Discriminating Between Similar Nordic Languages

René Haas
IT University of Copenhagen
renha@itu.dk

Leon Derczynski
IT University of Copenhagen
leod@itu.dk

Abstract

Automatic language identification is a challenging problem. Discriminating between closely related languages is especially difficult. This paper presents a machine learning approach for automatic language identification for the Nordic languages, which often suffer miscategorisation by existing state-of-the-art tools. Concretely we will focus on discrimination between six Nordic languages: Danish, Swedish, Norwegian (Nynorsk), Norwegian (Bokmål), Faroese and Icelandic.

1 Introduction

Automatic language identification is a core problem in NLP but remains a difficult task (Caswell et al., 2020), especially across domains (Lui and Baldwin, 2012; Derczynski et al., 2013). Discriminating between closely related languages is often a particularly difficult subtask of this problem (Zampieri et al., 2014).

Language technology for Scandinavian languages is in a nascent phase (e.g. Kirkedal et al. (2019)). One problem is acquiring enough text with which to train e.g. large language models. Good quality language ID is critical to this data sourcing, though leading models often confuse similar Nordic languages.

This paper presents data and baselines for automatic language identification between six closely-related Nordic languages: Danish, Swedish, Norwegian (Nynorsk), Norwegian (Bokmål), Faroese and Icelandic.

Further, we investigate feature extraction methods for Nordic language identification and evaluates the performance of a selection of baseline models.

Finally, we test the models on a data set from a different domain in order to investigate how well the models generalize in distinguishing similar Nordic languages when classifying sentences across domains.

2 Related Work

The problem of discriminating between similar languages has been investigated in recent work (Goutte et al., 2016; Zampieri et al., 2015) which discuss the results from two editions of the "Discriminating between Similar Languages (DSL) shared task". Over the two editions of the DSL shared task different teams competed to develop the best machine learning algorithms to discriminate between the languages in a corpus consisting of 20K sentences in each of the languages: Bosnian, Croatian, Serbian, Indonesian, Malaysian, Czech, Slovak, Brazil Portuguese, European Portuguese, Argentine Spanish, Peninsular Spanish, Bulgarian and Macedonian.

Similar work has included (Toftrup et al., 2021), who include Nordic languages in a larger exercise in reproducing a commercial language ID system; and (Rangel et al., 2018), who attempt native language extraction, a task complex in the Nordic context which is rich in cognates and shared etymologies.

However, no prior work has focused specifically on the group of Nordic languages, leaving users of those languages without high quality automatically-extracted single language corpora (Derczynski et al., 2020). This is particularly disadvantageous for some Nordic language pairs, such as Danish/Norwegian and Faroese/Icelandic, where general-purpose many-language systems fall down (Toftrup et al., 2021). Thus, we focus specifically on data for this language and baseline methods.

3 The Nordic DSL data set

This section describes the construction of the Nordic DSL (Distinguishing Similar Lanugages)

Proceedings of the 8th VarDial Workshop on NLP for Similar Languages, Varieties and Dialects, pages 67–75

data set.

Data was scraped from Wikipedia. We downloaded summaries for randomly chosen Wikipedia articles in each of the languages, saved as raw text to six `.txt` files of about 10MB each. While Bornholmsk would be a welcome addition (Derczynski and Kjeldsen, 2019), exhibiting some similarity to Faroese and Danish, there is not yet enough digital text.

After the initial cleaning (described in the next section) the data set contained just over 50K sentences in each of the language categories. From this, two data sets with exactly 10K and 50K sentences respectively were drawn from the raw data set. In this way the data sets are stratified, containing the same number of sentences for each language.

We split these data sets, reserving 80% for the training set and 20% for the test set.

3.1 Data Cleaning and encoding.

This section describes how the data set is initially cleaned and how sentences are extracted from the raw data.

Extracting Sentences The first pass in sentence tokenisation is splitting by line breaks. We then extract shorter sentences with the sentence tokenizer (`sent_tokenize`) function from NLTK (Loper and Bird, 2002). This does a better job than just splitting by `'.'` due to the fact that abbreviations, which can appear in a legitimate sentence, typically include a period symbol.

Cleaning characters The initial data set has many characters that do not belong to the alphabets of the languages we work with. Often the Wikipedia pages for people or places contain names in foreign languages. For example a summary might contain Chinese or Russian characters which are not strong signals for the purpose of discriminating between the target languages.

Further, it can be that some characters in the target languages are mis-encoded. These mis-encodings are also not likely to be intrinsically strong or stable signals.

To simplify feature extraction, and to reduce the size of the vocabulary, the raw data is converted to lowercase and stripped of all characters which are not part of the standard alphabet of the six languages.

In this way we only accept the characters:

```
'abcdefghijklmnopqr
stuvwxyzáäåæéíðóöøúýþ '
```

and replace everything else with white space before continuing to extract the features. For example the raw sentence

```
'Hesbjerg er dannet ved
sammenlægning af de 2 gårde
Store Hesbjerg
og Lille Hesbjerg i 1822.'
```

will be reduced to

```
'hesbjerg er dannet ved
sammenlægning af de gårde store
hesbjerg og lille hesbjerg i ',
```

We thus make the assumption that capitalisation, numbers and characters outside this character set do not contribute much information relevant for language classification.

Feature encoding After the initial cleaning of the data we consider two methods for feature encoding: 1) Character level n-grams and 2) Skipgram and CBOW encodings created by training an unsupervised fasttext model on the dataset.

The Skipgram and CBOW methods encode sentences into fixed-length vectors by considering world level n-grams which are augmented with character level n-gram information (Bojanowski et al., 2016).

In the experiments presented in this paper, the CBOW and Skipgram encodings have the following settings: We use individual words (uni-grams) augmented with character level n-grams of size 2-5 with a context window of 5. The encoding result in fixed-length vectors in $\mathbb{R}^{100}$.

4 Baselines

4.1 langid.py

We compare with an off-the-shelf language identification system, `langid.py` (Lui and Baldwin, 2012). `langid.py` comes with a pretrained model which covers 97 languages. The data for langid.py comes from five different domains: government documents, software documentation, newswire, online encyclopedia and an internet crawl. Features are selected for cross-domain stability using the LD heuristic (Lui and Baldwin, 2011).

We evaluated how well langid.py performed on the Nordic DSL data set. It is a peculiar feature of

the Norwegian language that there exist two different written languages but three different language codes. Since langid.py also returned the language id “no" (Norwegian) on some of the data points we restrict langid.py to only be able to return either “nn" (Nynorsk) or “nb" (Bokmål) as predictions.

Figure 1: Confusion matrix with results from langid.py on the full Wikipedia data set

Figure 1 shows the confusion matrix for the langid.py classifier which achieved an accuracy of 78.3% on the data set. The largest confusions were between Danish and Bokmål, and between Faroese and Icelandic. `langid.py` was able to correctly classify most of the Danish instances; however, approximately a quarter of the instance in Bokmål were incorrectly classified as Danish and just under an eighth was misclassified as Nynorsk.

Furthermore, langid.py correctly classified most of the Icelandic data points; however, over half of the data points in Faroese were incorrectly classified as Icelandic.

4.2 Baseline with linear models

Table 1 shows results for running the models on a data set with 10K sentences in each language category. Models tend to perform better if we use character bi-grams instead of single characters. Logistic regression and SVM outperform Naive Bayes and K-nearest neighbors in all cases. Furthermore, for all models, we get the best performance if we use the skipgram model from FastText.

Comparing the CBOW mode from FastText with character bi-grams, the CBOW model is on par with bi-grams for the KNN and Naive Bayes classifiers, while bi-grams outperform CBOW for Logistic Regression and support vector machines.

Model	Encoding	Accuracy
Knn	cbow	0.780
Log-Reg	cbow	0.819
Naive Bayes	cbow	0.660
SVM	cbow	0.843
Knn	skipgram	0.918
Log-Reg	skipgram	**0.929**
Naive Bayes	skipgram	0.840
SVM	skipgram	**0.928**
Knn	char bi-gram	0.745
Log-Reg	char bi-gram	0.907
Naive Bayes	char bi-gram	0.653
SVM	char bi-gram	0.905
Knn	char uni-gram	0.620
Log-Reg	char uni-gram	0.755
Naive Bayes	char uni-gram	0.614
SVM	char uni-gram	0.707

Table 1: Overview of results for the data set with 10K data points in each language.

5 Our Approach

5.1 Using FastText

The methods described above are quite simple. We also compared the above method with FastText, which is a library for creating word embeddings developed by Facebook (Joulin et al., 2016).

Bojanowski et al. (2016) explain how FastText extracts feature vectors from raw text data. FastText makes word embeddings using one of two model architectures: continuous bag of words (CBOW) or the continuous skipgram model.

The skipgram and CBOW models are first proposed in (Mikolov et al., 2013) which is the paper introducing the word2vec model for word embeddings. FastText builds upon this work by proposing an extension to the skipgram model which takes into account sub-word information.

Both models use a neural network to learn word embedding from using a context windows consisting of the words surrounding the current target word. The CBOW architecture predicts the current word based on the context, and the skipgram predicts surrounding words given the current word (Mikolov et al., 2013).

5.2 Using A Convolutional Neural Network

While every layer in a classic multilayer perceptron is densely connected, such that each of the nodes in a layer are connected to all nodes in the next layer, in a convolutional neural network we use

one or more convolutional layers. Convolutional Neural Networks have an established use for text classification (Jacovi et al., 2018).

Our CNN is implemented with keras. We use an embedding layer followed by two blocks with a 1D convolutional layer and dropout. The first block has 128 filters while the second has 64. Both convolutional blocks use a kernel size of 5 and stride of 1. The two blocks are followed by a dense layer with 32 hidden nodes before the output layer which has 6 nodes. We use ReLu activation in the convolutional and fully connected layers and SoftMax in the output layer.

6 Results

6.1 Results with neural networks

Results for the neural network architectures are in Table 2. Here we compare the result of doing character level uni- and bi-grams using Multilayer Perceptron and Convolutional neural networks. The CNN performs the best, achieving an accuracy of 95.6% when using character bi-grams. Both models perform better using bi-grams than individual characters as features while the relative increase in performance is greater for the MLP model.

Model	Encoding	Accuracy
MLP	char bi-gram	0.898
CNN	char bi-gram	**0.956**
MLP	char uni-gram	0.697
CNN	char uni-gram	0.942

Table 2: Overview of results for the neural network models for the data set with 10K data points in each language.

6.2 Increasing the size of the data set

Often the performance of supervised classification models increases with more training data. To measure this effect we increase the amount of training data to 50K sentences in each of the language categories. Due to longer training times only the baseline models were included, with the skipgram encoding from FastText which we saw achieved the highest accuracy.

Table 3 shows that the performance of the logistic regression model and the K-nearest-neighbors algorithm improved slightly by including more data. Unexpectedly, performance of the support vector machine and Naïve Bayes dropped slightly with extra data.

Model	Encoding	Accuracy
Knn	skipgram	0.931
Logistic Regression	skipgram	**0.933**
Naive Bayes	skipgram	0.806
SVM	skipgram	0.925

Table 3: Overview of results for the "classical" ML models on the Wikipedia data set with 50K data points in each language.

Model	Encoding	Accuracy
MLP	char bi-gram	0.918
CNN	char bi-gram	**0.970**

Table 4: Overview of results for the MLP and CNN on the Wikipedia data set with 50K data points in each language.

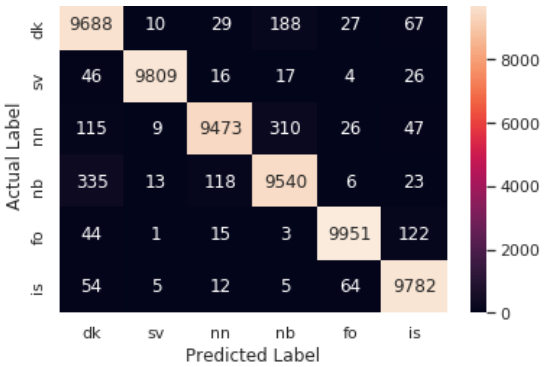


Figure 2: Confusion matrix with results from the CNN on the full Wikipedia data set.

Even when including five times the amount of data, the best result, logistic regression with an accuracy of 93.3%, is still worse than for the Convolutional Neural Network trained on 10K data points in each language.

Table 4 shows results for running the neural networks on the larger data set. Both models improve by increasing the amount of data and the Convolutional Neural Network reached an accuracy of 97% which is the best so far.

Figure 2 shows performance of the CNN trained on the Wikipedia data set with 50K data points per language. The model achieved an accuracy of 97% on the data set. The largest classification errors are between Danish, Bokmål and Nynorsk as well as between Icelandic and Faroese.

6.3 Using FastText supervised

FastText can also be used for supervised classification. In Joulin et al. (2016) the authors show that FastText can obtain performance on par with meth-

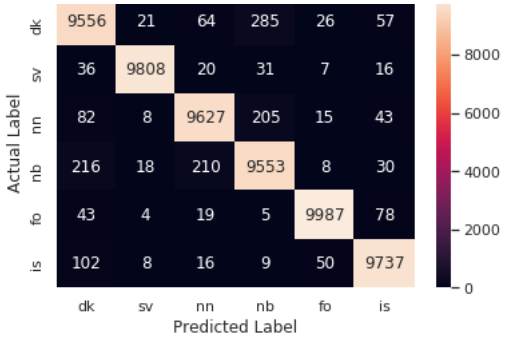


Figure 3: Confusion matrix with results from a supervised FastText model on the full Wikipedia data set.

ods inspired by deep learning, while being much faster on a selection of different tasks, e.g. tag prediction and sentiment analysis. We apply FastText classification to the Nordic DSL task. The confusion matrix from running the FastText supervised classifier can be seen in Figure 3. The supervised FastText model achieved an accuracy of 97.1% and thus the performance is similar to that of the CNN.

6.4 Cross-domain evaluation

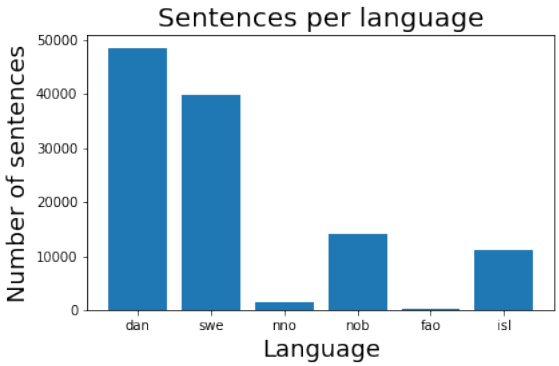
Training on single-domain data can lead to classifiers that only work well on a single domain. To see how the two best performing models generalize, we tested on a non-Wikipedia data set.

For this, we used Tatoeba,[1] a large database of user-provided sentences and translations.

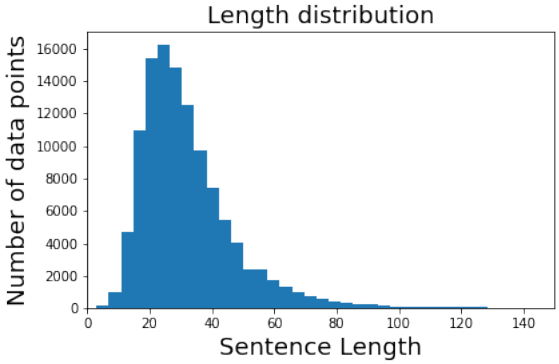
The language style used in the Tatoeba data set is different from the language used in Wikipedia. The Tatoeba data set mainly consists of sentences written in everyday language. Below are some examples from the Danish part of Tatoeba.

> Hvordan har du det? (How are you?)
>
> På trods af al sin rigdom og berømmelse, er han ulykkelig. (Despite all his riches and renown, he is unlucky.)
>
> Vi fløj over Atlanterhavet. (We flew over the Atlantic Ocean.)
>
> Jeg kan ikke lide æg. (I don't like eggs.)
>
> Folk som ikke synes at latin er det smukkeste sprog, har intet forstået. (People who don't think Latin is the most beautiful language have understood nothing.)

[1] tatoeba.org/



(a) Distribution of the number of sentences in each language in the Tatoeba data set.



(b) Distribution of the length of sentences in the Tatoeba data set.

Figure 4: Distribution of the lengths and language classes of Tatoeba sentences.

Figure 4a shows the number of sentences in each language in the Tatoeba data set.

Performance drops when shifting to Tatoeba conversations. For reference the accuracy of langid.py on this data set is 80.9% so FastText actually performs worse than the baseline with an accuracy of 75.5% while the CNN is better than the baseline with an accuracy of 83.8%.

One explanation for the drop in performance is that the sentences in the Tatoeba data are significantly shorter than the sentences in the Wikipedia data set as seen in Figure 4b. Both models tend to mis-classify shorter sentences more often than longer sentences. This and the fact that the text genre is different might explain why the models trained on the Wikipedia data set does not generalise to the Tatoeba data set without a drop on performance.

The CNN uses character bi-grams as features while, with the standard settings, FastText uses only individual words to train. The better performance of the CNN might indicate that character

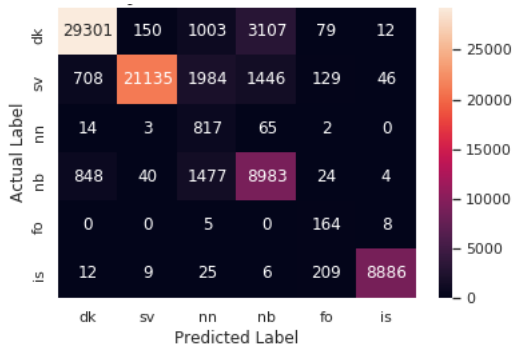


Figure 5: Confusion matrix for FastText trained using only character level n-grams on the Wikipedia data set and evaluated on the Tatoeba data set.

Figure 6: Results for FastText trained w. char n-grams on Wikipedia+Tatoeba and evaluated on Tatoeba.

level n-grams are more useful features for language identification than words alone.

To test this we changed the setting of FastText to train using only character level n-grams in the range 1-5 instead of individual words. Figure 5 shows the confusion matrix for this version of the FastText model. This version still achieved 97.8% on the Wikipedia test set while improving the accuracy on the Tatoeba data set from 75.4% to 85.8% which is a substantial increase.

Thus, using character-level features seems to improve the FastText models' ability to generalize to sentences belonging to a domain different from the one they have been trained on, supporting findings in prior work (Lui and Baldwin, 2011).

6.5 Retraining on the combined data set

To improve the accuracy over the Tatoeba data set, we retrained the FastText model on a combined data set consisting of data points from both Wikipedia and Tatoeba data.

The FastText model achieved an accuracy of 97.2% on this combined data set and an accuracy of 93.2% when evaluating this model on the Tatoeba test set alone - the confusion matrix is Figure 6.

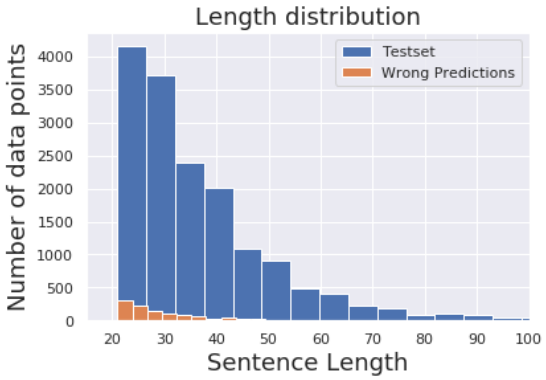


Figure 7: Distribution of sentence lengths Tatoeba test set along with the mis-classified sentences.

As was the case with the Wikipedia data set the mis-classified sentences tend to be shorter than the average sentence in the data set. Figure 7 shows the distribution of sentence lengths for the Tatoeba test set along with the mis-classified sentences. In the Tatoeba test set the mean length of sentences is 37.66 characters with a standard deviation of 17.91 while the mean length is only 29.70 characters for the mis-classified sentences with a standard deviation of 9.65. This again indicates that shorter sentences are harder to classify.

7 Analysis

7.1 Visualisation

To gain additional insight on how the different word embedding capture important information about each of the language classes, we visualized the embeddings using two different techniques for dimensionality reduction.

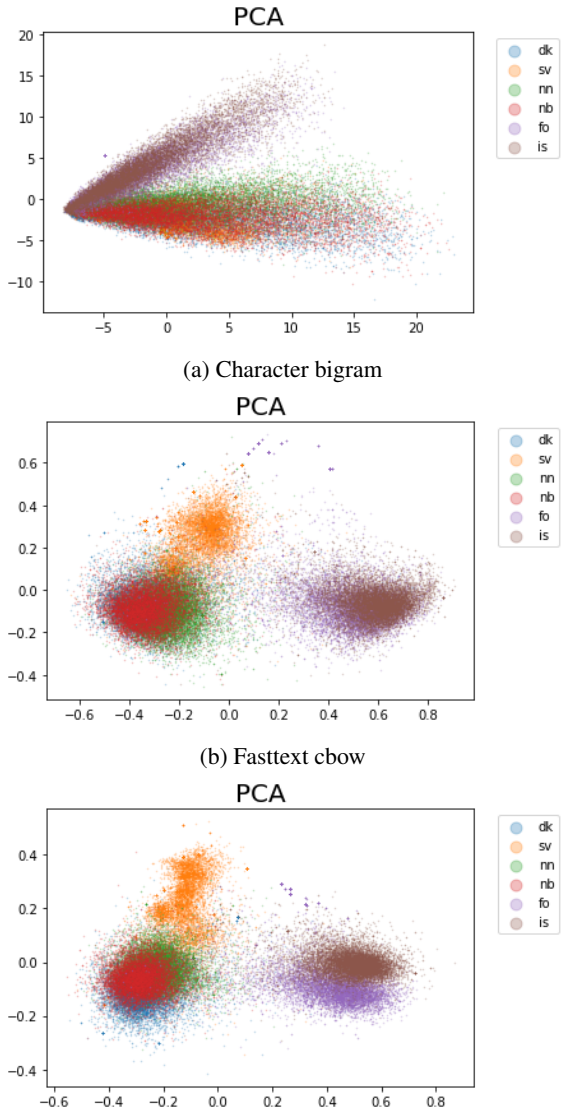
We used two different methods: Principal Component Analysis (PCA) and T-distributed Stochastic Neighbor Embedding (t-SNE). We begin with a brief explanation of the two techniques and proceed with an analysis of the results.

Principal Component Analysis The first step is to calculate the covariance matrix of the data set, with components:

$$K_{X_i,X_j} = E[(X_i - \mu_i)(X_j - \mu_j)] \quad (1)$$

where X_i is the ith component of the feature vector and μ_i is the mean of that component.

The next step is to calculate the eigenvectors and eigenvalues of the covariance matrix by solving the eigenvalue equation. The eigenvalues are the variances along the direction of the eigenvectors



(a) Character bigram

(b) Fasttext cbow

(c) Fasttext skipgram

Figure 8: Dimensionality reduction using PCA

or “Principal Components". To project our data set onto 2D space we select the two eigenvectors' largest associated eigenvalue and project our data set onto this subspace.

In Figure 8 we see the result of running PCA on the wikipedia data set where we have used character level bi-grams as features, as well as the CBOW and skipgram models from FastText.

In the figure for encoding with character level bi-grams, the PCA algorithm resulted in two elongated clusters. Without giving any prior information about the language of each sentences, PCA is apparently able to discriminate between Danish, Swedish, Nynorsk and Bokmål on one side, and Faroese and Icelandic on the other, since the majority of the sentences in each language belong to either of these two clusters.

With the FastText implementations we observe three clusters. For both CBOW and skipgram we see a distinct cluster of Swedish sentences. When comparing the two FastText models we see that the t-SNE algorithm with skipgrams seems to be able to separate Faroese and Icelandic data points to a high degree compared with the CBOW model. For the cluster of Danish, Bokmål, and Nynorsk sentences the skipgram models seem to give a better separation.

t-SNE The T-distributed Stochastic Neighbor Embedding method (van der Maaten and Hinton, 2008) favours retaining local relationships over remote ones.

In t-SNE, for a given data point x_i, the probability of picking another data point x_j as a neighbor to x_i is given by:

$$p_{ji} = \frac{\exp(||x_i - x_j||^2/2\sigma_i^2)}{\sum_{k \neq i} \exp(||x_i - x_k||^2/2\sigma_i^2)} \quad (2)$$

Given this probability distribution the goal is to find the low-dimensional mapping of the data points x_i which we denote y_i follow a similar distribution. To solve what is referred to as the “crowding problem", t-SNE uses the Student t-distribution which is given by:

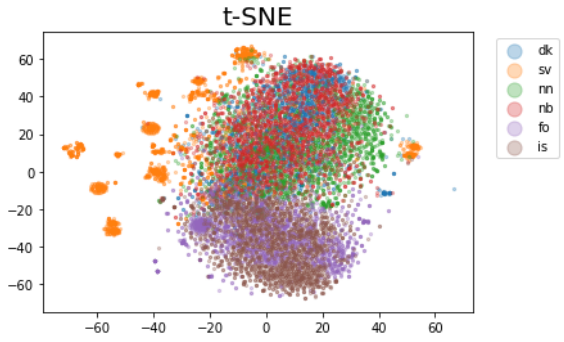
$$q_{ij} = \frac{(1 + ||y_i - y_j||^2)^{-1}}{\sum_{k \neq l}(1 + ||y_k - y_l||^2)^{-1}} \quad (3)$$

Optimization of this distribution is done using gradient decent on the Kullback-Leibler divergence which is given by:

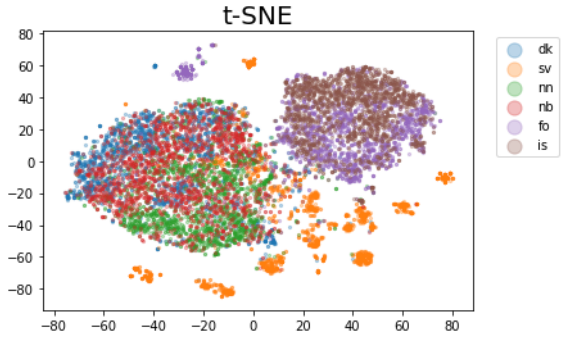
$$\frac{\delta C}{\delta y_i} = 4 \sum_j (p_{ij} - q_{ij})(y_i - y_j)(1 + ||y_i - y_j||^2)^{-1} \quad (4)$$

t-SNE results over the Wikipedia data sets can be seen in Figure 9. As was the case with PCA, it appears that the encoding with FastText seem to capture the most relevant information to discriminate between the languages; especially the skipgram mode does well at capturing information relevant to this task.

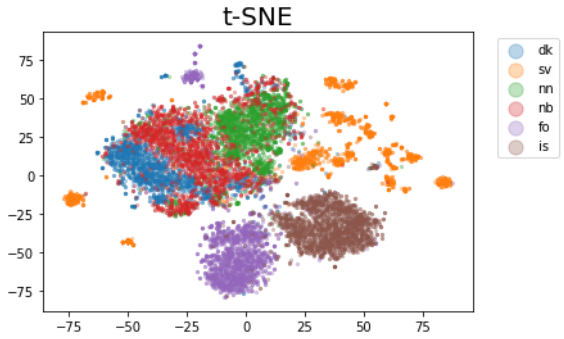
Here we recover some interesting information about the similarity of the languages. The data points in Bokmål lie between those in Danish and



(a) Character bi-gram



(b) FastText CBOW



(c) FastText skip-gram

Figure 9: Dimensionality reduction using t-SNE

Nynorsk, while Icelandic and Faroese have their own two separate clusters.

This fits speaker intuitions about these languages. Interestingly the Swedish data points and quite scattered, and t-SNE does not make a coherent Swedish cluster.

This does not however mean that the Swedish data points are not close in the original space. Some care is needed when interpreting the plot since t-SNE groups together data points such that neighboring points in the input space will tend to be neighbors in the low dimensional space.

7.2 Discussion

The dimensionality reduction techniques applied, PCA and t-SNE, were able to cluster the input sentences into three main language categories: (1) Danish-Nynorsk-Bokmål; (2) Faroese-Icelandic; (3) Swedish. Generally the supervised models made the most errors when discriminating between languages belonging to either of these language groups.

For the “classical" models, Logistic Regression and SVMs achieved better performance than KNN and Naive Bayes, where the latter performed worst. This was true in all cases irrespective of the method of feature extraction.

Additionally we saw that when we used feature vectors from the FastText skipgram model the classification models achieved better results than when using either FastText CBOW or character n-grams.

Generally we saw that increasing the number of data points lead to better performance. When comparing the CNN with the “classical" models however the CNN performed better than any of the other models even when trained on less data points. In this way it seems that the CNN achieves higher sample efficiency compared to the other models.

8 Conclusion

This paper presented dataset, baseline approaches, and analyses on automatically distinguishing similar Nordic languages. We visualized embeddings produced by character level bi-grams, CBOW and skipgram. We argue that, of these, FastText’s skipgram embeddings capture most information for discriminating between languages.

Data and code are available at https://github.com/renhaa/NordicDSL.

As baselines, we compared four different classical models: KNN, Logistic regression, Naive Bayes and a linear SVM with two neural network architectures: Multilayer perceptron and a convolutional neural network. The two best performing models, FastText supervised and CNN, saw reduced performance when going off-domain. Using character n-grams as features instead of words increased the performance for the FastText supervised classifier. Training on multiple domains resulted in an expected performance increase.

References

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2016. Enriching word vectors with subword information. *arXiv preprint arXiv:1607.04606*.

Isaac Caswell, Theresa Breiner, Daan van Esch, and Ankur Bapna. 2020. Language id in the wild: Unexpected challenges on the path to a thousand-language web text corpus. *arXiv preprint arXiv:2010.14571*.

Leon Derczynski, Rebekah Baglini, Morten H Christiansen, Manuel R Ciosici, Jacob Aarup Dalsgaard, Riccardo Fusaroli, Peter Juel Henrichsen, Rasmus Hvingelby, Andreas Kirkedal, Alex Speed Kjeldsen, et al. 2020. The Danish Gigaword Project. *arXiv preprint arXiv:2005.03521*.

Leon Derczynski and Alex Speed Kjeldsen. 2019. Bornholmsk natural language processing: Resources and tools. In *Proceedings of the 22nd Nordic Conference on Computational Linguistics*, pages 338–344.

Leon Derczynski, Diana Maynard, Niraj Aswani, and Kalina Bontcheva. 2013. Microblog-genre noise and impact on semantic annotation accuracy. In *Proceedings of the 24th ACM Conference on Hypertext and Social Media*, pages 21–30.

Cyril Goutte, Serge Léger, Shervin Malmasi, and Marcos Zampieri. 2016. Discriminating similar languages: Evaluations and explorations. In *Proceedings of Language Resources and Evaluation (LREC). Portoroz, Slovenia. pp 1800-1807 (2016).*

Alon Jacovi, Oren Sar Shalom, and Yoav Goldberg. 2018. Understanding convolutional neural networks for text classification. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 56–65, Brussels, Belgium. Association for Computational Linguistics.

Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2016. Bag of tricks for efficient text classification. *arXiv preprint arXiv:1607.01759*.

Andreas Kirkedal, Barbara Plank, Leon Derczynski, and Natalie Schluter. 2019. The lacunae of danish natural language processing. In *Proceedings of the 22nd Nordic Conference on Computational Linguistics*, pages 356–362.

Edward Loper and Steven Bird. 2002. Nltk: The natural language toolkit. In *In Proceedings of the ACL Workshop on Effective Tools and Methodologies for Teaching Natural Language Processing and Computational Linguistics. Philadelphia: Association for Computational Linguistics*.

Marco Lui and Timothy Baldwin. 2011. Cross-domain feature selection for language identification. In *Proceedings of 5th international joint conference on natural language processing*, pages 553–561.

Marco Lui and Timothy Baldwin. 2012. Langid.py: An off-the-shelf language identification tool. In *Proceedings of the ACL 2012 System Demonstrations*, ACL '12, pages 25–30, Stroudsburg, PA, USA. Association for Computational Linguistics.

Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-SNE. *Journal of Machine Learning Research*, 9:2579–2605.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Francisco Rangel, Paolo Rosso, Julian Brooke, and Alexandra L Uitdenbogerd. 2018. Cross-corpus native language identification via statistical embedding. In *Proceedings of the Second Workshop on Stylistic Variation*, pages 39–43.

Mads Toftrup, Søren Asger Sørensen, Manuel R. Ciosici, and Ira Assent. 2021. A reproduction of Apple's bi-directional LSTM models for language identification in short strings. In *Proceedings of the EACL Student Research Workshop*.

Marcos Zampieri, Liling Tan, Nikola Ljubesic, and Jörg Tiedemann. 2014. A report on the dsl shared task 2014. In *VarDial@COLING*.

Marcos Zampieri, Liling Tan, Nikola Ljubešic, Jörg Tiedemann, and Preslav Nakov. 2015. Overview of the dsl shared task 2015. In *Joint Workshop on Language Technology for Closely Related Languages, Varieties and Dialects*, page 1.