Cross-lingual sentiment classification using a biLSTM
Second Year Project (BSSEYEP1KU)
IT University of Copenhagen

First Author
Anna Mejlhede Jensen
anmj@itu.dk

Second Author
Christian Hjorth
chhj@itu.dk

Third Author
Eva Christelsdóttir
evac@itu.dk

Fourth Author
Julie Skoven Hinge
juhi@itu.dk

Abstract

This research paper examines the problem of labelling reviews with either a positive or negative sentiment, in a cross-lingual setting using a biLSTM model. Two methods were implemented to compare different approaches. A method using `Google Translate` to translate a target language to a source language, compared to using aligned word embeddings using `FastText`. As expected, our project shows that the method using `Google Translate` achieves higher evaluation scores in sentiment classification tasks than the alignment method. Our results can be reproduced by cloning our repository: https://github.itu.dk/juhi/nlp-project.

1 Introduction

Sentiment classification tasks have grown tremendously throughout the past years as state-of-the-art algorithms reaching high evaluation scores have been developed. These algorithms mainly focus on models trained on English data, and as a result, many resources are spent on training models in other languages. Hence, this paper will explore cross-lingual sentiment classification with English as a source language, so new models will not have to be trained every time a new language is introduced.

This study uses a bidirectional Long Short-Term Memory network (biLSTM), to predict the sentiment of Amazon product reviews. The aforementioned problem is tackled by adapting the resources available in one language to other languages. The transfer learning approach is applied by creating embeddings using aligned vectors from `FastText` and corpora in the chosen languages, English and German. These languages will then be passed to our biLSTM network to predict if a review has a positive or negative sentiment. Using this approach, we will examine the novel part of the research, which is cross-lingual sentiment classification with English as the source language and German as the target language. The main objectives are presented, including current resources in sentiment classification for the target language and exploration of the chosen methods. These methods consist of an alignment method and a method using `Google Translate`.

2 Related work

Several previous studies focus on cross-lingual sentiment classification using natural language processing (NLP). For example, one of the earlier studies, *"Cross-Language Text Classification Using Structural Correspondence Learning"* (Prettenhofer and Stein, 2010), builds on Structural Correspondence Learning (SCL) which is a method of building similar word embeddings across languages. Further, this study uses English as the source language and German, French, and Japanese as target languages.

Word embedding models are often used in sentiment classification tasks to learn word usages in various contexts. There are several different word embedding models, with Word2Vec being one of the most utilized. In a related paper, *"An LSTM Approach to Short Text Sentiment Classification with Word Embeddings"* (Wang, Liu, and Xiong Luo, 2018), the Word2Vec word embedding model is used to represent short sentences. Further, an LSTM model is trained to capture the long-term dependencies among the words. This is done with the aim of being able to classify the sentences as positive or negative. The research found that deep learning methods can learn the context of words in social media if given enough training data.

Our project builds on similar core ideas as the mentioned related work by doing cross-lingual sentiment classification with different methods as a

foundation for comparison and having English as the source language and German as the target language.

3 Data

This project works on data from the dataset *Amazon Multilingual Reviews*. This is a publicly available dataset gathered from the `datasets` library containing reviews across six languages - English, Japanese, German, French, Spanish and Chinese.

The Amazon dataset consists of 8 features:
`'review_id', 'product_id', 'reviewer_id', 'stars', 'review_body', 'review_title', 'language', 'product_category.'`

In total, the dataset contained 1.260.000 reviews prior to the pre-processing, with 2.5% set aside for validation and 2.5% for testing. This dataset seemed fitting due to its size, as well as it having a category with number of stars, which could be transformed into a sentiment. Additionally, this dataset was balanced, meaning it contained the same amount of positive and negative reviews.

Further, aligned vectors were downloaded from `FastText` for our source language, English, and target language, German, which was used for our embedding phase.

3.1 Data pre-processing

It was decided only to utilize the `'stars', 'review_body', 'language', 'product_category'` columns, as these contained information relevant for this project. The column containing the number of stars given to each review ranged from 1 to 5 stars. It was decided to binary encode these such that 0 would contain negative reviews and 1 would contain the positive reviews. Aiming to do so, all reviews with a rating of 3 stars were removed, as it was not possible to deem these either positive or negative. Then, reviews having a 1- or 2-star rating were encoded as having a negative sentiment (`sentiment:0`), whereas reviews having a 4- or 5-star rating were encoded as having a positive sentiment (`sentiment:1`).

This project works exclusively with reviews from the home category on Amazon. This was chosen since the many different categories seemed to differ quite a lot, presumably making it harder to then train a precise model. After this pre-processing was done, we were left with 14.064 English reviews in our test data, 493 English reviews in our validation data and 524 German reviews in our test data. Then, stop words were removed, and the (`review_body`) column was tokenized, making the data almost ready to be passed on to the embedding layer in our pipeline.

Before embedding our data, the reviews were given the same length. This was done by truncating sentences such that no reviews exceeded 128 words, which was arbitrarily set. Additionally, the reviews under 128 words were padded, so all reviews had a length of 128 words. It was decided to use pre-trained aligned vectors from `FastText`, which were loaded using the `Gensim` library. After creating the embeddings, they were converted into the format of a Torch Tensor to be used in our biLSTM model.

4 Experiments

4.1 biLSTM

As the main focus of this paper is cross-lingual sentiment classification, it was decided to use a biLSTM due to its ability to understand long-range temporal dependencies in word sequences. While other models predict classes purely based on statistics, the LSTM (Hochreiter and Schmidhuber, 1997) can further find actual meaning in text. This is done by using three gates to decide which information to store, which is important, and which must be passed into the next hidden state - *Forget Gate*, *Input Gate* and *Output Gate*. Since human language is complex and nuanced, it was decided to add a bidirectional layer to the LSTM. This layer reverses the direction of information flow, meaning that the outputs from both layers can then be combined, e.g., by concatenating, as in our case, resulting in a more precise model.

The biLSTM was used to extract global features related to the text context and, furthermore, to finally predict the sentiment. BiLSTM is a sequence processing model that gives sequence information to the network in two directions (Graves, Fernández, and Schmidhuber, 2005). The biLSTM takes the input in a forward and backward direction. This way increases the available contextual information and, thus, there is more context for the algorithm. These aspects of the algorithm make it useful for sentiment classification tasks. Furthermore, biLSTMs are powerful at learning from data with long-range temporal dependencies and

have proved to be very efficient at modelling word sequences.

An illustration of our biLSTM model is shown in figure 1. Our model was created using `PyTorch` by implementing a class. It takes input in the form of torch embeddings mentioned in the previous section. Our model consists of two linear layers, two activation functions, namely ReLU (Rectified Linear Unit) and Sigmoid. Further, a dropout of 0.2 is applied to each time step to prevent overfitting. In addition, the parameter `bidirectional` was set to true to make the model bidirectional.

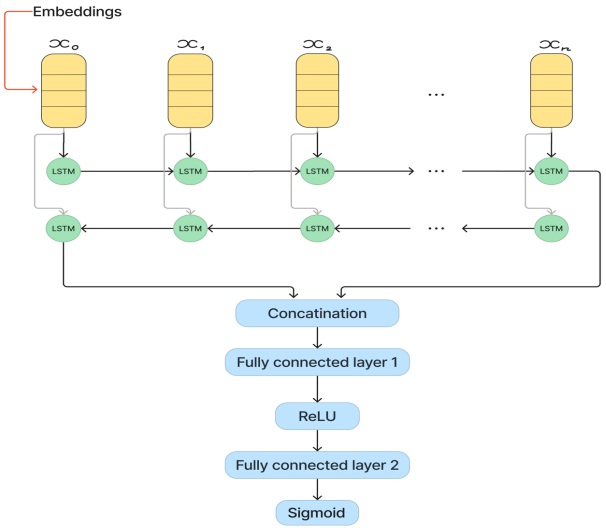

Figure 1: Drawing illustrating our biLSTM model

To find the optimal parameters, different parameters were checked on the validation data and compared with multiple different combinations. Further, different optimizers were attempted and implemented to find the combination yielding the best results. It was found that the optimizer Adam with a learning rate of 0.003, using a Binary Cross-Entropy (BCE) loss function, gave the best results. A Stochastic Gradient Decent (SGD) optimizer with Mean Squared Error loss was also tried, but this combination gave worse results.

Parameters		
Batch size	*Epochs*	*Learning rate*
25	10	0.003
Optimizer	*Loss function*	
Adam	BCE	

Figure 2 shows the training and validation loss. The validation loss indicates that the model will generalize well to new data, as it behaves similarly to the training loss. The number of epochs was set relatively low, as our dataset was extensive, and, thus, iterating through it many times would not make sense. Another reason for the low epochs was to avoid overfitting so our model would generalize better. Finally, due to the amount of data, the batch size was set to 25, as using a higher number degraded the quality of the model.

Figure 2: Training vs. validation loss

4.2 Two methods: FastText and Google Translate

The goal of this project was to implement and compare two different approaches to cross-lingual sentiment classification. As the image below illustrates, English training data was first used to train the model. Secondly, English validation data was used to find optimal hyper-parameters for the model.

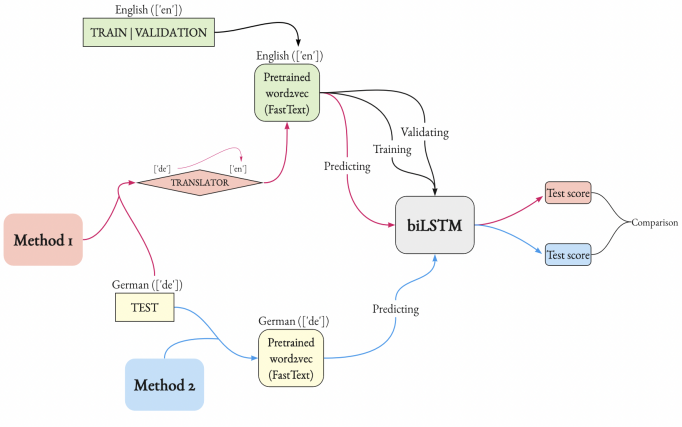

Figure 3: Illustration of our methodology

4.2.1 Method 1

The objective of the first method was to evaluate the model on translated German reviews. To do

this, the `Translators` library in Python was imported, which uses Google Translate to translate German (target language) reviews into English (source language). They were then tokenized and embedded using Word2Vec in English. Finally, these embeddings were transformed into a Torch Tensor, which was passed to be tested on our English trained biLSTM model. This approach was expected to perform better, using languages with good resources available for translation or closely related languages (e.g., German would perform better compared to Japanese). These languages have similar grammatical structures of sentences, thereby obtaining potentially better translation.

4.2.2 Method 2

The second method utilizes aligned word vectors from the `FastText` library. Here, the embeddings were made using German reviews that had *not* been translated first. The created embeddings from the German reviews were then fed to the biLSTM model to predict. `FastText` uses aligned word embeddings for different languages and, thus, this method was expected to work reasonably well since words with a similar meaning in English and German would have similar word embeddings. However, in the example below, it can be observed that even though words have the same meaning in two languages, their embeddings do not achieve a high cosine similarity. Hence, the method is not presumed to yield perfect results.

```
print(FastVector.cosine_similarity(fr_dictionary["chat"], ru_dictionary["кот"]))
# Result should be 0.43
```

Figure 4: Cosine similarity between two words

5 Results

The results obtained from the two methods can be seen in the table containing our test scores. Method 1 proved to perform better than method 2. This was expected as the model was trained on English data, so feeding the model English word vectors would outperform feeding it aligned German word vectors. Additionally, a baseline made up of a logistic regression model was included to have more results to compare against. The data was fed to our baseline, giving an accuracy score of 50% on our German test data. This result was expected as the model was trained on English data, and because no method of aligning the languages was attempted.

Test Scores		
Method	*F1*	*Acc*
Method 1 (translator)	79%	80%
Method 2	48%	64%
Baseline	-	50%

6 Analysis

As described in the Results section, it was not possible to achieve perfect results. This could be due to many reasons, such as the nature of our data or the alignment methods used. In general, there is no certainty that the reviews classified as positive contain exclusively positive language. A review like *"This shit used to be so bad, but has gotten tolerable now"* with 4 stars would be encoded as being positive. However, this contains more negative words than positive, and our model would likely classify it as negative. There are likely many more instances of such reviews being classified as negative by our model when they, in reality, should be positive and vice versa.

As mentioned, method 1 using translated data performed better than method 2 using aligned word vectors. This could be due to the fact that in method 2, the word vectors were not completely similar, as seen in figure 4. Furthermore, our model could potentially achieve better results if an alignment method having a higher cosine similarity between words was used. Another reason why method 1 outperformed method 2 could be attributed to `Google Translate` being very efficient at translating languages, as it can capture the context very well.

6.1 biLSTM

Implementing a good model that understands context and language is essential in NLP tasks like sentiment classification. Our biLSTM model's structure allowed it to learn long, contextual dependencies. A disadvantage with using LSTMs is their tendency of overfitting to their training data. In order to prevent overfitting, we experimented with using weight decay in order to regularize our data. Unfortunately, this yielded worse results than without weight decay, and thus we decided to remove this parameter.

6.2 Method 1 and 2

Method 1 gave an accuracy score of 80% and an F1-score of 79%. This is better than the accuracy score of 64% and F1-score of 48%, which method

2 yielded. This was expected since method 1 theoretically performs well for similar languages. However, it has some disadvantages, e.g., it cannot be used on languages like Faroese that are not supported by `Google Translate`. Additionally, distant languages such as Japanese and Chinese might give more grammatical errors when translated to English, giving a worse performance using method 1, as context and nuances might get lost in translation. Since our biLSTM was trained on English data, it was anticipated that method 2, having German data, would perform worse than the first method, and this is proven through our study.

6.3 Limitations

Overall, due to time constraints and limited computing power, it was only possible to a limited extent to experiment with different model structures of the biLSTM or overall combinations of different model types.

7 Conclusion and Future Work

The performance of this study was better than our baseline, as discussed in a previous section. In addition, our study has proven that the method using German data translated to English works better than using German data on our English trained biLSTM model. By this means our results are in accordance with our expectations. In order to further research the modelling of cross-lingual classification tasks, it could prove fruitful to experiment with Convolutional Neural Networks (CNNs). Due to the structure of CNNs, they prove to be great at capturing local dependencies, which could improve the overall performance of our model (Kim, 2014). Furthermore, papers and experiments found online with similar classification tasks, show that such a CNN-LSTM combination will in many cases give great outcomes compared to experiments using a single model.

Additionally, it would be interesting to have more target languages, as some of the related work did. This would give a more in-dept analysis and present more accurate performance of our methods. Here it would be relevant to experiment with low-resource languages, as for this project it was not possible due to time constrains to do so. It was considered to use Japanese as a possible target language, due to it having enough resources to perform high quality experiments, whilst at the same time not being related to the West Germanic Languages, as opposed to English and German.

References

Graves, Alex, Santiago Fernández, and Jürgen Schmidhuber. "Bidirectional LSTM Networks for Improved Phoneme Classification and Recognition". In: 1 (2005), pp. 799–804. URL: https://mediatum.ub.tum.de/doc/1290195/file.pdf.

Hochreiter, Sepp and Jürgen Schmidhuber. "Long Short-term Memory". In: *Neural computation* 9 (Dec. 1997), pp. 1735–80. DOI: 10.1162/neco.1997.9.8.1735. URL: https://www.researchgate.net/publication/13853244_Long_Short-term_Memory.

Kim, Yoon. "Convolutional Neural Networks for Sentence Classification". In: (Aug. 2014). URL: https://arxiv.org/abs/1408.5882.

Prettenhofer, Peter and Benno Stein. "Cross-Language Text Classification using Structural Correspondence Learning". In: *48th Annual Meeting of the Association of Computational Linguistics (ACL 2010)*. Association for Computational Linguistics, July 2010, pp. 1118–1127. URL: https://webis.de/downloads/publications/papers/prettenhofer_2010.pdf.

Wang, Jenq-Haur, Ting-Wei Liu, and Long Wang Xiong Luo. "An LSTM Approach to Short Text Sentiment Classification with Word Embeddings". In: (2018), pp. 214–223. URL: https://aclanthology.org/O18-1021.pdf.

Appendix

Group contribution

As a group, we contributed equally in the project. However, our git log does not reflect this precisely due to our usage of a real time collaborative platform.

Phase 2: Difficult cases

Our baseline did not work well in the break-it phase with the difficult cases and yielded bad scores. Due to the difficult cases being in English, we decided to not test our model on it, since it is irrelevant as our project focuses on cross-lingual sentiment classification.