Title	Unsupervised deep language and dialect identification for short texts
Author(s)	Goswami, Koustava; Sarkar, Rajdeep; Chakravarthi, Bharathi Raja; Fransen, Theodorus; McCrae, John P.
Publication Date	2020-12
Publication Information	Koustava Goswami, Rajdeep Sarkar, Bharathi Raja Chakravarthi, Theodorus Fransen, and John P. McCrae. 2020. Unsupervised Deep Language and Dialect Identification for Short Texts. In Proceedings of the 28th International Conference on Computational Linguistics, pages 1606–1617, Barcelona, Spain (Online). International Committee on Computational Linguistics.
Publisher	International Committee on Computational Linguistics
Link to publisher's version	https://doi.org/10.18653/v1/2020.coling-main.141
Item record	http://hdl.handle.net/10379/17393
DOI	http://dx.doi.org/10.18653/v1/2020.coling-main.141

Unsupervised Deep Language and Dialect Identification for Short Texts

Koustava Goswami, Rajdeep Sarkar, Bharathi Raja Chakravarthi,
Theodorus Fransen, and John P. McCrae
Insight SFI Research Centre for Data Analytics,
Data Science Institute, National University of Ireland Galway
{koustava.goswami, rajdeep.sarkar, bharathi.raja,
theodorus.fransen, john.mccrae}@insight-centre.org

Abstract

Automatic Language Identification (LI) or Dialect Identification (DI) of short texts of closely related languages or dialects, is one of the primary steps in many natural language processing pipelines. Language identification is considered a solved task in many cases; however, in the case of very closely related languages, or in an unsupervised scenario (where the languages are not known in advance), performance is still poor. In this paper, we propose the Unsupervised Deep Language and Dialect Identification (UDLDI) method, which can simultaneously learn sentence embeddings and cluster assignments from short texts. The UDLDI model understands the sentence constructions of languages by applying attention to character relations which helps to optimize the clustering of languages. We have performed our experiments on three short-text datasets for different language families, each consisting of closely related languages or dialects, with very minimal training sets. Our experimental evaluations on these datasets have shown significant improvement over state-of-the-art unsupervised methods and our model has outperformed state-of-the-art LI and DI systems in supervised settings.

1 Introduction

Automatic Language Identification (LI) and Dialect Identification (DI) has become a very crucial part of natural language processing (NLP) pipelines and is part of tasks such as language modelling, categorization and analysis of code-mixed datasets. Many researchers have carried out experiments in these domains both in speech and texts starting from House and Neuburg (1977). Unsupervised LI is an under-explored area, but is highly useful as it can exploit the large amount of unlabelled data and, more importantly, can be employed when the languages to be identified are not known in advance. However, unsupervised LI for short texts is a very difficult task, for which performance is still comparatively poor. Zhang et al. (2016) have explored unsupervised language identification but this does not work well with closely related short texts. Zaidan and Callison-Burch (2014) have worked on the identification of Arabic dialects. Ciobanu et al. (2018) studied German dialect identification. These works are mostly supervised works. The task is even harder when it comes to unsupervised DI.

These previous works have generated some questions: What is the best way to construct sentence embeddings and how to cluster them efficiently for short texts when there is no labelled training data? How to address the hard task of DI and LI for closely related languages in an unsupervised way without any manual intervention? In this paper, we address these problems by taking inspiration from iterative clustering (Xie et al., 2016) and self-attention-based sentence embeddings from Lin et al. (2017). Both of the papers have shown good results on unsupervised clustering and sentence embeddings for a single language dataset whereas our goal is to design a system which can work on different short texts of closely related languages and understand every distinct feature. Our system takes different n-gram character features of a sentence and computes attention weights based on their importance in sentence construction. We then fine-tune sentence embeddings with an iterative clustering process with the help of the Stochastic

Proceedings of the 28th International Conference on Computational Linguistics, pages 1606–1617
Barcelona, Spain (Online), December 8-13, 2020

Gradient Descent (SGD) and backpropagation. We call the system Unsupervised Deep Language and Dialect Identification (UDLDI).

One of the main challenges of the model is to get sentence embeddings. As we do not have any labels here, we cannot use typical loss functions to update character-to-sentence embeddings. Further, as the datasets are short text small datasets, there is much less scope to learn features to construct efficient sentence embeddings. To overcome this challenge we propose a loss function which considers the probability distribution of assigning sentences in each cluster and tries to maximize the unique sentence assignments in each cluster. These sentence embeddings define the initial cluster centre assignments, which we then fine-tune with the iterative clustering learning process. Our method is tested against three different datasets consisting of under-resourced and closely related languages, the language families and/or varieties in question being South African, Dravidian and Swiss German. The South African dataset consists of languages from different language families, some of which are similar. We have achieved good results on each dataset and the results are better than other base results. We have also compared our sentence-embedding model to supervised LI and DI methods which outperformed other state-of-the-art models. The robustness of the model is, it does not require any manual intervention and it works well for short texts in any dataset consisting of closely related languages, for any language family.

Our contributions are: (1) the mentioned UDLDI model for language and dialect identification in short texts of closely related languages, (2) efficient sentence embedding construction for short texts, and (3) efficient clustering of closely related languages with a very small dataset.

2 Related Work

There are some LI and DI models which work efficiently for short texts and closely related languages on different datasets (Hammarström, 2007; Medvedeva et al., 2017). Jauhiainen et al. (2019) have described different LI and DI models. Language identification of closely related languages are explored in Malay-Indonesian languages (Ranaivo-Malançon, 2006), Indian languages (Murthy and Kumar, 2006), South Slavic languages (Tiedemann and Ljubešić, 2012; Ljubešic and Kranjcic, 2014; Ljubešić and Kranjčić, 2015). Zampieri and Gebre (2014) made a LI system which can identify 27 languages.

There is some research on LI for short texts. Vatanen et al. (2010) explored an LI method based on n-gram character sequences for messages that are 5-21 characters long. Bergsma et al. (2012) have explored LI for Twitter data. King and Abney (2013) used Conditional Random Fields (CRF) (Lafferty et al., 2001) to build word-level LI method. Giwa and Davel (2013) have investigated LI in terms of code-switching. These methods are all supervised and this does not allow them to exploit the large amount of unlabelled data and situations where the languages are not known in advance. Unsupervised LI is a challenging task. Selamat and Ching (2008) used Fuzzy ART NNs for clustering of documents written in Arabic, Persian, and Urdu. In Fuzzy ART, they have shown how to update clusters dynamically. Amine et al. (2010) used a character n-gram representation for text. Later k-means algorithm followed by particle-swarm optimization is applied to cluster the languages which results in different small clusters. Shiells and Pham (2010) worked on unsupervised tweet LI with the Chinese Whispers algorithm of Biemann (2006) and Graclus clustering. Elfardy and Diab (2012) deal with code-switching identification between Modern Standard Arabic (MSA) and dialectal Arabic. Wan (2016) used word-level k-means clustering for LI and clustered languages based on co-occurring words. Poulston et al. (2017) used word2Vec word embeddings and k-means clustering to make their LI model. Rijhwani et al. (2017) researched short-text tweets for unsupervised language detection which mainly focuses on seven languages. These algorithms work well for some specific languages and long texts.

There is some research on dialect identification (DI). DI has been explored in Serbo-Croatian dialects (Zečević and Vujicic-Stankovic, 2013), English varieties (Simaki et al., 2017), Dutch dialects (Trieschnigg et al., 2012), German dialects (Hollenstein and Aepli, 2015), Mainland–Singaporean–Taiwanese Chinese (Huang and Lee, 2008), Arabic dialects in social media (Huang, 2015), Portuguese varieties (Zampieri and Gebre, 2012). One of the more famous dialect identification systems is an Arabic dialect system which was given much emphasis in 2015 (Zampieri et al., 2015). But most of these works deal with supervised dialect identification. Our model bridges the gap between DI and unsupervised learning.

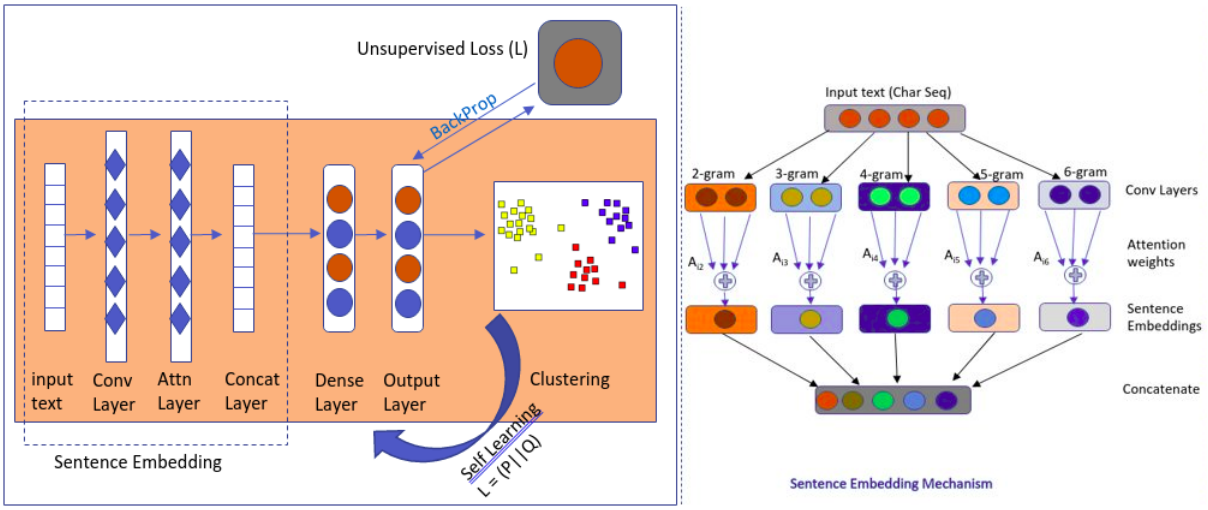

Figure 1: Unsupervised deep LI and DI model. The left-hand side shows the architecture design of the UDLDI mechanism; the sentence embedding mechanism is explained on the right-hand side

3 Unsupervised Deep Language and Dialect Identification (UDLDI)

Unsupervised Deep Language and Dialect Identification (UDLDI) clusters the short texts of closely related languages in an efficient way using character-to-sentence embeddings with self-attention, based on an unsupervised loss function and an iterative clustering method (refer to Figure 1). The model uses an unsupervised loss function to build the character-to-sentence embeddings. The iterative clustering process fine-tunes the sentence embedding and enhances the cluster assignment.

3.1 Sentence Embedding

The sentence-embedding model has two parts. The first one is an n-gram character-level CNN and the second one is a self-attention mechanism, which gives the weighted summed features of sentences based on n-gram ($n \in \{2,3,4,5,6\}$) characters. The attention weights are then multiplied with the feature vectors of the CNN to get the sentence embedding. The different character-level features are achieved with the help of 1-dimensional CNN (Zhang et al., 2015). The sentence embeddings are then fed into a dense layer to get the language level probability distribution of the sentence.

The 1-dimensional CNN accepts the input sequence as a character sequence. If a sentence has m characters then the input sequence would be $S = [w_1, w_2, ..., w_m]$, where w_i is a character in the sentence (including spaces). The one-dimensional convolution implements 1-dimensional filters which slide over the sentences as a feature extractor. Let the filters have a shape of $1 \times k$ where k is the filter size. Filter sizes are the same with different n-grams. Let $x_i \in \mathbb{R}^d$ denote the d-dimensional character embedding of the i-th character considering the character vocabulary size is n. For each position j in the sentence, we have a window vector w_j with k consecutive character vectors, as denoted in Equation 1.

$$w_j = [x_j, x_{j+1}, ..., x_{j+k-1}] \tag{1}$$

The 1-dimensional k-sized filters slide over the window vector w_j to create the feature map s where $s \in \mathbb{R}^{m-k+1}$ and m is the input size. Each element s_j of the feature map for window vector w_j is produced according to Equation 2,

$$s_j = a(w_j \cdot m + b_j) \tag{2}$$

where vector m is a filter for convolution operation, b_j is the bias for the j-th position and a is the non-linear function. The new feature representation F after rearranging will be represented as $F = [s_1, s_2, ..., s_n]$, where n is the number of convolution filters, s_i is the feature map generated with i-th filter and $F \in \mathbb{R}^{(m-k+1)\times n}$.

To understand the different n-gram feature representation of the sentences, multiple *k*-sized filters are used which is represented as F_k where $k \in \{2,3,4,5,6\}$.

We aim to get sentence embeddings from different n-gram representations. We have achieved this by getting the weighted summation of *T* elements of new feature representation *F*. We perform this for each F_k separately. To achieve this, we use a self-attention mechanism. The attention mechanism takes the feature representation as input and gives *a* as an output weight vector, as shown in Equation 3,

$$a = \text{softmax}(\tanh(W_h \cdot F^T + b_h)) \tag{3}$$

where W_h is the weight matrix and b_h is the bias for the attention mechanism. The use of softmax is to ensure that the summation of attention weights $(a_1, a_2, .., a_T)$ are non-negative and sum up to 1. Then we sum up the elements of feature representation *F* according to the weight vectors provided by *a* to get a vector representation *r* of the input sentence by Equation 4,

$$r = \sum_{i=1}^{T} a_i \cdot F_i \tag{4}$$

where a_i is the attention weights for each element of the feature representation, and $\cdot$ is the element-wise product between elements.

Different n-gram representations carry different weighted sentence embeddings. To capture all the representations, the final representation is a concatenated vector $r = [r_2, r_3, r_4, r_5, r_6]$ (where n-grams $\in \{2,3,4,5,6\}$), which is then passed to the fully connected layer to get the probability distribution of there being one language or dialect.

The sentence embeddings are used in two phases. The first phase is the pre-training phase which can be obtained as described in section 3.2; the second phase is the fine-tuning of sentence embeddings based on self-training with the clustering algorithm, described in section 3.3.

3.2 Dense Layer and Loss function

The sentence embeddings $r \in \mathbb{R}^{N \times M}$ are passed to a fully connected (FC) layer which gives the output $z \in \mathbb{R}^{N \times K}$ and then a softmax layer ($p \in \mathbb{R}^{N \times K}$) returning the probability distribution of all classes, as per Equation 5,

$$p_{ij} = \frac{exp(z_{ij})}{\sum_{t=1}^{K} exp(z_{ik})} \tag{5}$$

where *K* is the number of languages or dialects and p_{ij} gives the probability that sample *i* belongs to class *j* ($j \in \{1,2,...,K\}$).

The loss function maximizes the same feature sentence assignment to one language. It can be considered as maximizing cluster purity based on the probability distribution. The loss function can be computed by means of Equation 6,

$$L_u = \sum_{i=1}^{N} \max_{j=1}^{i} p_{ij} - \max_{i=1}^{N} \sum_{j=1}^{i} p_{ij}^2 \tag{6}$$

where *N* is the batch size. In contrast to existing loss functions, our loss function does not depend on entropy-based calculation as there is no label. Instead the loss function is trying to maximize the probability distribution function based on the feature assignments on each class (or cluster) and minimize the maximum summation of the squared probability distribution of all the languages assigned to one class. The maximization of the probability function ensures the best possible assignment of the features in each individual classes. But in this process, there is a chance of biased feature assignments (for the datasets where features of data points are very near in distance) to a single class. To overcome this trivial situation, the model is penalised with the maximum summation of the feature assignments over all samples. This stabilizes the system by assigning features to different classes in the best possible way.

After training the character embeddings and sentence embedding we cluster the dataset. Xie et al. (2016) have shown in the Deep Embedding for Clustering (DEC) architecture how to use clustering as

self-training (Nigam and Ghani, 2000) and how to fine-tune the mapping and do the clustering. The DEC framework uses the idea of parameterized non-linear mapping from the data space *X* to a lower-dimensional feature space *Z* to optimize a clustering objective. It uses stochastic gradient descent (SGD) via backpropagation on the clustering objective to learn the mapping, which is parameterized by a deep neural network. In our case, we have made use of this iterative clustering process to fine-tune the sentence embeddings, which improve the clustering.

3.3 Self Training and Clustering

The initialization of this phase starts by obtaining initial cluster centroids $u_j{}_{j=1}^{k}$ by applying the cluster algorithm to the sentence embeddings from the pre-training phase, where *k* is the set of cluster centres. The self-training phase has the following two steps: (i) the soft assignment between sentence embedding and initial cluster centroids, and (ii) fine-tuning sentence embeddings and cluster centroids using the auxiliary target distribution.

As a first step, we calculate the soft assignment between the cluster centroid u_j and sentence embedding z_i based on Student's t-distribution (Maaten and Hinton, 2008) with a single degree of freedom ($\alpha = 1$), as per Equation 7,

$$q_{ij} = \frac{\left(1 + \|z_i - u_j\|^2\right)^{-1}}{\sum_{j'} \left(1 + \|z_i - u_{j'}\|^2\right)^{-1}} \tag{7}$$

where $z_i \in \{\mathrm{Z}\}$ and q_{ij} can be considered as the probability of assigning sample *i* to cluster *j*.

The second step involves refining the cluster learning from their high confidence assignments with the help of an auxiliary target distribution (Xie et al., 2016). The auxiliary target distribution (p_{ij}) helps to improve cluster purity and puts more emphasis on data points assigned with high confidence. The p_{ij} can be calculated by means of Equation 8,

$$p_{ij} = \frac{q_{ij}^2 / \sum_i q_{ij}}{\sum_{j'} \left(q_{ij}^2 / \sum_i q_{ij'}\right)} \tag{8}$$

where $\sum_i q_{ij}$ is the frequency of the soft clusters.

The training loss is then designed based on KL divergence loss between the soft assignments q_i and the auxiliary distribution p_i, as shown in Equation 9.

$$L = KL\left(P \| Q\right) = \sum_i \sum_j p_{ij} \log \frac{p_{ij}}{q_{ij}} \tag{9}$$

4 Experimental Setup

4.1 Data

We have taken three different datasets for the training and testing of our model, for the following languages/dialects.

South African languages. This dataset was used by Duvenhage et al. (2017) and consists of short texts for 11 South African languages, many of which are related; these are the Nguni languages (zul, xho, nbl, ssw), Afrikaans (afr) and English (en), the Sotho languages (nso, sot, tsn), tshiVenda (ven) and Xitsonga (tso). The texts are on average 15-20 characters long. We have taken 15,000 samples for training and 5,000 for testing.

Dravidian languages are under-resourced languages spoken mainly in the southern part of India. This dataset contains the languages Tamil (ISO 639-3: tam), Telugu (ISO 639-3: tel), Malayalam (ISO 639-3: mal) and Kannada (ISO 639-3: kan) (Chakravarthi et al., 2018). They have different orthographies with a unique script for each language (Chakravarthi et al., 2019). The training set contains a total of around 14,000 instances and the test set contains a total of 3,000 instances.

Swiss German. This dataset is based on the ArchiMob corpus (Samardžić et al., 2016) and is used for dialect identification. It contains transcriptions of 34 interviews with native speakers of various German

dialects spoken in Switzerland. The subset used for German Dialect Identification contains 18 interviews (14 for training and 4 for testing) from four Swiss German dialect areas, i.e., Basel, Bern, Lucerne, and Zurich, with each interview being transcribed using the 'Schwyzertutschi Dialäktschrift' writing system (Dieth, 1986). The training set contains a total of around 14,000 instances and the test set contains a total of 3,638 instances.

4.2 Evaluation Metric

We have used the standard cluster performance evaluation process. The number of clusters is set to the same number as the original number of classes in the data and evaluated with unsupervised cluster accuracy, as shown in Equation 10,

$$ACC = max_m \frac{\sum_{i=1}^{n} 1(l_i = m(c_i))}{n} \tag{10}$$

where l_i is the ground truth label, c_i is the cluster assignment produced by the algorithm and m ranges over all possible one-to-one mappings between clusters and labels.

The number of clusters is also an evaluation process in case of unsupervised clustering. The accuracy is determined with the same number of clusters as the ground truth data. But in general, the number of clusters also needs to be optimised and evaluated. It is evaluated by Normalized Mutual Information (NMI). The NMI is useful for comparing the clustering results with different cluster numbers. It is shown in Equation 11,

$$NMI(l, c) = \frac{I(l, c)}{\sqrt{H(l)H(c)}} \tag{11}$$

where l is the ground truth label and c is the predicted cluster. I is the mutual information and H is entropy. When data is partitioned perfectly, the NMI score is 1, and when l and c are independent, it becomes 0.

4.3 Training Details

We implemented our model using pytorch[1] and used the SGD (Stochastic Gradient Descent) optimizer. The starting learning rate for the South African, Dravidian and Swiss German datasets was hand-tuned to 1e-4, 5e-3 and 2e-3, respectively, based on training data convergence during training time. We have used LambdaLR[2] as the learning rate scheduler which varies based on the filter numbers of convolution network. The convolution networks have 128 filters each and it has 1 stride. The number of neurons in the feed-forward network is 1024. We have used *k*-means clustering (MacQueen, 1967). For initial cluster centroid detection on batches, we have used minibatch *k*-means clustering[3]. The weight decay is 0.01 and the momentum is 0.9. For supervised learning, we have used Cross-Entropy loss [4].

5 Results and Discussions

We have compared our results with various baselines, both for supervised and unsupervised setup, which are provided in Table 1 and Table 2.

Unsupervised setup. The model is compared with one of the most well known unsupervised clustering methods Auto-encoder based *k*-means clustering. The sentences are projected in a latent space *Z* from the feature space *X* and then clustered it applying *k*-means. Our second baseline model is fasttext[5] word embeddings and *k*-means. The sentence embedding is the average of word embeddings. We have also compared our experimental results with the LSTM sentence-embedding model (Lin et al., 2017) along with the DEC framework (Table 1). Our model performed substantially better than other state-of-the-art models, achieving 37% accuracy (16.82% improvement upon the LSTM model) for the South African

[1] https://pytorch.org

[2] https://pytorch.org/docs/stable/optim.html

[3] https://scikit-learn.org/stable/modules/generated/sklearn.cluster.MiniBatchKMeans.html

[4] https://pytorch.org/docs/master/generated/torch.nn.CrossEntropyLoss.html

[5] https://fasttext.cc/docs/en/unsupervised-tutorial.html

Model	South African Dataset	Dravidian Dataset	Swiss German Dataset
AE + Kmeans	19.04%	51.00%	14.00%
Fasttext +Kmeans	21.00%	67.94%	20.00%
LDA (Zhang et al., 2016)	21.56%	69.34%	19.00%
Skip-Thought (Kiros et al., 2015) + Kmeans	24.67%	71.00%	20.35%
AE + DEC (Xie et al., 2016)	25.00%	74.97%	22.35%
LSTM (Lin et al., 2017) + DEC	31.67%	81.50%	26.00%
UDLDI	**37.00%**	**82.00%**	**30.00%**

Table 1: Accuracy of different models for unsupervised LI and DI

Model	South African Dataset	Dravidian Dataset	Swiss German Dataset
BRNN (Kocmi and Bojar, 2017)	89.00%	93.50%	—
NB+Lex (Duvenhage et al., 2017)	96.00%	97.34%	—-
CLUZH (Clematide and Makarov, 2017)	—	—	67.00%
MAZA (Malmasi and Zampieri, 2017)	—	—	68.00%
tearsofjoy (Zampieri et al., 2019)	—	—	75.00%
LSTM (Lin et al., 2017)	96.67%	98.51%	68.67%
CharCNN (Zhang et al., 2015)	98.00%	96.56%	66.09%
UDLDI	**99.01%**	**99.34%**	**81.00%**

Table 2: Accuracy of different models for supervised LI and DI

dataset. It has also achieved 82% accuracy (.6% improvement upon the LSTM model) in the Dravidian dataset. For the Swiss German dataset, 30% accuracy (15.83% improvement upon the LSTM model) was achieved; the latter DI task can be considered one of the toughest tasks to perform, as the dialects are very similar. It means that the model is well capable of identifying important character n-gram features in sentence construction. Details about the attention weights distribution are discussed in the section 5.1.

Supervised setup. Table 2 shows different baseline models including previously submitted systems in the Swiss German DI shared tasks. Our model has outperformed all the state-of-the-art methods achieving 99%, 99.34% and 81% accuracy for the South African dataset, Dravidian dataset and Swiss German dataset, respectively. Though it has outperformed both the LI and DI method, significant efficiency can be observed in DI which has outperformed the highest performing system (Zampieri et al., 2019) by 8%.

5.1 Discussion and Analysis

To analyze sentence embeddings we will take reference of the Swiss German DI which can be considered one of the hardest tasks. The dialects are very closely related from four different parts —Basel (BS), Bern (BE), Lucerne (LU), and Zurich (ZH). Some of the sentence constructions are very similar to each other. For example we can take two sentences of Bern and Lucerne which are "*aber dää ìsch emaal dä*" and "*aber dää ìsch seer schträng ggsìì mit öis*" respectively. In the sentence constructions, we can see the first three words are exactly the same consisting of the same characters even though they represents two different dialects. Without extensive morphological and linguistic knowledge, it is very difficult to separate sentences spoken by people from these regions. Character n-grams represents distinct features of sentences of a particular region. Our model is also able to identify dialectal (pronunciation) variants for an inflected form of a verb. We can see this in Figure 2. There are two sub-columns under the "Text" column which consist of different sentences formed with the same verb in different contexts. The red colour is the importance degree of the attention weight and light blue is a less important part. The English word "have" has different forms in different dialects. For example, in case of BE, it is written as "hei" whereas in LU it is written as "hend". The verb form is used in different contexts in different sentences. In the first example in the Text1 column, a sentence is shown with words "mir hei" which means "we have". But the "mir" word is the same in different dialects. Our model has given less attention weight to this word whereas the highest attention weight is given to "hei". The character n-gram considered space as well. So

Text		Language
Text1	Text2	
und mir hei ja pro bateljoon nume zwei iikaa ghaa	wasmer do zeersch überchoo hei weissi nìd sölmer	BE
und mir hend glik ghaa isch immer es basler batelioon isch choo	oder die hend det unde	LU
und mir häi also s familielääbe	oder woo wo de wo ebe de alarm gsìì ìsch häi ìsch häi mir öisi	BS
und mir händ asoo seer mager müse dure	unfäll ghaa händ und züüg und sache soll me das fiire	ZH

Figure 2: Attention visualization of dialect-specific words pointed out by the model

in the Text2 column, though the sentence does not contain the word “mir”, depending on the “hei” word with space, the sentence is classified as BE. Here the word “söimer” is also given the highest attention weight as the word only appeared in BE. The same can be observed in other sentences as well for other dialects. Conjunctions like “und” (“and”) are very common in sentences and are often written the same. As shown in Figure 2, this word occurs both in BS and ZH, and as such has very low significance in terms of dialect identification. Consequently, our model has assigned it a low attention weight.

As Scherrer et al. (2019) have pointed out, the ArchiMob corpus shows variation on different levels. Not only are different lexical items pronounced and hence written differently across dialects, the data additionally incorporates intra-speaker variability (i.e., slight variation in the pronunciation of the same word by the same speaker) and even shows transcriber-related variation. These factors may overestimate the distinctive features as identified by the model. Moreover, the attention weights in our model may point to yet another type of variability, namely, idiosyncratic features of certain speakers such as the repetition of words (idiolect). In Figure 3a dialects are compared with each other in pairs. The word “ùùnd” (standard German “und”) in the second example is common in both LU and BS. But in the case of LU, the word followed by “ùnd” whereas in case of BS it followed by other words, so the character attention distribution gives the highest attention weights to the n-gram “ùnd ùn”. Though the word “ùnd” (like “und”) has very low significance in dialect identification as an individual word, in the Swiss German dataset this n-gram is a distinctive feature, although probably being more indicative of idiolect than dialect. The n-gram “im va” carries very little distinctiveness in case of LU, so it has the lowest attention score.

Text	Language
u nä han i de schlüssel gnoo u bi dene entgäge glüffe ⟺ u nächer bin i han i e ⇆ was nä mit er ggangen isch weis i ou nid	BE ⇆ BS
ùnd ùnd äbe òü wider aine vo tachau ⟺ ùùnd ùnd de hend mini maischtersliit duezmaal de loon gad im vatter ggää ⇆ ùùnd den isch	LU ⇆ BS

(a) The attention visualization for the Swiss German dataset

Text	Languages
a de gränze gsii	ZH
a de gränze sige	BE
aber es isch denn scho bau aus	BE
aber es isch s isch	LU

(b) Error analysis for the Swiss German dataset

Figure 3: The left-hand side shows the model attention visualization process for dialect identification; the right-hand side displays error examples in dialect identification due to similar sentence constructions

There are possibilities of very small differences in sentence construction for DI or LI. Our model efficiently captures these differences. In Figure 3a the character n-gram “u nä” exists in BE. Even in word “nächer”, the first two letters are “nä”, which leads to the n-gram “u nä”. In case of BS, the construction is different and n-gram “u nä” does not exist. Thus in the case of BE, the highest attention weight is assigned to “u nä” which makes the sentence embedding more robust and efficient.

5.2 Error analysis in the model

The UDLDI works efficiently in comparison to other state-of-the-art methods. However, there are some challenges. As can be seen in Figure 3b, the sentence construction "*a de gränze*" can be found both in Zurich and Bern region. Though the sentences consist of other words, our methods are unable to differentiate between sentence-level features as the character n-grams are also common in different dialects; for example, the word "gsii" also exists in the Lucerne region. This leads to an insufficient attention weights distribution for common character n-grams. The same is true for the word sequence "*aber es isch*". These lead to biased clustering. This observation is supported by the NMI score of the clustering.

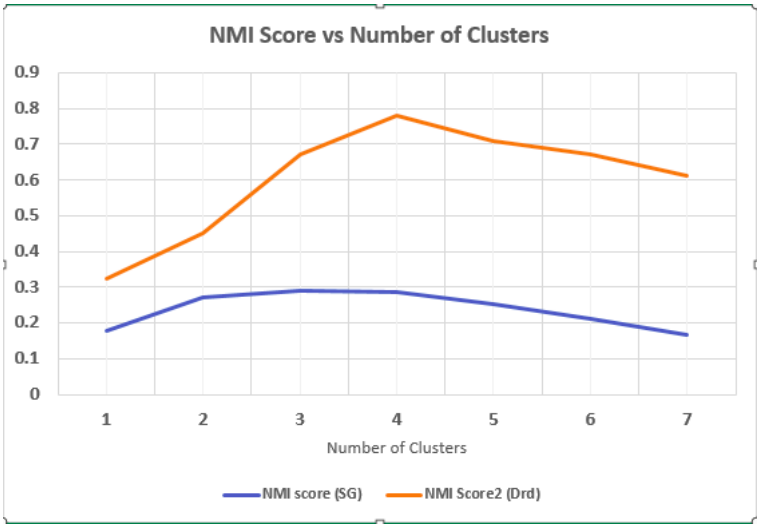

Figure 4: NMI Score vs Number of Clusters graph

Figure 4 shows that the NMI is highest when the number of clusters is 3 (blue line). We suspect that in the case of overlapping word sequences with a non-distinctive right context, the model is wrongly classifying sentences as belonging to the same cluster (dialect). We can compare this with Dravidian dataset. The graph shows that the NMI score is highest for 4 clusters (red line), which is equal to the number of languages present in the dataset. Though the Dravidian languages are very similar, they have different orthographies with a unique written script for each language. Thus the model can easily identify the script, directly captured by linguistic features on the character level. The somewhat unstable characterisation of the number of clusters as shown in the NMI graph for Swiss German dataset may to some extent be caused not only by linguistic features transcending one dialect area, but also, as mentioned in 5.1, by intra-speaker variation, transcriber-related variation and idiolect.

6 Conclusion

In this paper, we propose a novel character attention based unsupervised deep language and dialect identification (UDLDI) model for short texts of closely related languages. We have performed our experiments on three different language families and outperformed other state-of-the-art models both as a supervised and an unsupervised model. We have also achieved promising results for dialect identification which is considered to be one of the hardest tasks in NLP. Our experiments show that UDLDI is capable of doing language or even dialect identification irrespective of language families. The results in Section 5.1 show that the model is able to correctly identify some of the verbs which are distinguishable in different dialects. We have also shown that, based on NMI, our unsupervised model is correctly predicting the number of languages that are present in a dataset, opening up interesting applications to large-scale multilingual dataset processing.

Our future work will be the improvement of clustering efficiency for closely related languages and dialects. Though the model has outperformed other state-of-the-art models, there is a lot of scope to

improve the unsupervised learning model. In this model, for example, co-occurrence of words in different contexts has not yet been considered. In the future, we will aim to capture word contexts along with n-grams in sentence embeddings.

Acknowledgments

This publication has emanated from research in part supported by the Irish Research Council under grant number IRCLA/2017/129 (CARDAMOM-Comparative Deep Models of Language for Minority and Historical Languages). It is co-funded by Science Foundation Ireland (SFI) under Grant Number SFI/12/RC/2289_P2 (Insight_2) and Irish Research Council under project ID GOIPG/2019/3480. We would like to thank Ms. Omnia Zayed and Ms. Priya Rani for their valuable comments and suggestions towards improving our paper. We would also like to thank the anonymous reviewers for their insights on this work.

References

Abdelmalek Amine, Zakaria Elberrichi, and Michel Simonet. 2010. Automatic language identification: An alternative unsupervised approach using a new hybrid algorithm. *IJCSA*, 7(1):94–107.

Shane Bergsma, Paul McNamee, Mossaab Bagdouri, Clayton Fink, and Theresa Wilson. 2012. Language identification for creating language-specific twitter collections. In *Proceedings of the Second Workshop on Language in Social Media*, pages 65–74, Montréal, Canada, June. Association for Computational Linguistics.

Chris Biemann. 2006. Chinese whispers: an efficient graph clustering algorithm and its application to natural language processing problems. In *Proceedings of the first workshop on graph based methods for natural language processing*, pages 73–80. Association for Computational Linguistics.

Bharathi Raja Chakravarthi, Mihael Arcan, and John P. McCrae. 2018. Improving Wordnets for Under-Resourced Languages Using Machine Translation. In *Proceedings of the 9th Global WordNet Conference*. The Global WordNet Conference 2018 Committee.

Bharathi Raja Chakravarthi, Mihael Arcan, and John P. McCrae. 2019. Comparison of Different Orthographies for Machine Translation of Under-Resourced Dravidian Languages. In *2nd Conference on Language, Data and Knowledge (LDK 2019)*, volume 70 of *OpenAccess Series in Informatics (OASIcs)*, pages 6:1–6:14, Dagstuhl, Germany. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

Alina Maria Ciobanu, Shervin Malmasi, and Liviu P Dinu. 2018. German dialect identification using classifier ensembles. In *Proceedings of the Fifth Workshop on NLP for Similar Languages, Varieties and Dialects (VarDial 2018)*, pages 288–294.

Simon Clematide and Peter Makarov. 2017. CLUZH at VarDial GDI 2017: Testing a Variety of Machine Learning Tools for the Classification of Swiss German Dialects. In Preslav Nakov, Marcos Zampieri, Nikola Ljubesic, Jörg Tiedemann, Shervin Malmasi, and Ahmed Ali, editors, *Proceedings of the Fourth Workshop on NLP for Similar Languages, Varieties and Dialects, VarDial 2017, Valencia, Spain, April 3, 2017*, pages 170–177. Association for Computational Linguistics.

Eugen Dieth. 1986. *Schwyzertütschi Dialäktschrift*. Verlag Sauerländer, Aarau, 2 edition. Edited and published by Christian Schmid-Cadalbert.

B. Duvenhage, M. Ntini, and P. Ramonyai. 2017. Improved text language identification for the south african languages. In *2017 Pattern Recognition Association of South Africa and Robotics and Mechatronics (PRASA-RobMech)*, pages 214–218.

Heba Elfardy and Mona Diab. 2012. Token level identification of linguistic code switching. In *Proceedings of COLING 2012: Posters*, pages 287–296.

Oluwapelumi Giwa and Marelie H Davel. 2013. N-gram based language identification of individual words.

Harald Hammarström. 2007. A fine-grained model for language identification. In *Proceedings of iNEWS-07 Workshop at SIGIR 2007*, pages 14–20.

Nora Hollenstein and Noëmi Aepli. 2015. A resource for natural language processing of swiss german dialects.

Arthur S House and Edward P Neuburg. 1977. Toward automatic identification of the language of an utterance. I. Preliminary methodological considerations. *The Journal of the Acoustical Society of America*, 62(3):708–713.

Chu-Ren Huang and Lung-Hao Lee. 2008. Contrastive approach towards text source classification based on top-bag-of-word similarity. In *Proceedings of the 22nd pacific asia conference on language, information and computation*, pages 404–410.

Fei Huang. 2015. Improved arabic dialect classification with social media data. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2118–2126.

Tommi Sakari Jauhiainen, Marco Lui, Marcos Zampieri, Timothy Baldwin, and Krister Lindén. 2019. Automatic language identification in texts: A survey. *Journal of Artificial Intelligence Research*, 65:675–782.

Ben King and Steven Abney. 2013. Labeling the languages of words in mixed-language documents using weakly supervised methods. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1110–1119.

Ryan Kiros, Yukun Zhu, Russ R Salakhutdinov, Richard Zemel, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Skip-thought vectors. In *Advances in neural information processing systems*, pages 3294–3302.

Tom Kocmi and Ondřej Bojar. 2017. LanideNN: Multilingual language identification on character window. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 927–936, Valencia, Spain, April. Association for Computational Linguistics.

John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. 2001. Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data. In Carla E. Brodley and Andrea Pohoreckyj Danyluk, editors, *Proceedings of the Eighteenth International Conference on Machine Learning (ICML 2001), Williams College, Williamstown, MA, USA, June 28 - July 1, 2001*, pages 282–289. Morgan Kaufmann.

Zhouhan Lin, Minwei Feng, Cícero Nogueira dos Santos, Mo Yu, Bing Xiang, Bowen Zhou, and Yoshua Bengio. 2017. A structured self-attentive sentence embedding. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net.

Nikola Ljubešic and Denis Kranjcic. 2014. Discriminating between very similar languages among Twitter users. In *Proceedings of the Ninth Language Technologies Conference*, pages 90–94.

Nikola Ljubešić and Denis Kranjčić. 2015. Discriminating Between Closely Related Languages on Twitter. *Informatica*, 39(1).

Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-SNE. *Journal of machine learning research*, 9(Nov):2579–2605.

J MacQueen. 1967. Some methods for classification and analysis of multivariate observations. In *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Statistics*, pages 281–297, Berkeley, Calif. University of California Press.

Shervin Malmasi and Marcos Zampieri. 2017. German dialect identification in interview transcriptions. In *Proceedings of the Fourth Workshop on NLP for Similar Languages, Varieties and Dialects (VarDial)*, pages 164–169.

Maria Medvedeva, Martin Kroon, and Barbara Plank. 2017. When sparse traditional models outperform dense neural networks: the curious case of discriminating between similar languages. In *Proceedings of the Fourth Workshop on NLP for Similar Languages, Varieties and Dialects (VarDial)*, pages 156–163.

Kavi Narayana Murthy and G Bharadwaja Kumar. 2006. Language identification from small text samples. *Journal of Quantitative Linguistics*, 13(01):57–80.

Kamal Nigam and Rayid Ghani. 2000. Analyzing the effectiveness and applicability of co-training. In *Proceedings of the ninth international conference on Information and knowledge management*, pages 86–93.

Adam Poulston, Zeerak Waseem, and Mark Stevenson. 2017. Using TF-IDF n-gram and word embedding cluster ensembles for author profiling: Notebook for PAN at CLEF 2017. In *CEUR Workshop Proceedings*, volume 1866. CEUR.

Bali Ranaivo-Malançon. 2006. Automatic Identification of Close Languages-Case Study: Malay and Indonesian. *ECTI Transactions on Computer and Information Technology (ECTI-CIT)*, 2(2):126–134.

Shruti Rijhwani, Royal Sequiera, Monojit Choudhury, Kalika Bali, and Chandra Shekhar Maddila. 2017. Estimating code-switching on Twitter with a novel generalized word-level language detection technique. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1971–1982.

Tanja Samardžić, Yves Scherrer, and Elvira Glaser. 2016. ArchiMob - A Corpus of Spoken Swiss German. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, pages 4061–4066, Portorož, Slovenia, May. European Language Resources Association (ELRA).

Yves Scherrer, Tanja Samardžić, and Elvira Glaser. 2019. Digitising Swiss German: how to process and study a polycentric spoken language. *Language Resources and Evaluation*, 53(4):735–769.

Ali Selamat and Ng Choon Ching. 2008. Arabic script documents language identifications using fuzzy art. In *2008 Second Asia International Conference on Modelling & Simulation (AMS)*, pages 528–533. IEEE.

Karen Shiells and Peter Pham. 2010. Unsupervised clustering for language identification.

Vasiliki Simaki, Panagiotis Simakis, Carita Paradis, and Andreas Kerren. 2017. Identifying the authors' national variety of English in social media texts. In *Proceedings of the International Conference Recent Advances in Natural Language Processing, RANLP 2017*, pages 671–678, Varna, Bulgaria, September. INCOMA Ltd.

Jörg Tiedemann and Nikola Ljubešić. 2012. Efficient Discrimination Between Closely Related Languages. In *Proceedings of COLING 2012*, pages 2619–2634, Mumbai, India, December. The COLING 2012 Organizing Committee.

Dolf Trieschnigg, Djoerd Hiemstra, Mariët Theune, Franciska Jong, and Theo Meder. 2012. An exploration of language identification techniques in the dutch folktale database. In *Proceedings of the Workshop on Adaptation of Language Resources and Tools for Processing Cultural Heritage (LREC 2012)*.

Tommi Vatanen, Jaakko J. Väyrynen, and Sami Virpioja. 2010. Language identification of Short Text Segments with N-gram Models. In *Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC'10)*, Valletta, Malta, May. European Language Resources Association (ELRA).

Ada Wan. 2016. Leveraging data-driven methods in word-level language identification for a multilingual alpine heritage corpus. In *Proceedings of the Workshop on Multilingual and Cross-lingual Methods in NLP*, pages 45–54.

Junyuan Xie, Ross Girshick, and Ali Farhadi. 2016. Unsupervised deep embedding for clustering analysis. In *International conference on machine learning*, pages 478–487.

Omar F Zaidan and Chris Callison-Burch. 2014. Arabic dialect identification. *Computational Linguistics*, 40(1):171–202.

Marcos Zampieri and Binyam Gebrekidan Gebre. 2012. Automatic identification of language varieties: The case of Portuguese. In *KONVENS2012-The 11th Conference on Natural Language Processing*, pages 233–237. Österreichischen Gesellschaft für Artificial Intelligende (ÖGAI).

Marcos Zampieri and Binyam Gebrekidan Gebre. 2014. VarClass: An open-source language identification tool for language varieties. In *LREC 2014: 9th International Conference on Language Resources and Evaluation*, pages 3305–3308.

Marcos Zampieri, Liling Tan, Nikola Ljubešić, Jörg Tiedemann, and Preslav Nakov. 2015. Overview of the DSL shared task 2015. In *Proceedings of the Joint Workshop on Language Technology for Closely Related Languages, Varieties and Dialects*, pages 1–9.

Marcos Zampieri, Shervin Malmasi, Yves Scherrer, Tanja Samardzic, Francis Tyers, Miikka Silfverberg, Natalia Klyueva, Tung-Le Pan, Chu-Ren Huang, Radu Tudor Ionescu, et al. 2019. A report on the third Vardial evaluation campaign. In *Proceedings of the Sixth Workshop on NLP for Similar Languages, Varieties and Dialects*, pages 1–16.

Andjelka Zečević and Stasa Vujicic-Stankovic. 2013. The mysterious letter J. In *Proceedings of the Workshop on Adaptation of Language Resources and Tools for Closely Related Languages and Language Variants*, pages 40–44.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *Advances in neural information processing systems*, pages 649–657.

Wei Zhang, Robert AJ Clark, Yongyuan Wang, and Wen Li. 2016. Unsupervised language identification based on Latent Dirichlet Allocation. *Computer Speech & Language*, 39:47–66.