Multi-class classification of Twitter Sentiments using Deep learning

Ga Jun Young
ga.young@ucdconnect.ie
16440714

Royal Thomas
royal.thomas@ucdconnect.ie
16326926

Zheng Ju
zheng.ju@ucdconnect.ie
16205656

Abstract

Deep learning is a relatively new field of computer science, stemming from the work of a group of computer scientists in the 1940s. The field gained traction in recent years due to breakthroughs in NLP, computational power, and big data techniques. There is a large quantity of methodologies on classifying sentiment (Positive, Negative, Neutral) of social media services due to the availability of data. Thus, we present an alternative, state-of-the-art, deep learning approach for classifying sentiment of Twitter messages. We trained a model on a dataset (1.6 million tweets) utilizing pre-trained embedding and BiLSTM layers, which resulted in an increased performance to baseline models.

1 Introduction

1.1 Project Background

Twitter is known as a micro-blogging platform. Micro-blogging enables users to publish and receive short textual updates. Bloggers or Tweeters can express their sentiments towards a wide range of topics, whether it is about politics, food, sports, etc. These real-time messages called tweets are instantaneous that have a limit of 140 characters. In general, tweets may have spelling errors, abbreviations, and special characters. As a result, due to the nature of short textual tweets, sentiment analysis becomes an important yet challenging task.

Sentiment analysis is the contextual mining of text that attempts to answer questions about consumer habits, using online data as a starting point. It is extremely important, because it allows marketers to identify which demographics are most receptive to their products. Analysis of social media are different in purpose; however, it is still necessary to interpret and classify the emotions (positive, negative, and neutral) within the textual based data using text analysis techniques.

To address this issue, we propose a neural network-based model that utilizes the deep learning mechanism of Tensorflow to analyse the sentiment of a tweet. It has been demonstrated by the experiments conducted in this paper that this approach increases the performance and accuracy of sentiment, multi-class classification of tweets.

The main motivation behind our approach is the flexibility of constructing deep learning models with Tensorflow. As well as that, there has been a great amount of expression towards Twitter based sentiment analysis. Go et al. [6] serves as the state-of-the-art-approach that utilizes machine learning to analyse sentiment in tweets. The research [6] conducted follows the base-line approach of Pang et al. [15] who utilized machine learning algorithms to classify sentiments of product reviews. Other

Preprint. Under review.

researchers like Alharbi and de Doncker [1] proposes architecture of a Convolutional Neural Network (CNN) that considers more than just text but user's behaviour. Thus, combining a set of ideas from a multitude of research, the direction of the project is to build a deep learning model that handles multi-class classification on a balanced data set provided by Go et al. [6].

1.2 Problem Statement

The construction of a deep learning model takes the form of a pipeline consisting of multiple stages. It is important to make careful decisions at each stage of the pipeline to build an effective model for sentiment analysis. As well as comparing the results to base-line models [6] [19]. The deep learning pipeline flows from a high-level perspective (human language) to that of which a machine understands. The following are the 5 stages of the pipeline:

1. **Textual Preprocessing** - special character removal, lowercase text, contractions
2. **Textual Representation** - Natural Language Processing (NLP) - a field of AI that gives the machine the ability to read, understand, and derive meaning from human languages.
3. **Generate Embedding Matrix** - GLoVe, fastText
4. **Modelling** - LSTM, Bi-LSTM, CNN + LSTM
5. **Deployment** - Evaluation and Predictions

Another problem arises when deciding on a deep learning library. Table 1 provides a basic comparison between Tensorflow and PyTorch. The table views Tensorflow as a library with a steep learning curve, but with the advantage of having a larger community. As a result, the approach of the project is to utilize Tensorflow and it's vast documentation. Furthermore, utilizing the TPU to speed up the training process is much more user-friendly in Tensorflow than PyTorch. Thus, reducing the deep learning model's training time significantly when compared to training on the GPU or CPU.

	Tensorflow	PyTorch
Developed	Google	Facebook
Computational graphs	Static	Dynamic
Learning Curve	Steep	Pythonic
Community	Larger	
Visualization	TensorBoard + Matplotlib	Matplotlib

Table 1: A comparison table of Tensorflow and PyTorch based on [10][23]

1.3 Outline of Report

The remainder of this paper is structured as follows. Section 2 summarizes core sentiment analysis methodologies while presenting results from various researchers. Section 3 provides a detailed run down of the proposed experimental approach. Next, Section 4, shows the results achieved by the proposed system. Finally, section 5 concludes the research and discusses relevant future works.

2 Related Work

2.1 Sentiment Analysis Methodology

Sentiment analysis can be viewed as a set of machine learning algorithms that provide a continuous evaluation of product relationship with customers by exploring reviews, content, and intent [15]. However, there are a multitude of sentiment analysis techniques beyond machine learning algorithms. Alshammari and AlMansour [2] presents 4 main approaches for sentiment analysis: (i) machine learning (supervised learning) [6] [15], (ii) lexicon (unsupervised learning), (ii) hybrid, and (iv) deep learning [1]. As well as that, the research [2] provides state-of-the-art reviews from the 4 approaches.

2.2 Machine Learning Approach (Supervised learning)

The full set of steps involved in a machine learning approach includes preprocessing, labelling, training, testing, and sentiment classification [2]. Go et al. [6] approaches sentiment analysis of

a Twitter dataset by building 3 separate models (multinomial Naïve Bayes, Maximum Entropy, and Support Vector Machines) with a combination of unigram (one word), bigram (sequence of two words), and part-of-speech (POS) features. Go et al. [6] proved that multinomial Naïve Bayes worked well on text categorization with unigram and bigram, achieving an accuracy of 82.7%. On the other-hand, Maximum entropy, a feature-based model achieved 83% with unigram and bigram. Lastly, Support Vector Machine achieved an accuracy of 82.2% with the uni-gram. It is important to note that the achieved accuracy by Go et al. [6] would not have been possible without textual preprocessing such as the removal of emoticons and repeated tweets, letters, and Twitter related features.

2.3 Lexicon Approach (Unsupervised learning)

An opinion lexicon is a database of opinions of various genres of literature. They are widely used to support automatic sentiment analysis of text segments. However, they are required to combine with various techniques such as tagging words with POS, polarity lexicons, and word-level features [4]. Nonetheless, there are a significant amount of dictionaries that are developed for automatic or semi-automatic sentiment lexicons (General Inquirer [17], SentiWordNet [3], and WKWSCI Sentiment Lexicon [12]). Khoo and Johnkhan [12] demonstrates the results of the WKWSCI model with an accuracy rate of 75% to 77% when utilizing appropriate weights for specific sentiment categories. However, the high accuracy rates of these sentiment lexicons are only possible when the lexicon database has enough coverage [1]. Therefore, limitations exist when it comes to tweet based messages that carry informal expressions. The lexicon approach may not be sufficiently satisfactory. As a result, other attempts has been made to extend opinion lexicon dictionaries with automatically annotated Twitter tweets [4].

2.4 Hybrid Approach (Combined Supervised and Unsupervised learning)

Many papers have been presented, describing one of the two main approaches used for sentiment analysis but, both supervised and unsupervised approaches have their own drawbacks. As mentioned in Martín-Valdivia et al. [14], the availability of labeled datasets is not guaranteed for the supervised approach due to the novelty of the task. On the other hand, the unsupervised approach always requires a large amount of linguistic resources. Often, it obtains a lower recall due to its nature which depends on the presence of the words in the dictionary. To overcome those drawbacks and obtain a better performance, Martín-Valdivia et al. [14] proposed a model which combined the supervised and unsupervised approaches. The supervised learning was applied to the corpus using TF-IDF. An unsupervised approach, SentiWordNet [3], was employed over the corpus to generate the unsupervised sentiment classifier. By combining these approaches together, a promising result was obtained with an accuracy of 88.57%. While only applying the supervised approach, an accuracy of 87.66% was generated.

2.5 Word Embedding

A majority of the deep learning models in sentiment analysis have been based on word vector representation [13]. These representations have shown to be very effective in the application of sentiment analysis [21]. Embeddings are methods for learning vector representation of categorical data. Two popular frameworks for learning word embeddings are GloVe [16] and Word2vec. These produce embeddings that can be used to judge whether or not a word belongs in a given context [7]. GloVe and Word2vec are examples of unsupervised learning problems. GloVe is an improvement over Word2vec. GloVe exploits the global statistical information regarding word co-occurrences in order to build better embeddings [16]. FastText is another word embedding method that is based on the Word2vec model. FastText represents words as an n-gram of characters [11]. On the other hand, we can include an embedding layer within our model which can be trained to solve any given problem based on the outcome you provide. Therefore, there are instances, especially when your dataset is small when using a pre-trained embedding layer is useful. On the contrary, there are instances where learning embeddings from your data, by optimizing it for a particular problem, can yield better outcomes [18]. Hartmann et al. [8] trained word embeddings using both GloVe and fastText to build Portuguese word embeddings, between the two GloVe performed the best in terms of syntactic and semantic analogies. Schmitt et al. [20] received the best results with FastText, among the three, for aspect-based sentiment analysis.

2.6 Deep Learning

Uysal and Murphey [22] compared the performance of three different models (CNN+LSTM, CNN and LSTM) on the same dataset as this project. They also compared three different embeddings (one-hot representation, semantic word embedding and fine-tuned semantic word embedding). They received their best results using the LSTM network (0.715) with one-hot encoding for the embedding layer of the model, followed by a fine-tuned semantic word embedding. Their model's accuracy are considerably low compared to our results. They concluded based on their findings that it is better to initialize deep learning models with either one-hot vectors or fine-tuned semantic word embedding rather than using pure word embedding.

Dang et al. [5] compared Term Frequency-Inverse Document Frequency (TF-IDF) and word embedding to discover the better performing model on the Sentiment140 dataset [6]. They employed both the Keras and Tensorflow libraries, and built DNN, CNN, and RNN models to verify the performance of their problem. According to their findings, word embedding work much better than TF-IDF for the purpose of sentiment analysis due to the difference in accuracy (0.76 vs 0.828). Their best accuracy was created from an RNN model utilizing an embedding layer rather than a pre-trained embedding.

Zhong and Yi [24] used deep learning models to classify electronic health record (EHR) text into ICD-10 codes. In their experiment, they compared the abilities of CNN, LSTM, GRU and SVM. For their dataset, CNN outperformed all the other types. Instead of using an embedding layer or a pre-trained embedding matrix, they trained their embedding using word2vec and Chinese medical documents that they crawled and fetched online. As per the authors, the results show that max-pooling produced a much better accuracy (.8) as compared to k-max pooling (.75) with the CNN model.

3 Experimental Setup

3.1 Dataset

The proposed dataset contains 1.6 million tweets extracted from the Twitter API provided by Go et al. [6]. It is a sequential, balanced dataset. The dataset contains 6 fields.

1. **target**: The sentiment of the tweet (0 = negative, 4 = positive)
2. **ids**: The id of the tweet (1234)
3. **date**: The date of the tweet (Sat May 16 23:58:44 UTC 2009)
4. **flag**: The query (lyx). NO_QUERY if no query exists
5. **user**: The user that tweeted
6. **text**: The text of the tweet

3.2 Textual Preprocessing

Tweet messages contain noisy data (special characters, emoticons, and Twitter features). The task of cleaning or preprocessing the data is as important as building a deep learning model. The proposed approach removes noise from the dataset leaving only the text and target fields. The following is a set of steps to preprocess the 1.6 million datasets:

1. **Remove redundant fields**: only the text and target fields are required
2. **Convert tweets to lower case**: standardize words like 'Ham' and 'ham'
3. **Remove URL links**
4. **Remove # in #hashtags**: the words following a # may yield sentiment
5. **Contraction expansion**: words like 'aren't' and 'are not' will be standardized
6. **Remove special characters, mentions, and numbers**: to standardize text
7. **Convert repeated letters in a word**: the repetition of a letter within a word can be a max of 2 e.g. 'hellllo' is converted to 'hello'
8. **Removal of empty text**: empty text yields no sentiment information

3.3 Textual Representation

In NLP projects, the deep learning model cannot accept raw text as input. It can only handle numerical tensors that represent text. Therefore, vectorizing the text to obtain tensorial representation becomes the first task after textual preprocessing. The Tensorflow Keras architecture employed provides a text tokenizer tool to vectorize text. Initially, the preprocessed dataset is split into a training and test set (80% to 20% ratio) and as a result, the text, training set is fitted onto the tokenizer. The results of the tokenizer will map words to their occurrences as a dictionary object. Afterwards, the tokenized text can be sequenced to a vector where each element represents the index of the word. To feed the vectorized text into the model, we must make all text sequences the same length. We simply apply the 'pad_sequences()' method to pad or truncate the sequences – if a max length was provided. After tokenization, the text sequences should all be of equal length which will be fed into our model for training.

3.4 Generate Embeddings Matrix

Tensorflow Keras offers an embedding layer that can be used for neural networks on text data. It requires integer encoding input data which was carried out in textual representation. The embedding is initialized with random weights to map embeddings for all the words within the training set. It is flexible, as it can learn a variety of word embeddings and it can be saved in the form of a matrix for the usage on other models. As well as that, we can utilize pre-trained word embedding models. As a result, the approach for embedding is to use the word embedding of both GloVe [16](Wikipedia word vectors - 300D) and fastText [11](Crawl word vectors - 300D) mapped upon our text representation. This is a type of transfer learning which has been proven to be very efficient in this project. The concatenation of the two embedding, results in a embedding matrix of 600D.

3.5 Model Architecture

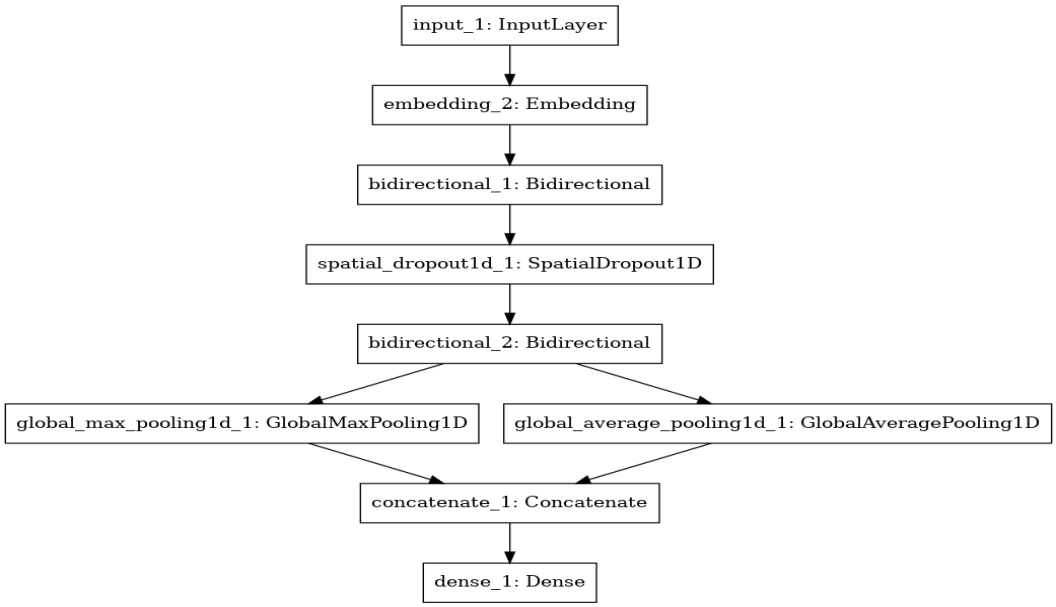

Figure 1: BiLSTM model created using Keras

Figure 1 presents a diagram of the architectural view of the 8-layer model constructed for the purpose of our sentiment analysis problem. It is built by leveraging Tensorflow Keras with varying parameters. The first layer consists of the input with the output shape of (None, 53). The value 53 represents the length of the padded sequence obtained after our text tokenization process. The embedding layer consists of an output shape of (None, 53, 600) with a total number of 137,976,000 parameters. It has inputs from both the input layer and the embedding matrix generated at an earlier stage. The generated embedding matrix is a 600D weighted matrix built using a concatenation of GloVe and

fastText. As a result, the shape of the weight matrix (213269, 600) produced the total number of parameters $(229,960 \times 600 = 137,976,000)$. These parameters are not trainable due to pre-trained embeddings. The embedding layer does not require weight updates during this training process.

The next three layers were 2 Bidirectional LSTM (BiLSTM) layers with a 'Spatial Dropout' between them. Both BiLSTM layers had an output shape of (None, 53, 200). The number of units (hidden neurons) in each LSTM cell was configured as 100. The dimension of each cell is the concatenation of the current and previous states. This results in an input vector with 700D (600 + 100) for each LSTM. Each cell consists of 4 activation functions (3 sigmoid and 1 tanh). Each activation function has 100 parameters. Therefore, the total number of parameters in the first LSTM layer can be computed as $700 \times (100 \times 4) + (100 \times 4)$ resulting to 280,400 parameters. Since we employed BiLSTM, the total number of parameters are doubled (560800). The same explanation applies to the second BiLSTM. The spatial dropout layer resides between the two layers to prevent overfitting. Unlike 'Dropout', spatial dropout disconnects the entire feature map, not a single neuron. A common Dropout cannot regularize its output if a strong correlation between the adjacent elements of a feature map exists. This will cause a significant decrease in the learning efficiency.

Next, the model relies on two parallel layers, GlobalMaxPooling1D (GMP) and GlobalAverage-Pooling1D (GAP). The pooling layers receive the output of the second BiLSTM layer (None, 53, 200) as input. The GMP and GAP layer extract the maximum and average value of each feature map respectively, with a shape of (None, 53). The two layers generate two feature vectors of the shape (1, 200). They are concatenated in the subsequent layer. The purpose of GMP and GAP is to provide a simpler and smoother conversion of feature map to classification. They also reduce dimensionality and parameters making the model stronger and less likely to overfit. The last layer is a fully connected layer. It integrates the output of the previous layer and employs the sigmoid function to classify the input.

Figure 1 relies on an early stopping call back function available on Keras. The function is called during the training procedure at different stages (training begins, training ends, epoch begins, epoch ends). For our approach, we stopped training when the validation loss increases between each epoch. This effectively stops training and saves the best model into a '.h5' file. This enables reproducibility, allowing the model to be loaded and tested upon.

4 Results

For our analysis, we considered the performance of 1 metric to evaluate the model. Accuracy is a common metric that measures all the correctly identified cases. It is generally used when all classes are important. Since we perform training on a balanced dataset of positive and negative classes, the accuracy metric fits well. Furthermore, many research on sentiment analysis [2] [5] [6] [14] compares through accuracy. As a result, this observation encourages our approach to base on the same metric for comparison purposes.

Table 4 provides a synopsis of 3 models that we have configured with varying hyper-parameters. The best accuracy obtained is through BiLSTM with pre-trained gloVe and fastText embedding. To analyse how we generated such a model we look at batch size, dropout rate, and the number of epochs to reach the accuracy. Using a dropout rate of 0.2, a batch size of 512, and an 5 epochs, the best BiLSTM model produced an accuracy of 84.30% with a validation loss of 35.59%. To prove this case, we fine-tuned the model with dropout rates from 0 to 1, and a set of batch sizes in the power of 2. The CPU and GPU are organized memory in power of 2. As a result, a batch size of power of 2 will help align CPU and GPU to page boundary, speeding up fetching of data to memory. Nevertheless, even with further fine-tuning the accuracy of this model did not exceed 84.30% without overfitting. We also trained a LSTM and a CNN+LSTM model, however, as shown on Table 4 the accuracy does not exceed the BiLSTM model.

In comparison to state-of-the-art approaches, Go et al. [6] yields their best result with maximum entropy, a machine learning methodology, achieving 83%. Dang et al. [5] produces a deep learning approach with TF-IDF with word embeddings on the sentiment140 dataset [6] achieving 82.8%. In both cases, our approach of BiLSTM produces a much higher accuracy on the same dataset with an increase of about 1%.

We consider the LSTM model as our baseline approach. It is a generic layer that resides in some variant throughout our 6 models. The LSTM model with a pre-trained concatenation of word embeddings produced an accuracy of 83.93%. Overall, the baseline exceeds the accuracy of 4 other models. This was configured with a batch size of 125 and an early stop at the 7th epoch. In comparison with state-of-the-art approaches, the LSTM model beats the machine learning approach by Go et al. [6] and the deep learning approach by Dang et al. [5]. However, we discovered a superior variant of the LSTM model which is the bidirectional form. As a result, the best model yields 84.30%.

Model	*Trainable Embedding Layer*	*Pre-trained Glove+FastText*
LSTM	0.8288	0.8393
CNN+LSTM	0.8273	0.8345
BiLSTM	0.8216	0.8430

Table 2: Performance of Models with and without Pre-Trained Word Embeddings

5 Conclusion & Future Work

This paper proposes different deep learning models that classify the sentiment of tweets into positive, negative, or neutral classes. We compared a wide variety of models trained using a large corpus of tweets (1.6 million) called 'Sentiment140'. Our proposed model achieved a high accuracy above 84%. Furthermore, we compared the performance of a trainable embedding layer with a set of pre-trained embedding layers (GloVe + fastText). We discovered that the pre-trained embedding layers improved the performance of our models considerably (1-3% based on model). Our results show that even with a large corpus of training data, some models can benefit from pre-trained embedding layers.

Based on the task of multi-class classification, we have utilized the sigmoid function as the activation function for our final layer. As a result, the output of the prediction is categorized into three ranges as follows:

$$x = \begin{cases} Negative & \text{if } x < 0.4 \\ Neutral & \text{if } 0.4 \geq x \leq 0.6 \\ Positive & \text{if } x > 0.6 \end{cases}$$

While the sigmoid function is not a probability distribution, it is a cumulative distribution function of logistic distribution [9] with an interval of [0-1]. This allows us to assume that the output is a confidence of classification for a given label. It also enables us to produce a multi-class classification from a dataset that only contains data with two classes.

There are a very few interesting avenues to take with respect to future work,

1. While we checked our efficiency with various pre-trained GloVe (Twitter, Wikipedia, crawl) and fastText embeddings, there are other state-of-the-art embedding methods that can explore and test this dataset (Sentiment140). During our research, we discovered BERT and ELMo as possible options, but we encountered implementation issues due to technical limits.
2. Another interesting approach is to use classifiers to produce additional features for a given dataset and examine the performance of the aided task. For example, we could apply the classifier on historic tweets talking about companies before their stock had considerably changed. Then, we could examine if a correlation exists between the sentiment of tweets, before the event and the change of the stock value ex post facto.

Acknowledgments

The authors would like to thank Ellen Rushe for providing feedback on specific queries.

References

[1] A. S. M. Alharbi and E. de Doncker. Twitter sentiment analysis with a deep neural network: An enhanced approach using user behavioral information. *Cognitive Systems Research, Elsevier*, 54:50–61, 2019. ISSN 1389-0417.

[2] N. F. Alshammari and A. A. AlMansour. State-of-the-art review on twitter sentiment analysis. In *2019 2nd International Conference on Computer Applications Information Security (ICCAIS)*, pages 1–8, 2019.

[3] S. Baccianella, A. Esuli, and F. Sebastiani. Sentiwordnet 3.0: an enhanced lexical resource for sentiment analysis and opinion mining. In *Lrec*, volume 10, pages 2200–2204, 2010.

[4] F. Bravo-Marquez, E. Frank, and B. Pfahringer. Building a twitter opinion lexicon from automatically-annotated tweets. *Knowledge-Based Systems*, 108:65 – 78, 2016. ISSN 0950-7051.

[5] N. C. Dang, M. N. Moreno-García, and F. D. la Prieta. Sentiment analysis based on deep learning: A comparative study. *Electronics*, 9(3):483, Mar. 2020.

[6] A. Go, R. Bhayani, and L. Huang. Twitter sentiment classification using distant supervision. *CS224N Project Report, Stanford*, page 12, 2009.

[7] Y. Goldberg and O. Levy. word2vec explained: deriving mikolov et al.'s negative-sampling word-embedding method. 2014.

[8] N. Hartmann, E. R. Fonseca, C. Shulby, M. V. Treviso, J. S. Rodrigues, and S. M. Aluísio. Portuguese word embeddings: Evaluating on word analogies and natural language tasks. *CoRR*, abs/1708.06025, 2017.

[9] G.-B. Huang, Q.-Y. Zhu, K. Mao, C.-K. Siew, P. Saratchandran, and N. Sundararajan. Can threshold networks be trained directly? *IEEE Transactions on Circuits and Systems II: Express Briefs*, 53(3):187–191, Mar. 2006. doi: 10.1109/tcsii.2005.857540. URL `https://doi.org/10.1109/tcsii.2005.857540`.

[10] Y. Jain. Tensorflow or pytorch : The force is strong with which one?, Apr 2018. URL `https://medium.com/@UdacityINDIA/tensorflow-or-pytorch-the-force-is-strong-with-which-one-68226bb7dab4`.

[11] A. Joulin, E. Grave, P. Bojanowski, and T. Mikolov. Bag of tricks for efficient text classification. *arXiv preprint arXiv:1607.01759*, 2016.

[12] C. S. Khoo and S. B. Johnkhan. Lexicon-based sentiment analysis: Comparative evaluation of six sentiment lexicons. *Journal of Information Science*, 44(4):491–511, 2018.

[13] Y. Kim. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*, 2014.

[14] M.-T. Martín-Valdivia, E. Martínez-Cámara, J.-M. Perea-Ortega, and L. A. Ureña-López. Sentiment polarity detection in spanish reviews combining supervised and unsupervised approaches. *Expert Systems With Applications*, 40(10):3934–3942, 2013.

[15] B. Pang, L. Lee, and S. Vaithyanathan. Thumbs up? sentiment classification using machine learning techniques. In *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing (EMNLP 2002)*, pages 79–86, July 2002.

[16] J. Pennington, R. Socher, and C. D. Manning. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, 2014.

[17] M. S. S. D. M. O. Philip J. Stone, Dexter C. Dunphy. The general inquirer: A computer approach to content analysis. *American Journal of Sociology*, 73(5):634–635, 1968. ISSN 00029602, 15375390.

[18] Y. Qi, D. S. Sachan, M. Felix, S. J. Padmanabhan, and G. Neubig. When and why are pre-trained word embeddings useful for neural machine translation?, 2018.

[19] P. Ripamonti. Twitter sentiment analysis, 2019. URL https://www.kaggle.com/paoloripamonti/twitter-sentiment-analysis?scriptVersionId=9102236.

[20] M. Schmitt, S. Steinheber, K. Schreiber, and B. Roth. Joint aspect and polarity classification for aspect-based sentiment analysis with end-to-end neural networks. *CoRR*, abs/1808.09238, 2018. URL http://arxiv.org/abs/1808.09238.

[21] D. Tang, F. Wei, B. Qin, T. Liu, and M. Zhou. Coooolll: A deep learning system for twitter sentiment classification. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, pages 208–212, Dublin, Ireland, Aug. 2014. Association for Computational Linguistics. doi: 10.3115/v1/S14-2033. URL https://www.aclweb.org/anthology/S14-2033.

[22] A. K. Uysal and Y. L. Murphey. Sentiment classification: Feature selection based approaches versus deep learning. In *2017 IEEE International Conference on Computer and Information Technology (CIT)*. IEEE, Aug. 2017. doi: 10.1109/cit.2017.53. URL https://doi.org/10.1109/cit.2017.53.

[23] S. Yegulalp. What is tensorflow? the machine learning library explained. *InfoWorld.com*, Jun 06 2018.

[24] J. Zhong and X. Yi. Categorizing patient disease into ICD-10 with deep learning for semantic text classification. In *Recent Trends in Computational Intelligence [Working Title]*. IntechOpen, Feb. 2020. doi: 10.5772/intechopen.91292. URL https://doi.org/10.5772/intechopen.91292.