followed by linguistic analysis of the NL-Query. The system executes the whole
pipeline with all the identified entities and takes the union of all of the retrieved
results. Using the dependencies provided by the linguistic analysis, our system
determines the next term that will be compared with all the relations associ-
ated with the identified entity to find the best matched relation. For instance, in
example shown in Figure 7.6, the system identified “Bill Clinton” as entity and
“Tochter” as next term. Following the process, it calculates the relatedness score
with every relation associated with “Bill Clinton” in the DBpedia, and finds the
most related relation to obtain the next entity from DBpedia.

7.3.2.1 Entity identification

The first step of the interpretation process is the identification of potential enti-
ties, i.e. the DBpedia entities appearing in the NL-Query. We built an index of
all entity labels in the DBpedia, and retrieve a label with an exact match against
the term appearing in the NL-Query. For example, DBpedia: Bill_Clinton shown
in Figure 7.6. “Bill Clinton” is the name of a person and it appears as a label of
DBpedia: Bill_Clinton URI. In addition, CroNL also disambiguates the selected
entity candidates based on their associated relations in the knowledge base. For
instance, in a given NL-Query “Wie viele Angestellte hat Google?”\(^9\) two dif-
ferent DBpedia entities can be found with the label “Google”, i.e. “DBpedia:
Google_Search” and “DBpedia: Google”. We calculate relatedness scores with all
associated relations of both, and find that the term “Angestellte” in the NL-Query
obtained a maximum relatedness score with the relation “number Employees”,
which is associated with “DBpedia: Google”.

\(^8\) Translated from the QALD-2 challenge dataset, which has 100 NL-Queries in English, over
DBpedia

\(^9\) Translation of “How many employees does Google have?” from the English test dataset.
7.3.2.2 Linguistic analysis

Linguistic analysis of the NL-Queries is needed to get the dependencies between the identified entities and terms. We use the Stanford parser\textsuperscript{10} for German to generate the dependencies. Following these dependencies, we convert the given NL-Query into a Direct Acyclic Graph (DAG). Vertices of the generated DAG represent the entities and edges reflect the terms directly dependent on the obtained entities (vertices). Figure 7.6 shows the DAG obtained from our example query “Mit wem is die Tochter von Bill Clinton verheiratet?” To generate the DAG, first we obtain the central entity from the previous step. With the relations of this central entity, semantic matching will be performed. Therefore, we retrieve the directly dependent terms of the central entity provided by the generated Stanford typed dependencies, and add them into the DAG. Similarly, we perform this action for all the other terms in the list. For instance, in the example NL-Query shown in Figure 7.6, the system identifies “Bill Clinton” as a central entity,\textsuperscript{11} and then “Tochter” as direct dependent of “Bill Clinton” followed by “verheiratet” as direct dependent of “Tochter”.

\textsuperscript{10}http://nlp.stanford.edu/software/lex-parser.shtml
\textsuperscript{11}The term to start the search around in the whole DBpedia graph
7.3 Cross-lingual natural language querying

7.3.2.3 Knowledge graph traversing using semantic relatedness

DBpedia contains the semantically structured data of entities and their relations. Our next step is to find such relations of a selected central entity in DBpedia that is best match with the term directly dependent on this central entity in the generated DAG. First, we search for the entity “Bill Clinton” in DBpedia, and retrieve all of the relations (DBpedia properties) associated with it. Then, we find the best semantically matching DBpedia property of the direct dependent term “Tochter” by calculating a cross-lingual relatedness score between all the DBpedia properties of Bill Clinton and “Tochter”. After obtaining the relevant property, i.e. “child”, we find the entity DBpedia: Chelsea_Clinton, connected with entity “DBpedia: Bill_Clinton” by property “child”. We perform the same steps with the retrieved entity for the directly dependent term “verheiratet” of “Tochter”, and so on until the end of the DAG. Finally, we retrieve the most relevant entity and all the associated documents in different languages containing a description about this entity.

7.3.3 Evaluation

7.3.3.1 Datasets

In order to evaluate CroNL, we created a test dataset of 50 NL-Queries in German. This benchmark is created by manually translating the English NL-Queries provided by the “Question Answering over Linked Data (QALD-2)” dataset, consisting of 100 NL-Queries in English over DBpedia. All of the NL-Queries are annotated with keywords, corresponding SPARQL queries and answers retrieved from DBpedia. Also, every NL-Query specifies some additional attributes, for example, if a mathematical operation such as aggregation, count or sort is needed in order to retrieve the appropriate answers.

We translated QALD-2 dataset and divided it into two parts, one for development and other for testing. Therefore, each dataset contains 50 NL-Queries in German. We performed a manual analysis to keep the same complexity level in both the datasets. We divided all the NL-Queries into three different categories:
simple, template-based and SPARQL aggregation. Simple queries contain the DBpedia entities and their relations (DBpedia properties), and do not need a predefined template or rule to construct the corresponding SPARQL query. However, these queries include semantic and linguistic variations, that means they express the DBpedia properties by using different terms rather than having the exact label of a property. For instance, in a given query “How tall is Michael Jordan?”, “tall” does not appear in the vocabulary of DBpedia properties, however, the answer of the query can be retrieved by using the DBpedia property “height” appearing with “DBpedia: Michael_Jordan”. The queries requiring predefined templates or rules are categorized as template-based queries [139], for example, the query “Give me all professional skateboarders from Sweden” requires a predefined template for retrieving professionals with occupation Skateboarding and born in Sweden. We filter the template-based queries if a query does not have any keyword corresponding to the properties used in its SPARQL. The SPARQL aggregation type of queries needs a mathematical operation such as aggregation, count or sort, therefore, this also requires a predefined rule.

Following the categorization, we divided the dataset into two parts by keeping an equal number of queries in each category. We then performed our experiments on the prepared test dataset of 50 NL-Queries in German. Table 7.2 shows the statistics about both the datasets.

### 7.3.3.2 Experiment

We evaluated the outcome of CroNL at all three stages of the processing pipeline: 1) entity identification, 2) linguistic analysis, and 3) semantic relatedness measures. This way, we can investigate the errors introduced by individual components. As shown in Figure 7.6, the third component “cross lingual relatedness
7.3 Cross-lingual natural language querying

<table>
<thead>
<tr>
<th>Error Type</th>
<th>No of NL-Queries</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>out of 50</td>
</tr>
<tr>
<td>Entity Identification</td>
<td>7</td>
</tr>
<tr>
<td>Linguistic Analysis</td>
<td>14</td>
</tr>
<tr>
<td>At least one</td>
<td>18</td>
</tr>
</tbody>
</table>

Table 7.3 Error type and its distribution over 50 natural language queries and 28 selected natural language queries in German

measures” relies on the correctness of the constructed DAG, i.e. on the performance of both the previous components (entity identification and linguistic analysis). Therefore, it is important to examine the performance of individual components. We evaluated the outcome of entity identification and linguistic analysis on all 50 NL-Queries of the test dataset. However, all of the template-based and SPARQL aggregation type NL-Queries are out of scope in our settings. Therefore, we discuss the results obtained for the remaining 28 NL-Queries. Table 7.3 shows that appropriate entities could not be found in 7 out of 50 NL-Queries and 1 out of the 28 queries. To evaluate the performance of the linguistic analysis component, we counted the number of NL-Queries for which the Stanford parser was unable to generate dependencies. The statistics of the errors in linguistic analysis are shown in Table 7.3. As explained in Section 7.3.2.3, to find the relevant properties associated with the selected DBpedia entity, a comparison of all the properties and the next term from the DAG is needed. We used following three settings in calculating the relatedness scores:

- **Translation with edit distance**: we translate the next term obtained from the DAG into English. However, the translated term might not match exactly with a property in DBpedia. Therefore, we further calculate Levenshtein edit distance [87] between the translated term and all the properties associated to the given entity, and select the best matched property which has minimum Levenshtein distance to the translated term.

- **Translation with NESA** Similar to previous setting, we translate the term into English, however, rather using edit distance, we calculate NESA score between the translated term and all the properties in DBpedia.
<table>
<thead>
<tr>
<th>NL-Queries in German and English</th>
<th>Translation with edit dist.</th>
<th>Translation with NESA</th>
<th>CLNESA</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Wer war der Nachfolger von John F. Kennedy?@de Who was the successor of John F. Kennedy?@en</td>
<td>1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0</td>
<td>1.0 1.0 1.0</td>
<td>1.0 1.0 1.0</td>
</tr>
<tr>
<td>2. Wer ist der Gouverneur von Texas?@de Who is the governor of Texas?@en</td>
<td>1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0</td>
<td>1.0 1.0 1.0</td>
<td>1.0 1.0 1.0</td>
</tr>
<tr>
<td>3. Aus welchem Land kommt der Schöpfer von Nijntje?@de Which country does the creator of Miffy come from?@en</td>
<td>0.33 1.0 0.5 0.5 0.5 0.5 0.33 1.0 0.5</td>
<td>1.0 1.0 1.0</td>
<td>1.0 1.0 1.0</td>
</tr>
<tr>
<td>4. Wer hat die Musik f¨ur Harold und Maude komponiert?@de Who composed the music for Harold and Maude?@en</td>
<td>1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0</td>
<td>1.0 1.0 1.0</td>
<td>1.0 1.0 1.0</td>
</tr>
<tr>
<td>5. Wer hat Intel gegründet?@de Who founded Intel@en</td>
<td>1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0</td>
<td>1.0 1.0 1.0</td>
<td>1.0 1.0 1.0</td>
</tr>
<tr>
<td>6. Wer war der Nachfolger von John F. Kennedy?@de Who was the successor of John F. Kennedy?@en</td>
<td>1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0</td>
<td>1.0 1.0 1.0</td>
<td>1.0 1.0 1.0</td>
</tr>
<tr>
<td>7. Wer hat die Musik f¨ur Harold und Maude komponiert?@de Who composed the music for Harold and Maude?@en</td>
<td>0.33 1.0 0.5 0.5 0.5 0.5 0.33 1.0 0.5</td>
<td>1.0 1.0 1.0</td>
<td>1.0 1.0 1.0</td>
</tr>
<tr>
<td>8. Wer ist der Schöpfer von Miffy?@de Which country does the creator of Miffy come from?@en</td>
<td>1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0</td>
<td>1.0 1.0 1.0</td>
<td>1.0 1.0 1.0</td>
</tr>
<tr>
<td>9. Aus welchem Land kommt der Schöpfer von Miffy?@de Which country does the creator of Miffy come from?@en</td>
<td>1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0</td>
<td>1.0 1.0 1.0</td>
<td>1.0 1.0 1.0</td>
</tr>
<tr>
<td>10. Wer hat die Musik f¨ur Harold und Maude komponiert?@de Who composed the music for Harold and Maude?@en</td>
<td>0.33 1.0 0.5 0.5 0.5 0.5 0.33 1.0 0.5</td>
<td>1.0 1.0 1.0</td>
<td>1.0 1.0 1.0</td>
</tr>
<tr>
<td>11. Wer hat die Musik f¨ur Harold und Maude komponiert?@de Who composed the music for Harold and Maude?@en</td>
<td>1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0</td>
<td>1.0 1.0 1.0</td>
<td>1.0 1.0 1.0</td>
</tr>
<tr>
<td>12. Wer hat die Musik f¨ur Harold und Maude komponiert?@de Who composed the music for Harold and Maude?@en</td>
<td>0.33 1.0 0.5 0.5 0.5 0.5 0.33 1.0 0.5</td>
<td>1.0 1.0 1.0</td>
<td>1.0 1.0 1.0</td>
</tr>
<tr>
<td>13. Wer hat die Musik f¨ur Harold und Maude komponiert?@de Who composed the music for Harold and Maude?@en</td>
<td>1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0</td>
<td>1.0 1.0 1.0</td>
<td>1.0 1.0 1.0</td>
</tr>
<tr>
<td>14. Wer hat die Musik f¨ur Harold und Maude komponiert?@de Who composed the music for Harold and Maude?@en</td>
<td>0.33 1.0 0.5 0.5 0.5 0.5 0.33 1.0 0.5</td>
<td>1.0 1.0 1.0</td>
<td>1.0 1.0 1.0</td>
</tr>
<tr>
<td>15. Wer hat die Musik f¨ur Harold und Maude komponiert?@de Who composed the music for Harold and Maude?@en</td>
<td>1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0</td>
<td>1.0 1.0 1.0</td>
<td>1.0 1.0 1.0</td>
</tr>
<tr>
<td>16. Wer hat die Musik f¨ur Harold und Maude komponiert?@de Who composed the music for Harold and Maude?@en</td>
<td>0.33 1.0 0.5 0.5 0.5 0.5 0.33 1.0 0.5</td>
<td>1.0 1.0 1.0</td>
<td>1.0 1.0 1.0</td>
</tr>
<tr>
<td>17. Wer hat die Musik f¨ur Harold und Maude komponiert?@de Who composed the music for Harold and Maude?@en</td>
<td>1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0</td>
<td>1.0 1.0 1.0</td>
<td>1.0 1.0 1.0</td>
</tr>
<tr>
<td>18. Wer hat die Musik f¨ur Harold und Maude komponiert?@de Who composed the music for Harold and Maude?@en</td>
<td>0.33 1.0 0.5 0.5 0.5 0.5 0.33 1.0 0.5</td>
<td>1.0 1.0 1.0</td>
<td>1.0 1.0 1.0</td>
</tr>
<tr>
<td>19. Wer hat die Musik f¨ur Harold und Maude komponiert?@de Who composed the music for Harold and Maude?@en</td>
<td>1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0</td>
<td>1.0 1.0 1.0</td>
<td>1.0 1.0 1.0</td>
</tr>
<tr>
<td>20. Wer hat die Musik f¨ur Harold und Maude komponiert?@de Who composed the music for Harold and Maude?@en</td>
<td>0.33 1.0 0.5 0.5 0.5 0.5 0.33 1.0 0.5</td>
<td>1.0 1.0 1.0</td>
<td>1.0 1.0 1.0</td>
</tr>
<tr>
<td>21. Wer hat die Musik f¨ur Harold und Maude komponiert?@de Who composed the music for Harold and Maude?@en</td>
<td>1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0</td>
<td>1.0 1.0 1.0</td>
<td>1.0 1.0 1.0</td>
</tr>
<tr>
<td>22. Wer hat die Musik f¨ur Harold und Maude komponiert?@de Who composed the music for Harold and Maude?@en</td>
<td>0.33 1.0 0.5 0.5 0.5 0.5 0.33 1.0 0.5</td>
<td>1.0 1.0 1.0</td>
<td>1.0 1.0 1.0</td>
</tr>
<tr>
<td>23. Wer hat die Musik f¨ur Harold und Maude komponiert?@de Who composed the music for Harold and Maude?@en</td>
<td>1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0</td>
<td>1.0 1.0 1.0</td>
<td>1.0 1.0 1.0</td>
</tr>
<tr>
<td>24. Wer hat die Musik f¨ur Harold und Maude komponiert?@de Who composed the music for Harold and Maude?@en</td>
<td>0.33 1.0 0.5 0.5 0.5 0.5 0.33 1.0 0.5</td>
<td>1.0 1.0 1.0</td>
<td>1.0 1.0 1.0</td>
</tr>
<tr>
<td>25. Wer hat die Musik f¨ur Harold und Maude komponiert?@de Who composed the music for Harold and Maude?@en</td>
<td>1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0</td>
<td>1.0 1.0 1.0</td>
<td>1.0 1.0 1.0</td>
</tr>
<tr>
<td>26. Wer hat die Musik f¨ur Harold und Maude komponiert?@de Who composed the music for Harold and Maude?@en</td>
<td>0.33 1.0 0.5 0.5 0.5 0.5 0.33 1.0 0.5</td>
<td>1.0 1.0 1.0</td>
<td>1.0 1.0 1.0</td>
</tr>
<tr>
<td>27. Wer hat die Musik f¨ur Harold und Maude komponiert?@de Who composed the music for Harold and Maude?@en</td>
<td>1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0</td>
<td>1.0 1.0 1.0</td>
<td>1.0 1.0 1.0</td>
</tr>
<tr>
<td>28. Wer hat die Musik f¨ur Harold und Maude komponiert?@de Who composed the music for Harold and Maude?@en</td>
<td>0.33 1.0 0.5 0.5 0.5 0.5 0.33 1.0 0.5</td>
<td>1.0 1.0 1.0</td>
<td>1.0 1.0 1.0</td>
</tr>
</tbody>
</table>

Table 7.4 Evaluation on 28 German NL-Queries
7.3 Cross-lingual natural language querying

- **CLNESA** In this setting, we do not perform any translation. We calculate CLNESA score between the term in German with all the properties in English DBpedia.

We use Bing\textsuperscript{12} to perform translation. Translation is not performed on the full text of a NL-Query but only on the properties because the quality of translation is not good enough to get the correct linguistic dependencies by using the Stanford parser.

In the first setting, we perform the translation and check if we can find the translated term in the listed properties by using Levenshtein edit distance [87] approximation. While in the second one, we calculate relatedness scores using NESA after performing automatic translation to investigate if automatic translation and semantic relatedness can complement each other. We do not use automatic translation in the third setting but only rely on the scores generated by CLNESA. The quality of the final results generated by all three settings are analyzed manually and shown in Table 7.4 and we discuss it in detail below.

### 7.3.3.3 Results and Discussion

Table 7.4 compares the results obtained by using the above described three different settings of CroNL. It shows that automatic translation can not bridge the vocabulary gap between NL-Queries and DBpedia. It means that there are large lexical variations in defining the relations of entities. Although the score generated by the combination improved significantly over the score obtained by just using automatic translation, the best results are generated by CLNESA. The reason may be that combined errors introduced by using both translation and NESA is more than the error generated by CLNESA. Table 7.3 shows that 5 out of these 28 NL-Queries pose at least one type of error (entity identification or linguistic analysis), meaning that DAGs can be generated only for 23 queries out of 28. To reduce this error, we consider that keywords appearing in a given NL-Query may depend on the selected entity. For instance, the Stanford parser failed to generate the correct dependencies for Q28 and Q5 (listed in Table 7.4) but by considering

\textsuperscript{12}https://www.bing.com/translator
the terms “groß” and “malte” to be dependent on the identified entities “Michael Jordan” and “Christus im Sturm auf dem See Genezareth” respectively, we could generate the correct DAGs. Therefore, we can test the third component of our approach on 25 out of 28 NL-Queries, as we got the correct DAGs for these 25 queries. We can see in Table 7.4 that by using translation we can retrieve the correct answers for 10 NL-Queries and partially correct for 2 NL-Queries; by using translation with the combination of NESA we can retrieve the correct answers for 15 NL-Queries and partially correct for 3 NL-Queries; and by using CLNESA we can retrieve the correct answers for 21 NL-Queries and partially correct for 3 NL-Queries.

In the case of Q10, automatic translation followed by NESA failed because when the system tried to find the maximum related property with the term “developed” (translation of “entwickelt”), it obtained a higher NESA score for another property “operating system” than “developer”. CroNL failed to find the results for Q18 due to the appearance of more than one highly related properties, such as “mission name”, “mission duration”, and “launch pad” with “Astronauten}@de and “astronauts”@en. We also report the performance of CroNL on the overall test dataset of 50 NL-Queries. The results are shown in Table 7.5.

<table>
<thead>
<tr>
<th></th>
<th>Average Precision</th>
<th>Average Recall</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Translation</td>
<td>0.217</td>
<td>0.24</td>
<td>0.228</td>
</tr>
<tr>
<td>Translation with NESA</td>
<td>0.34</td>
<td>0.386</td>
<td>0.361</td>
</tr>
<tr>
<td>CLNESA</td>
<td>0.459</td>
<td>0.506</td>
<td>0.481</td>
</tr>
</tbody>
</table>

Table 7.5 Evaluation on 50 German NL-Queries

7.4 Summary

This chapter presented EnRG, which is a big graph of connected entities based on Wikipedia. Given an entity, it retrieves a ranked lists of related entities for different DBpedia types. It provides further exploration on the results based on the specific types given by YAGO classes. Although EnRG-UI is a very useful tool to
explore further about Wikipedia entities, it can benefit other related tasks such as query expansion by providing related entities to the search query, and document semantic enrichment by proving entities related to the text in a document. Further we presented MCR, a knowledge base lookup task in which terms expressed in medical text are identified in UMLS. We presented a set of new algorithms for performing MCR and showed that our methods outperformed state-of-the-art methods. In particular, we investigated that different similarity measures and context effect the overall performance. We also presented CroNL that performs NL-Query over DBpedia across different languages. CroNL was evaluated against 50 NL-Queries in German over the English DBpedia, and achieved an average precision of 0.459, an average recall of 0.506 and F1 score of 0.361. However, on the NL-Queries that can be covered by CroNL, the system achieved an average precision of 0.815, an average recall of 0.831 and a F1 score of 0.823. We showed that our cross-lingual relatedness measure outperforms the automatic translation for cross-lingual NL-Querying over DBpedia.
Chapter 8

Conclusion

In this thesis, we investigated different techniques to exploit Wikipedia knowledge in developing an entity relatedness and recommendation approach. We proposed a distributional semantic model for entity relatedness (DiSER) that is based on the distributional hypothesis discussed in psycholinguistic and computational linguistic studies. The developed relatedness measure DiSER is used to build an entity relatedness graph (EnRG) that provides a ranked list of related entities. We combined DiSER with other features extracted from Wikipedia to build an entity recommendation approach specifically for the web search scenario. We evaluated DiSER in different application specific tasks like entity ranking and entity disambiguation. Further, we proposed an improved text relatedness model “Non-Orthogonal Explicit Semantic Analysis (NESA)” that takes benefit from DiSER.

8.1 Summary

This thesis proposes the following contributions to the study of entity relatedness and entity recommendation.

1. Entity Relatedness

   • We developed an entity relatedness measure called “Distributional Semantics for Entity Relatedness (DiSER)” that builds high dimensional vectors of entities to calculate their relatedness score. DiSER builds
distributional vectors by considering only the manually annotated entities in Wikipedia articles. Therefore, it builds unambiguous distributional vectors for ambiguous surface forms of the given entities. We showed in chapter 4 that DiSER outperforms state of the art methods and improves by 12% over the best performing entity relatedness measure. We also show an evaluation of DiSER in the Entity Disambiguation task on a dataset of 50 sentences with highly ambiguous entity mentions. It shows an improvement of 10% in precision over other existing methods.

• DiSER is limited to generate the distributional vectors for entities that have Wikipedia pages. Although, the English Wikipedia covers more than 4 million entities, it may not contain many of the entities that are relatively new or recently became popular. Therefore, to overcome this issue, we proposed an approach called Context-DiSER to generate the DiSER vectors for long tail and non-popular entities which do not have a Wikipedia page.

2. Entity Recommendation

• A very big graph called Entity Relatedness Graph (EnRG) is built by using DiSER. EnRG is constructed by calculating the DiSER scores between 16.83 trillions of entity-pairs (4.1 millions x 4.1 millions). We implemented a sparse matrix multiplication and reduce the number of computations to 98%. With our implementation, EnRG can be built with in \(\approx 48\) hours. Similar to major search engines, EnRG recommends related entities for a given entity that provides an opportunity to the users to explore about things related to their favorite topics.

• We developed a supervised entity recommendation approach called “WiFER” that combines DiSER with other Wikipedia based features by using a learning to rank method. All the features are investigated in detail to provide a deep insight into the characteristics of different features. The main contribution of this work is to develop an entity recommendation approach that uses the publicly available datasets.
• Evaluation of our approach for entity recommendation by comparing DiSER and other Wikipedia-based features with a commercial entity recommendation system “Spark” which uses several features from proprietary data like query logs and search session. A comprehensive study of the importance of different features is provided in chapter 5. We showed that WiFER achieves a comparable accuracy to Spark. Moreover, we combine WiFER with Spark features and compute the feature importance in our learning algorithm. This showed that there are 5 Wikipedia based features in the top 10 most effective ones. This also shows the effectiveness of DiSER as it appears in the top 5 most effective features among 128 in total.

3. **Non-orthogonal Explicit Semantic Analysis (NESA)**

• NESA reduces the orthogonality issue between explicit concepts in the ESA model without compromising with the explicit property of the ESA concept space. We use different entity relatedness measures to calculate the relatedness scores between explicit concepts. In chapter 6, we showed that the accuracy of word relatedness varies with the different entity relatedness measures. We showed that NESA outperforms the existing methods for word and text relatedness. Particularly, NESA that uses DiSER to compute concept relatedness achieves the highest correlation with the gold standard for word relatedness.

EnRG and Medical Concept Resolution (MCR) are two applications we built based on the methods described above. Section 7.1 describes several functionalities provided in EnRG. We provide a REST API that takes an entity as query and returns a ranked list of related entities. These related entities provided in EnRG can be used in different applications such as query expansion, feature vector generation and user interest mining. We also use our relatedness measure in the MCR approach. MCR identifies the most appropriate medical concept for a given medical factor in a natural language question. We developed MCR for a commercial question answering system called “Watson” [44].
8.2 Lessons Learned

Even though we presented a novel and advanced approach over state of the art methods for measuring relatedness between entities, the problem is not completely solved yet. We leaned about different hidden challenges of measuring entity relatedness, which we discuss as follows:

1. **Temporal shift**
   
   - In chapter 5, we evaluated DiSER on KORE dataset [62] as it has been used by other existing methods for calculating entity relatedness. KORE dataset was prepared with a high inter-annotator agreement for ranking the related entities. The annotators can only judge the relatedness based on their current intuition about the given entities, while relatedness between entities may change over time. For instance, due to ongoing elections in a country, two politicians could be very related at the moment but it is not necessary that they will remain related with the same strength for long time.
   
   - It is hard to address the temporal changes in preparing a gold standard dataset. Therefore, we can not make a concrete conclusion about different entity relatedness measures based on their performance on a single static dataset like KORE.

2. **Localized relatedness**
   
   - The annotations about the relatedness between popular entities may have a high agreement across different locations. However, in case of long tail entities, it highly depends in which contexts the given entity is popular for a particular location. For instance, an athlete might become recently popular in a country due to loosing against one player, while in another country for getting married with someone. These locations might prefer different sets of related entities for the athlete.
   
   - Some of the entities are only popular in a particular country and may have other entities which are related due to a very specific context.
For instance, a regional actor might be known in a country but not popular worldwide.

3. **Incomplete knowledge bases**

- Most of the knowledge bases are incomplete. For instance, DBpedia does not have “DateOfBirth” relation for 70% people. Moreover, these knowledge bases can not be completed as there will always be new entities and their relations. Therefore, a knowledge base like DBpedia can not provide different aspects of entities which are required to capture their relatedness.

4. **Entity relatedness in real world**

- In chapter 2, we define entity relatedness by the degree of associativity between them. In real world, humans define the associativity between given entities by their knowledge and experience about them, which is generally based on their co-occurrence in similar contexts. For instance, two entities can be considered very related in the context of an ongoing elections in a country if they are competing against each other. Therefore, we can intuitively hypothesize that the co-occurrence based model can better capture the relatedness between the entities.

### 8.3 Directions for Future Research

We believe that this research constitutes a step towards intelligent web search by providing an opportunity to users to interact with the available world knowledge. Our research addresses different important components to support an intelligent search, such as by recommending related entities to explore further, linking text to knowledge base, and querying over knowledge bases. Further, this study opens up different directions for future research.
8.3.1 Relationship Explanation

Major search engines like Google and Yahoo have greatly simplified the knowledge discovery process by recommending related entities with natural language explanations of their relatedness. However, they often fail to provide explanations of relatedness between long tail entities. For example, Google recommends “Aristotle”, “Olympias”, “Chandragupta Maurya” and others for the entity “Alexander the Great”\(^1\). Since these entities are relatively less popular, Google fails to provide any explanation about the relatedness of “Chandragupta Maurya”\(^2\) and “Alexander the Great”. However, the Wikipedia article of “Alexander the Great” contains the information about their relatedness that “the power vacuum he left in the northwest of the Indian subcontinent directly gave rise to one of the most powerful Indian dynasties in history, the Maurya Empire. The Empire was founded in 322 BCE by Chandragupta Maurya.”. Moreover, obtaining natural language explanations about many related entities is not trivial. Thus, a path of connected entities can provide some insights into explaining the relationships between entities. For instance, a path in DBpedia i.e. “Alexander the Great” → “Maurya Empire” → “Chandragupta Maurya” explains the relationship.

EnRG can also find weakly related entities. This provides a research direction into discovering hidden relations between weakly related entities. For instance, EnRG recommends the music band “Backstreet Boys” in relation to “Brad Pitt”. The relationship can be explained as A.J. McLean\(^3\) from “Backstreet Boys” played a similar character as of “Brad Pitt” in his version of “Fight Club”.

8.3.2 Entity Similarity vs Relatedness

This thesis focuses on entity relatedness as it covers a broad range of relations and capture the strength of their associativity. In chapter 2, we discussed different examples to show that the similar words can be considered related but it is not necessary that all the related words can be considered similar. However, en-

\(^1\)https://en.wikipedia.org/wiki/Alexander_the_Great
\(^2\)https://en.wikipedia.org/wiki/Chandragupta_Maurya
\(^3\)http://en.wikipedia.org/wiki/A._J._McLean
Entity relatedness is different from word relatedness in the sense that we can find two unrelated entities sharing similar contexts. For instance, in the sentence “The best action movie starring Brad Pitt I have seen so far,” if we replace Brad Pitt with some other actor who has never co-occurred with him in any text on the web, would not necessarily make these two actors related. However, these two actors can be considered similar as they appear in very similar context. Therefore, entity similarity defines a different notion than word similarity, as words with a high substitutability also associate with each other due to their functional similarities. However, in case of similar entities, it is not necessary to have many functional similarities but only the popular ones. For instance, an action Hollywood actor can be considered similar to an action Bollywood actor but they might not be related to each other. Therefore, the problem of measuring entity similarity requires a deep investigation to identify important features and appropriate methodology that can differentiate between similarity and relatedness for entities.
References


[17] Basile, P., Musto, C., de Gemmis, M., Lops, P., Narducci, F., and Semeraro,


[89] Li, F., Huang, M., and Zhu, X. (2010). Sentiment analysis with global topics and local dependency. In AAAI.


