



Multilingual Sentiment Analysis: A Deep Learning Approach

Saad Mboutayeb¹, Aicha Majda¹^a and Nikola S. Nikolov²^b

¹Laboratory of Intelligent Systems and Applications, University of Sidi Mohamed Ben Abdellah, Fez, Morocco

²Department of CSIS, University of Limerick, Limerick, Ireland

Keywords: Multilingual Sentiment Analysis, Deep Learning, Social Media, Language.

Abstract: Most user-generated text on social media is not in English but other languages such as Arabic, French, and Portuguese. This makes the text analysis tasks more difficult, especially sentiment analysis, because of its high dependence on the language. On the other hand, building a model for each language is time and resources consuming. In particular, there is a lack of linguistic resources such as datasets. In this paper, we examine if a sentiment analysis model trained on one language can correctly predict the sentiment of text originally written in another language and translated into the model's language. We present experimental results of training CNN, RNN and combined CNN-RNN models on a dataset of multilingual tweets. Our findings suggest that CNN gives the best results with an accuracy of 85.91% and an F1-score of 84.61%. Our best model also achieved high accuracy on unseen tweets in European languages different from the original languages of the tweets used for training.

1 INTRODUCTION


As opinions or sentiments are vital influencers of human behavior, they have become central to our daily activities. Before choosing, most of us routinely inspect online reviews of products and services by previous consumers on social media. This is true not only for individuals but also for businesses that utilize public sentiment about their products to have a comprehensive view of their performance. These tasks can be facilitated by *Sentiment Analysis* (also called *Opinion Mining*), which refers to the computational study of people's sentiments toward entities, individuals, issues, and topics (Liu & Zhang, 2012).


According to Statista (Statista, 2019), as of April 2019, English is the most common language on the Internet, with 25.2% of the Internet users. However, the other languages account for 74.8% of global Internet users, indicating that most user-generated text data on social media is not in English but in other languages such as Chinese, Spanish, Arabic, and Portuguese.

Given this fact, it has become necessary to create sentiment analysis models for each language used on social media. However, this poses a significant challenge since sentiment analysis depends on the language. Pre-processing techniques such as removing stop words and stemming, as well as vectorization methods, are language-specific. On the other hand, training a sentiment analysis model for each language can be too time and resource consuming.

In this work, we explore the possibility of building a multilingual sentiment analysis model by training it on a set of English tweets only. This model predicts sentiment in tweets written in languages other than English by translating them to English first before being passed as input to the model.

We compare the performance of a few deep learning models: Convolutional Neural Networks (CNN), Recurrent Neural Networks (RNN) with its two types: Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU), as well as the combination between CNN and LSTM and the combination between CNN and GRU.

^a <https://orcid.org/0000-0001-6419-3639>

^b <https://orcid.org/0000-0001-8022-0297>

2 RELATED WORKS

Most of the research in sentiment analysis has been in the context of a single language, in most cases English (Ravi & Ravi, 2015). In contrast, the number of publications on multilingual sentiment analysis does not exceed ten publications annually (Zhao et al., 2016). Nonetheless, most of the user-generated text on social is in languages different from English. Limiting the analysis of sentiment to English only leads to a significant loss of information (Zhao et al., 2016).

The principal challenge in sentiment analysis is that it strongly depends on the language. Language-specific resources for sentiment analysis, such as sentiment dictionaries and labeled data, are scarce, especially in languages different from English. The studies in multilingual sentiment analysis focus on using the available resources and tools in one language, such as lexicons (Taboada et al., 2011) or machine translation (Wu et al., 2016), to build sentiment classifiers in other languages with few resources. Reportedly, there are three approaches to overcome the unavailability of adequate resources for sentiment analysis in a language different from English.

First of all, documents written in other languages can be translated into English, and an English sentiment classifier determines their sentiment. Kim and Hovy (Kim & Hovy, 2006) translated German emails to English and applied an English lexicon to determine opinions expressed in these emails. In another study, Bautin et al. (Bautin et al., 2008) experimented with translating a corpus of documents written in eight languages into English and then used an English lexicon to determine sentiment.

Another approach is translating an English corpus into a target language(s) and training a model on the translated corpus. This is the approach taken by Banea et al. (Banea et al., 2010), who translated a labeled English corpus into five other languages and combined the translated versions with the original English version to create a single training corpus for a machine learning classifier.

A third approach is to use machine translation for translating an English sentiment lexicon into another (target) language and then utilize it for the lexicon-based classifier in the target language. This is the approach taken by Kim and Hovy (Kim & Hovy, 2006) in their second experiment. They translated an English lexicon into German and then used it to analyze German emails.

3 METHOD AND MATERIALS

In this section, we outline the methodology for multilingual sentiment analysis followed in our work. We describe each step of the workflow in detail.

Our entire workflow is presented in Fig.1. We created a multilingual corpus by combining three existing datasets of English, Arabic, and Portuguese tweets, respectively. After detecting the language of a tweet, it is translated using the Google Translate API if not in English and passed to the pre-processing module. After pre-processing, the cleaned data is passed to the word embedding module to represent each word in a vector form reflecting its meaning.

Then multiple classification models are tuned and trained on the training subset (80% of the dataset) and compared on the testing subset (20% of the dataset). Finally, the winning model is tested on additional tweets written in 5 different languages in order to give us an indication of the power of our model.

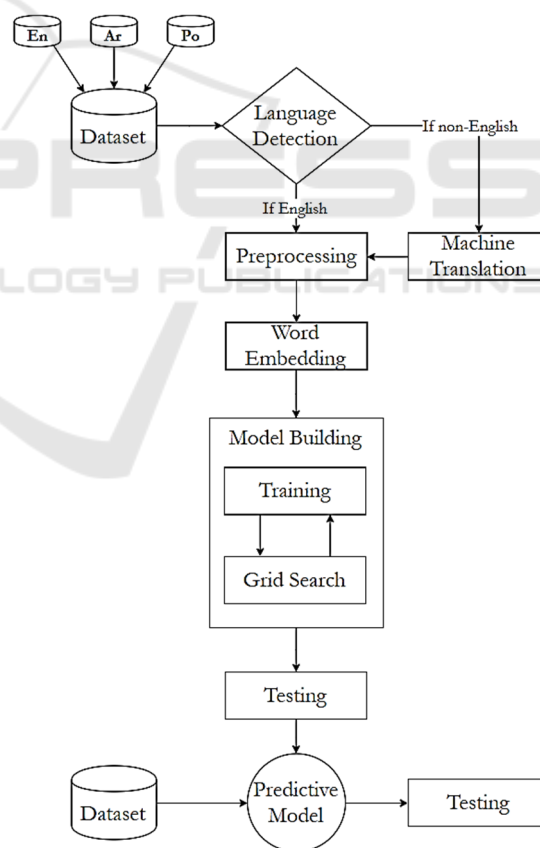


Figure 1: The proposed methodology for multilingual sentiment analysis.

3.1 Data Sources

In this work, we used three datasets of three different languages: English (Go et al., 2009), Arabic (Saad, 2020) and Portuguese (Portuguese Tweets for Sentiment Analysis, 2019). We selected 6,000 tweets (3,000 positive tweets and 3,000 negative tweets) from each dataset to build a balanced dataset dedicated to our project.

3.2 Data Preparation

3.2.1 Translation

We automatically translated the Arabic and Portuguese tweets into English. To qualify for being used in our project, a machine translation software must not translate text word by word but instead based on rules that capture the meaning of the text. The translator must also automate the translation from multiple languages into English. These conditions are met by the Google Translate API, which as of 2021, translates texts instantly from and into more than 100 languages (Google).

3.2.2 Pre-processing

We clean our dataset from useless words such as URLs, usernames, and stopwords. We also filter out the special characters (e.g., # and punctuations).

A negation word can influence the meaning of all words around it. Since ignoring negations may lead to misclassification, we replace all negation words (don't, can't, isn't, etc.) with "not". This allows taking into consideration the existence of negation in a tweet.

Some emojis are reliable indicators of sentiment polarity. For instance, some words have no sentiment value. However, if an emoji is used along with these words, the tweet may have a sentiment value. For this reason, emojis should not be ignored and removed from the corpus. We replace each emoji with an alias.

In order to reduce the sparsity and the vocabulary size, we applied two simple operations. The first is to lowercase each word in our corpus, and the second aims to remove the characters repeated in a word at least three times. For example, "gooood" is transformed into "good".

The last step of pre-processing is stemming, which reduces the morphological variations of words by reducing them to a common root (also called stem). For instance, a stemmer would reduce the words "saddest", "sadness", "sadly" to the root "sad". In our case, we used the Porter stemmer (Porter, 1980), which is widely used for its simplicity and

speed. The maximal length of a tweet is 58 tokens. The total number of words in the entire dataset is 181,558, while the vocabulary size is 20,084.

3.2.3 Word Embedding

Word embedding is a word vectorization process in which words are represented by vectors of real numbers. These vectors are constructed to reflect the meanings of the words. For instance, in perfect embedding space, synonyms must be represented by similar word vectors.

Word2vec (Mikolov et al., 2013) is one of the most commonly used techniques for word embedding. It is a shallow neural network that consists of one hidden layer. The trained weight matrix of the hidden layer contains the word vectors. The training can be done with one of two models: CBOW and Skip-gram. CBOW predicts a word based on its surroundings (context words), while Skip-gram predicts the context for a word (Mikolov et al., 2013). We choose to use the CBOW model because it is faster and works well with massive datasets that contain frequent words, unlike Skip-gram, which works well with relatively smaller datasets (Mikolov et al., 2013).

As the maximum length of a tweet is 58 tokens, each tweet that contains fewer than 58 tokens is padded with zeros. We set the vector dimensionality at 300 and the context window at five (the number of context words for every word is four).

3.3 Models

We optimized each model's hyperparameters using grid search guided by the model's accuracy as a metric. The accuracy is calculated by performing 5-fold cross-validation on the training set (80% of our dataset). Each fold contains the same distribution of sentiment labels and the original language of tweets. The tuned hyperparameters are Epoch, Batch size, Pooling size, Dropout rates, Output dense size.

Each of the models is considered as a set of layers. The first layer is the embedding layer, which embeds words into dense vectors. It takes the padded sequences as input and turns each integer of the sequence into its corresponding dense vectors (embedding). This process is based on a lookup matrix pre-trained using Word2Vec's CBOW model. This layer's output is a matrix of size 58x300, in which each row represents a word vector.

3.3.1 Convolutional Neural Network

Convolutional Neural Network (CNN) is a particular type of neural network used mostly in computer

vision and image processing. However, it could be employed in text classification tasks as a corpus can be represented as a two-dimensional matrix (grid) consisting of word embeddings.

As the convolutional layer, the pooling layer drags a pooling window over feature maps. Each map would be reduced in terms of size to keep just the dominant features. Like other studies (Shin et al., 2016; Wang et al., 2016; Y.Kim, 2014), we perform max pooling on the maps with a size set at 4. Then, we merge the two branches by concatenating the pooled maps. The generated shape is a unique vector, which is considered the final feature vector. This process is illustrated in Fig. 2.

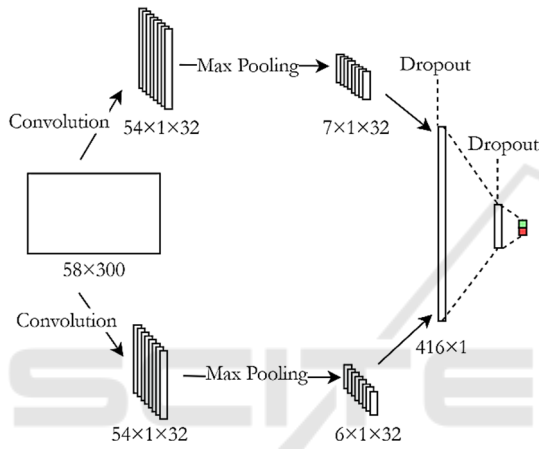


Figure 2: CNN model architecture.

In our CNN architecture, a fully connected (FC) layer is defined as two sub-layers. The first one is a densely connected layer of neurons that implements Sigmoid as an activation function. It takes the flattened vector as input, and its output is a vector of 64 values. Similarly, the second sub-layer performs the softmax activation function and returns a vector of two values (positive or negative).

A dropout regularization (Srivastava et al., 2014) is adopted to prevent our network from overfitting. Its idea is to ignore some units randomly (input units). We dropped out 40% of the unit of each FC sub-layer.

3.3.2 Recurrent Neural Networks

We used two RNN models that consist of the following scheme: embedding layer, bidirectional-RNN layer, and FC layers. The difference between the two models is only in the type of cells used in the bidirectional-RNN layer. The architectures of the two models are shown in Fig. 3.

Both models are bidirectional RNNs, each consisting of two independent RNNs with the same

number of cells (58 cells). The first RNN takes the tweet in regular order, while for the other RNN, the feeding of the tweet is done in reverse order. Since each cell's output is one value, the bidirectional RNN (either bi-LSTM or bi-GRU) returns a vector of 116 values. Each pair of values corresponds to one word of the tweet.

In the same way, as in the CNN model, a feedforward neural network takes the vector generated by the bidirectional layer as an input, but 40% of its values are dropped out. The first part of this network implements sigmoid as an activation function and outputs a vector of 16 values; 20% of its values are dropped out. This vector is passed to the second sub-layer, whose activation function is softmax, and its output is a vector of two values. Each one represents the probability that a tweet belongs to a class.

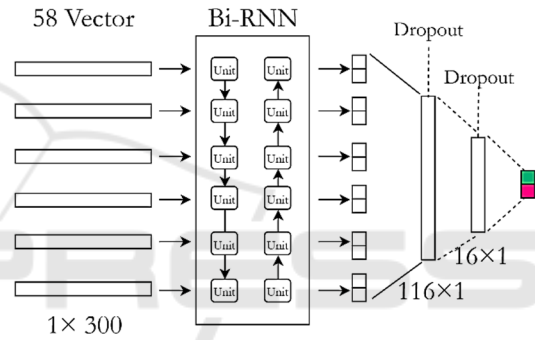


Figure 3: Bi-RNNs architecture.

3.3.3 RNN-CNN Networks

In this section, we present two models with architecture that combines elements of CNN and RNN.

The first one combines CNN and bi-LSTM, and the second combines CNN and bi-GRU. Both models are organized according to a layered structure consisting of four layers: embedding layer, convolutional layers, bi-RNN layers, and FC layer. The difference between them is only in the bi-RNN layer, where the CNN-LSTM model, as its name indicates, uses the LSTM cells, while the CNN-GRU depends on the GRU cells.

As illustrated in Fig. 4, these combined models consist of three branches with identical processes but different parameters. For example, the first branch begins with a convolutional layer, in which 32 kernels of size 3 are applied to the input matrix to generate 32 feature maps. Each feature map contains 56 local features. The maps are stacked as a matrix, the rows of which are taken as inputs to the bidirectional RNN

(LSTM or GRU), which generates a vector of 112 values.

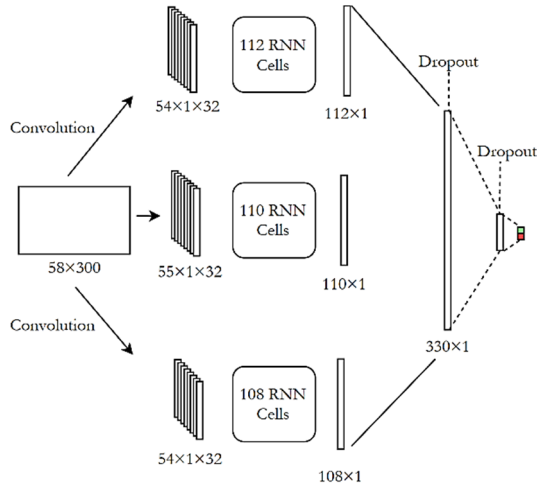


Figure 4: RNN-CNN models architecture.

In the end, all the branches are merged by concatenating their output vectors to form the input to the FC layers, which are not different from those in the CNN model.

4 EVALUATIONS

In this section, we present the results of the models' evaluation. Firstly, we trained our models on the training subset (80% of the dataset) and evaluated them on the testing subset (20% of the dataset).

4.1 Results and Discussion

We describe the result of our models' testing in terms of accuracy, F1-score, and AUC in Table 1.

Table 1: Accuracy, F1-score, and AUC of models.

Model	Accuracy	F1-score	AUC
CNN	85.91	84.61	94.33
Bi-LSTM	84.35	84.18	92.18
Bi-GRU	83.78	83.31	92.01
CNN-LSTM	84.92	84.57	93.54
CNN-GRU	84.52	84.50	93.35

The CNN model outperforms the other models and achieves the highest accuracy of 85.91%, the best F1-score (85.08%), and the largest AUC (94.33%).

The Bi-LSTM model achieves lower results than CNN-LSTM. The accuracy of Bi-LSTM is 0.56% lower than the accuracy of CNN-LSTM, and its F1-score is 0.45% lower than the F1-score of CNN-LSTM. Combining CNN and the Bi-GRU gives

accuracy and F1-score higher than those of the Bi-GRU model.

The Bi-LSTM model shows results that are superior to those of the Bi-GRU model. Similarly, the CNN-LSTM model achieves results that are better than the results of the CNN-GRU model.

In summary, the comparison of these models leads us to two main observations: (i) Models containing convolutional layers outperform models without convolutional layers. (ii) LSTM cells in RNN layers lead to better results than GRU cells in RNN layers.

4.2 Evaluation with New Data

Finally, we tested our best-performing CNN model on an independent set of tweets written in different languages: English, Arabic, Portuguese, Spanish, and French. We collected these tweets with the Twitter API by querying emojis that express emotion and restricted the results to the five languages listed above. In this dataset, we have 500 tweets from each of the five languages (250 positive and 250 negative) labeled as either positive or negative.

We used emojis because tweets containing them are most likely to be of that corresponding sentiment. We assume that this technique is almost as good as manual labeling. We followed the same labeling procedure used in the Sentiment140 dataset (Go et al., 2009).

Table 2: Accuracies of the CNN model for each language.

Language	Accuracy %
English	96.11
Arabic	86.00
French	94.34
Spanish	91.00
Portuguese	94.40

The results in Table 3 show that the CNN model performs best in English but also very well in Portuguese, French, and Spanish. For Arabic, the model has between 5 and 10% worse accuracy, probably because the variety of dialects in the Arabic collection of tweets is larger than the other four languages. These dialects are hugely different, causing Google translation to fail when translating tweets from non-standard Arabic into English.

5 CONCLUSION

In this paper, we presented an approach to multilingual sentiment classification. It demonstrates that combining machine translation software, a deep

learning model, and a set of existing NLP methods such as text pre-processing and word embedding leads to successful multilingual sentiment analysis.

We compared five deep learning models by training them on 14,400 (80%) tweets from our dataset and testing them on 3,600 (20%) tweets. Our CNN model achieves the best accuracy of 85.91% and an F1-score of 84.61%.

A second experiment was carried out by applying the winning CNN model on a balanced set of tweets with emojis collected from Twitter. The CNN model achieved satisfying accuracy (higher than 90%) for the European languages, while its accuracy for Arabic tweets is 86%. This difference can be explained by the translation software's inability to translate non-standard Arabic to English and the wide use of non-standard Arabic on social media.

Future work may explore the ability of transformer-based models to successfully tackle the sentiment analysis problem for corpora containing documents in multiple Arabic dialects.

REFERENCES

- Banea, C., Mihalcea, R., & Wiebe, J. (2010). Multilingual subjectivity: Are more languages better? Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010).
- Bautin, M., Vijayarenu, L., & Skiena, S. (2008). International sentiment analysis for news and blogs. ICWSM.
- Go, A., Bhayani, R., & Huang, L. (2009). Twitter sentiment classification using distant supervision. *CS224N project report, Stanford, 1*(12), 2009.
- Google. *Explore the world in over 100 languages*. Google Translate API
- Kim, S.-M., & Hovy, E. (2006). Identifying and analyzing judgment opinions. Proceedings of the human language technology conference of the NAACL, main conference.
- Liu, B., & Zhang, L. (2012). A survey of opinion mining and sentiment analysis. In *Mining text data* (pp. 415-463). Springer.
- Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.
- Porter, M. F. (1980). An algorithm for suffix stripping. *Program, 14*(3), 130-137.
- Portuguese Tweets for Sentiment Analysis*. (2019). Portuguese Tweets
- Ravi, K., & Ravi, V. (2015). A survey on opinion mining and sentiment analysis: tasks, approaches and applications. *Knowledge-Based Systems, 89*, 14-46.
- Saad, M. (2020). *Arabic Sentiment Twitter Corpus*. Arabic Tweets
- Shin, B., Lee, T., & Choi, J. D. (2016). Lexicon integrated CNN models with attention for sentiment analysis. *arXiv preprint arXiv:1610.06272*.
- Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research, 15*(1), 1929-1958.
- Statista. (2019). *Internet: most common languages online 2020* | Statista. Statista 2019
- Taboada, M., Brooke, J., Tofiloski, M., Voll, K., & Stede, M. (2011). Lexicon-based methods for sentiment analysis. *Computational linguistics, 37*(2), 267-307.
- Wang, J., Yu, L.-C., Lai, K. R., & Zhang, X. (2016). Dimensional sentiment analysis using a regional CNN-LSTM model. Proceedings of the 54th annual meeting of the association for computational linguistics (volume 2: Short papers).
- Wu, Y., Schuster, M., Chen, Z., Le, Q. V., Norouzi, M., Macherey, W., Krikun, M., Cao, Y., Gao, Q., & Macherey, K. (2016). Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*.
- Y.Kim. (2014). Convolutional neural networks for sentence classification. *Arxiv (2014)*, pp. 23-31. <https://doi.org/10.1145/1599272.1599278>