APPLIED MACHINE LEARNING SYSTEMS II (ELEC0135) 19/20 REPORT

SN: 19180271

ABSTRACT

Twitter based sentiment analysis is a very popular task for researchers. Twitter being a medium of hih public interation, opinions on various topics are posted quite often making it a haven for data mining. This data can be used to understand emotions of public towards various issues. This is useful for elections, product improvements etc. Here the task is done on SemEval 2017 competition, which is a prestigious competition for Sentiment Analysis. The two tasks chosen are Polarity classification of tweet as positive, negative and neutral, and Topic based sentiment classification of tweet. The approach used is a character based CNN and LSTM model, which has not been tried a lot in Twitter sentiment analysis. The project is in fact an attempt at testing character based models for twitter sentiment analysis. Test accuracies of 44.6% and 84.26% were obtained for both the tasks respectively, which is comparable with the best teams in SemEval 2017 competition. [1]

1. INTRODUCTION

The work in this report deals broadly with Twitter Sentiment Analysis (TSA). People use Twitter often to raise their opinions on various subjects from politics to movies to music to art. This makes Twitter a gold mine for data analysts in general, and sentiment analysts in particular. For example, one could understand the general trend of voters during an election from twitter. The inherent challenge in Twitter sentiment analysis owes to the short nature of tweet, due to the 140 character limitation, and the extensive use of short-form words and emoticons [1]. This makes it difficult to analyse tweets using common word embeddings.

Our task here is derived from SemEval Task 4 - Sentiment Analysis in Twitter [2]. The tasks chosen in particular are:

1. Subtask A: Given a tweet, decide whether it expresses POSITIVE, NEGATIVE or NEUTRAL sentiment [2].

2. Subtask B: Given a tweet and a topic, classify the sentiment conveyed towards that topic on a two-point scale: POSITIVE vs. NEGATIVE [2].

The main difference in the two tasks is than while Task A is just sentiment analysis, task B is Topic Based Sentiment Analysis. Often a statement might be negative in overall sentiment but positive towards a topic. Refer Figure 1 for example.

[1]The link to the code is provided here- GitHub project: https://github.com/ash95sv/AMLS_II_assignment19_20-

Tweet	Overall Sentiment	Topic-level Sentiment
Who are you tomorrow? Will you make me smile or just bring me sorrow? #HottieOfTheWeek Demi Lovato	NEUTRAL	Demi Lovato: POSITIVE
Saturday without Leeds United is like Sunday dinner it doesn't feel normal at all (Ryan)	WEAKLYNEGATIVE	Leeds United: HIGHLYPOSITIVE
Apple releases a new update of its OS	NEUTRAL	Apple: NEUTRAL

Fig. 1. Overall Sentiment vs. Topic based Sentiment

A deep learning approach has been adopted for both the tasks. Both tasks use character-based deep learning model making use of both convolutional layers as well as bi-directional LSTMs. LSTMs are a natural go-to for language related tasks because of how they process data sequentially and have inherent memory. Language documents, being sequences of words, which in turn are sequences of characters become naturally fit for LSTM structure.

The idea to use character based model is inspired by [3]. The intuition comes from the fact that characters are the most basic building blocks of written language. If images can be learnt from pixels, it is not a distant idea to learn document from characters. This project is an attempt at testing character models for sentiment analysis. The focus of this project is not just to get good accuracies, but to understand character based models and their suitability for such a task.

The report is organised as follows. Section 2 is Literature Survey, which gives a brief overview of potential approaches to solve the tasks. Section 3 deals with detailed description of models used for both tasks including data preprocessing steps and rationale for choice of model. Section 4 deals with implementation details of model. It discusses the libraries used, hyper-parameter selection, algorithms and learning curves. Section 5 discusses and analyses the results obtained. The report is concluded in Section 6 and discusses scope for future improvements.

2. LITERATURE SURVEY

Since both Tasks A and B essentially deal with Sentiment Analysis, the literature survey is discussed commonly for both tasks. The general approaches to TSA are Machine Learning and Deep Learning based, Lexicon based, hybrid approach and graph based. Lexicon based approach uses a dictionary of words to understand polarity of tweet [1].

Majority of the methods in Twitter Sentiment Analysis uses Machine learning algorithms, the most popular of them being either Support Vector Machines (SVM) , Naive Bayes

(NB) classifier or Maximum Entropy Classifier (MaxEnt) [1]. One of the first studies dealing with TSA was carried out by Go et al. [4], who treated the problem as a binary classification, classifying the tweets as either positive or negative. They examined NB, MaxEnt, and SVM classifiers. Bigrams, unigrams, and POS tags were used as features. They also reported that the most effective method was using NB with bigrams as features, which managed to achieve an accuracy of 82.7% [4].

But more recently, Deep learning approaches are gaining prominence due to their ability to learn with minimal manual feature extraction [1]. Wang et. al. used a join CNN-RNN architecture and were able to outperform then existing feature-engineered approaches [5]. The joint CNN-RNN architecture takes advantage of the coarse-grained local features generated by CNN and long-distance dependencies learned via RNN for sentiment analysis of short texts.They were able to achieve results better than existing state of the art methods with accuracy nearing 89% [5].

In their paper “Exploring the Limits of Language Modeling”, the Google Brain team show that a character level language model can significantly outperform state of the art models based on carefully tuned n-grams [6]. Their best performing model combines an LSTM with CNN input over the characters. Figure 2 represents their approach.

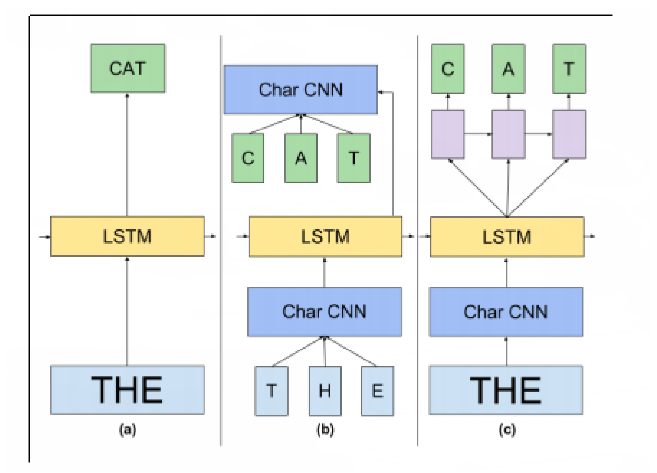


Fig. 2. Character level language model developed by Google Brain Team

“Text Understanding from Scratch” by Zhang et. al. [3] uses pure character level convolution networks to perform text classification with impressive performance. It is a seminal paper in understanding character level embeddings. They applied temporal convolutional networks called ConvNets to various large-scale datasets, including ontology classification, sentiment analysis, and text categorization [3]. They showed that temporal ConvNets can achieve astonishing performance without the knowledge of words, phrases, sentences and any other syntactic or semantic structures with regards to a human language [3]. They achieved an astonishing 95% accuracy for sentiment analysis of Amazon review dataset [3]. Figure 3 shows the strcuture of the ConvNet.

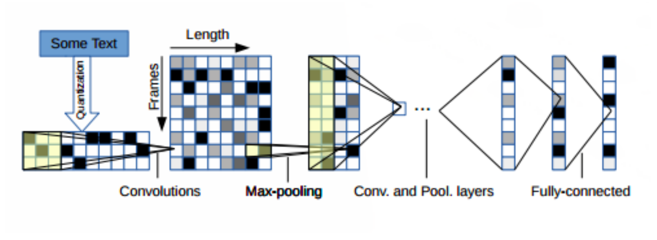


Fig. 3. The character based ConvNet developed in above paper

In ”A Character-based Convolutional Neural Networkfor Language-Agnostic Twitter Sentiment Analysis” , Jonatas et.al. proves the superiority of characted based models for sentiment analysis, even for unknown languages [7]. This is because of the fine-grained granularity offered by characters. They developed a Conv-Char-S network which understands sentences by processing a character-level input. Figure 4 depicts the proposed approach.The authors prove that the network is capable of correcting typos, understanding relationships between words, and characters present in the known vocabulary [7]. Theirs is the first paper to investigate character-based neural networks for language-agnostic translation-free sentiment classification in multilingual scenarios. They were able to outperform all other proposed deep architectures and traditional SVM-based approaches, achieving state-of-the-art results in the proposed task [7].

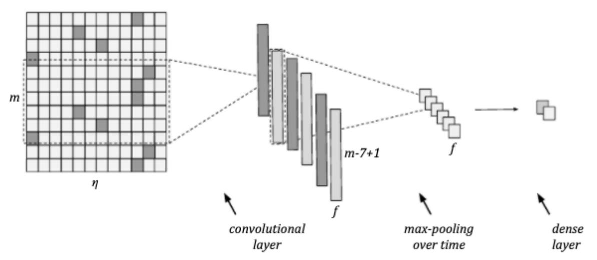


Fig. 4. Conv-Char-S network

Thus the literature survey suggests the efficacy of using character based deep neural networks for our tasks as well.

3. DESCRIPTION OF MODELS

Task A and Task B are primarily sentiment analysis tasks, with Task B on Topic-based sentiment Analysis. Hence the approach used is similar for both tasks, which is explained in detail in subsection 3.1. Subsection 3.2 will focus on the additional steps to account for the topic.

3.1. Task A: Tweet Polarity Classification

The first task focuses on classifying a tweet as Positive, Negative or Neutral in sentiment. This makes it a 3 class classification problem. Based on the literature review, it was decided to go ahead with character based deep learning model.

As part of data pre-processing, the tweets were divided into sentences. And each sentence in a tweet was parsed through to find the total set of unique chatacters in the entire dataset. These charcters were one-hot embedded and each tweet was represented in terms of their character embedding. Because tweets have a 140 character limitation, it was easy to choose the dimension of vector in which the tweets would be stored, which was 140. We also took 10 as the maximum number of sentences possible in a tweet. Hence each tweet would be represented as 10 sentences in a 140 dimension space.

The model has 2 levels of convolutional-pooling-dropout followed by 2 Bidirectional LSTM layers. This is followed by the classification layer.

The detailed structure is as follows :

1. 1D Convolution Layer which takes in as input the set of tweets that are encoded in terms of their characters. This layer has 8 filters, generated by kernel size 5. This layer is to learn the broader features from characters.

2. MaxPooling layer that chooses the maximum weight from a 2x2 sized area.

3. Dropout layer to randomly drop weights and prevent over-fitting.

4. The above 3 layers are repeated once more, but with 16 filters and kernel size 3 respectively. The purpose of the stacked 1D convolution and maxpooling layers is to learn words from characters, much like how lines, dots and shapes are learnt from pixels in image convolution.

5.The stacked 1D convolution layers are topped with a bidirectional LSTM. Bi directional LSTM can supposedly learn positions of words in a sentence better than unidirectional LSTM. This is because it is has two networks, one accesses information in forward direction and another accesses in the reverse direction 5. These networks have access to the past as well as the future information and hence the output is generated from both the past and future context. So it can better learn the position of a word in a sentence because it has access to words that come before and the words that come after [8].

The above 5 components make up the first part of our network, which is a sentence encoder. The sentence encoder learns sentences from characters.

6. Following the sentence encoder, the tweet encoder is developed using another Bi directional LSTM. This LSTM is used to determine the position of each sentence in the entire tweet.

The motive behind using bidirectional LSTM for encoding the tweets is that by getting information of sentence arrangement from both directions, sentiment can be analysed more effectively [9] . This is because context can be learnt better if we have access to the entire sentence at once, rather than just the first part. For example, a tweet like ”I love this song. Not anymore” is easily perceived as positive at first and then second sentence turns out to be negative. A biLSTM can will learn ”I love” and ”not anymore” together to see that it is not a positive emotion being conveyed.

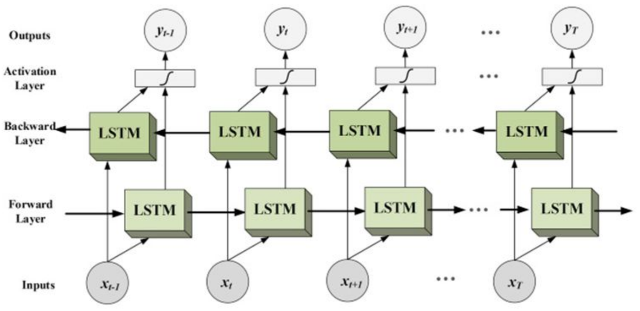


Fig. 5. Bidirectional LSTM

7. Finally the model is connected to Dense layer followed by classification layer that classifies the tweet as positive, negative or neutral.

The model is evaluated using Average Recall, which is Recall averaged over the 3 classes positive, negative and neutral. It is computed as follows:

$$AvgRec = \frac{1}{3}(R^P + R^N + R^U)$$

where R^P , R^N and R^U refer to recall with respect to the POSITIVE, the NEGATIVE, and the NEUTRAL class, respectively [2].The advantage of AvgRec over accuracy is that it is more robust to class imbalance. The accuracy of the majority-class classifier is the relative frequency of the majority class, that may be much higher than 0.5 if the test set is imbalanced [2]. Apart from measuring average recall, accuracy is also used as a secondary metric.

Further, in order to truly evaluate the model a baseline was developed based on SemEval2017 [2]. 3 baselines are developed, assigning all labels as either positive, negative or neutral respectively.

3.2. Task B: Topic based Tweet Sentiment Classification

The task of predicting sentiment of a tweet towards a topic is also performed similar to task A. Here there are only 2 classes - POSITIVE and NEGATIVE.

In order to account for the topic, the topic information which is obtained with the dataset is concatenated along with the tweet [10]. This is to make topic level information explicit in the tweet. This combined tweet is then processed according to the steps in task A.

The other tweak is in the model itself, where instead of 2 levels of convolution layer, there are 3 instead with 8,16 and 16 filters respectively with kernel size 5,3 and 3 respectively. The additional layer helps in further embedding the topic level information that we concatenated with the tweet.

The remaining layers are the same as Task A, with a BiLSTM for sentence encoding, a BiLSTM for tweet encoding and finally classification layer.

This task also uses Average Recall as the primary metric for evaluation in order to take care of class imbalances. It is computed as follows:

$$AvgRec = \frac{1}{2}(R^P + R^N)$$

where R^P and R^N refer to recall with respect to the POSITIVE and the NEGATIVE class respectively [2].

Further, in order to truly evaluate the model a baseline was developed again based on SemEval2017 [2]. 2 baselines are developed, assigning all labels as either positive or negative respectively.

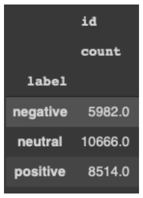
4. IMPLEMENTATION

This section provides the details of model implementation for both the tasks. It is divided into data preparation and pre-processing, and model implementation for each task.

4.1. Task A: Tweet Polarity Classification

4.1.1. Data Preparation

Data for this task is obtained from SemEval website. There were multiple datasets which had to be combined in order to make a substantial set of training , validation and testing data. The dataset consists of user id, sentiment label and tweet as its columns. The sentiment label takes 3 values - positive, negative and neutral. The combined dataset consisted of a total of 25162 tweets. Fig. 6 shows the distribution of each class label within the total dataset.



	id
	count
label	
negative	5982.0
neutral	10666.0
positive	8514.0

Fig. 6. Class distribution

This was split into training, validation and testing data using Sci-kit Learn's train_test_split function in the ratio 60:20:20 respectively.

Further the data was pre-processed to make it suitable for the learning algorithm. This requires the tweet to be stripped of unnecessary characters such as hashtags #, URLs, usernames which are not useful for analysing sentiment. The tweet is converted to lowercase character first and the stripping of unwanted characters are done. This is done with the help of stopwords and punkt packages in NLTK.

Each tweet is then divided into sentences. Each sentence is then tokenized using NLTK Tokenizer to get rid of misspellings, extra characters etc.

The steps are summarised in the following Fig.7

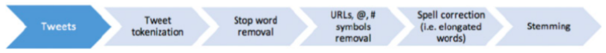

Tweets
Tweet tokenization
Stop word removal
URLs, @, # symbols removal
Spell correction (i.e. elongated words)
Stemming

Fig. 7. Tweet Pre-Processing steps

Once the tokenised tweets are obtained, we pass it to a function for character encoding. The character encoder first find the set of unique characters in the entire tokenised tweet dataset. In this case it turns out that there are 51 unique characters. Compared to word embeddings, this is far less because had we used word embeddings, we would have required thousands of words to represent our tokensied dataset. The characters are not yet one-hot encoded because saving the tweets in the form of one-hot encoded arrays will take up a lot of memory. Rather, the encoding of characters will be done on the fly as we train the neural networks.

The next step is to create the vectors for tokenized tweet. A tweet is composed of 10 sentences (the limit placed) and each sentence is limited to maximum 140 characters.

4.1.2. Baseline Models

3 baseline models are created for comparing our deep neural model. The baselines consist of predicted values being either all positive, all negative and all neutral.

On comparing with actual test values, the following accuracies were obtained:

Baseline	Accuracy
Positive	0.33
Negative	0.24
Neutral	0.42

The best baseline seems to be the neutral one, which is obvious because the dataset has mostly neutral labels.Our Char-CNN-LSTM can be deemed to be successful if the accuracy obtained is better than 42%, which is that of the neutral baseline.

4.1.3. Model Implementation

The model is implemented using Keras API for Tensorflow. As mentioned in the design part, there are 2 Convolution-MaxPooling-Dropout layers in series. The convolution is done with 1DConv, as test data is 1 dimensional. Each time the convolutional layer takes a in input tweet, the characters in the tweet are one-hot encoded on the fly for processing. The first 1D Conv layer has 8 filters with kernel size 5 and stride as 1. We choose stride=1 to not miss relation between any character. The second Convolution layer has 16 filters with kernel size 3. The smaller kernel size is to learn finer features and relations among characters and learn words.

A dropout of 0.3 is provided after trial and error. This was done to reduce overfitting, as a perfectly weighted model might learn the training data too well and then will not be suitable for unseen testing data.

Further the first bi-directional LSTM is chosen with 25 hidden units. This is again based on repeated training with different parameters. This LSTM layer is used to learn sentences from words. Hence it makes sense to have nearly as many hidden units in a LSTM as there are words in a tweet.

Finally the last biLSTM which learns the entire tweet from sentences has 10 hidden units.

The model is trained using ADAM optimiser [11]. ADAM uses adaptive learning rate based on parameters. So the learning rate adapts, for example, it decreases as the convergence is approached. It uses the squared gradients to scale the learning rate and it takes advantage of momentum by using moving average of the gradient instead of gradient itself [11],[12]. This makes ADAM a superior optimiser.

Other hyper-parameters tuned were batch size and epochs. The model is trained for 50 epochs and batch size of 32. But if validation loss was not observed to decrease for 5 epochs, training was stopped.These parameters were also tuned on a trial-and-error basis , observing the learning curve and losses. The training is done on Google Colab TPU, which made it possible to speed up the training process way more.

It was observed despite multiple hyperparameter changes that learning curve was somewhat overfitting. It was decided to go with the option that overfit the least.

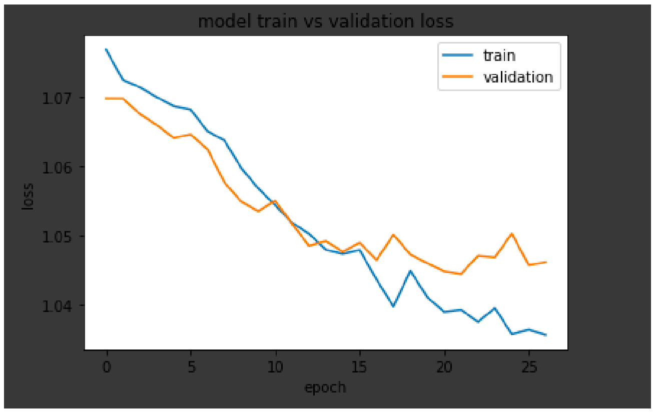
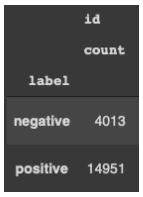


Fig. 8. Learning Curve for TaskA

4.2. Task B: Topic Based Tweet Sentiment Analysis

4.2.1. *Data Preparation*

The Data preparation steps are done similar to task A. Data for this task is obtained from SemEval website. There were multiple datasets which had to be combined in order to make a substantial set of training , validation and testing data. The dataset consists of user id, topic, sentiment label and tweet as its columns. The sentiment label takes 2 values - positive and negative. The combined dataset consisted of a total of 18964 tweets. Fig. 9 shows the distribution of each class label within the total dataset.



	id
	count
label	
negative	4013
positive	14951

Fig. 9. Distribution of labels in Task B

The remaining data pre-processing steps are same as in Task A, except for one step. The topic information is concatenated with the tweet, at the end of the tweet, based on approach mentioned in [10].

4.2.2. *Baseline Models*

2 baseline models are created for comparing our deep neural model. The baselines consist of predicted values being either all positive and all negative.

On comparing with actual test values, the following accuracies were obtained:

Baseline	Accuracy
Positive	0.79
Negative	0.20

The best baseline seems to be the positibe one, which is obvious because the dataset has mostly positive labels.Our Char-CNN-LSTM can be deemed to be successful if the accuracy obtained is better than 79%, which is that of the neutral baseline.

4.2.3. *Model Implementation*

The model is similar to task A, with one additional Conv-MaxPool-Dropout layer with 16 filters and kernel size 3. This additional layer seems to help with learning about the topic as well.

The BiLSTM layers here have 20 hidden units, based on trial-error tuning. Dropout is also maintained to be 0.3. ADAM optimizer is used for training convergence. Training was done over 50 epochs, but if validation loss was not observed to decrease for 5 epochs, training was stopped. The following learning curve was obtained as in fig. 10.

It can be observed that the learning curve is much better as compared to task A.

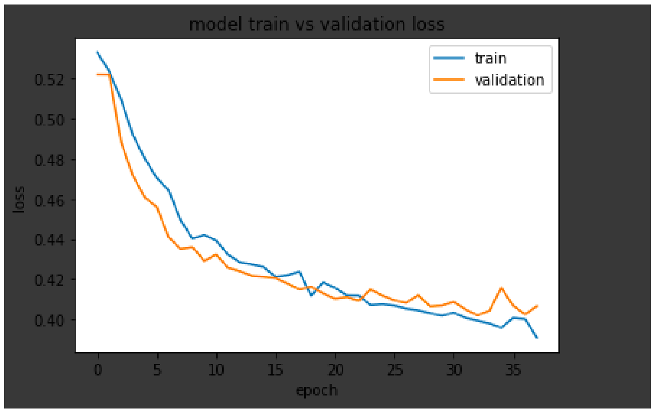


Fig. 10. Learning Curve for Task B

5. EXPERIMENTAL RESULTS AND ANALYSIS

The following final accuracies were obtained for Task A and Task B as in table 5. The accuracy values are reported in percentage.

Task	Train Acc	Val Acc	Test Acc
A	44.03	43.97	44.6
B	83.22	83.21	84.26

It can be observed that the model does perform better than the baseline, especially for Task B. In task A the improvement is only by about 2%. The model generally performs training also better on Task B dataset. This can be explained by the fact that the Task B has only 2 classes to predict whereas Task A had 3 classes, that too with imbalances.

The results obtained here compare quite well with SemEval 2017 contestants, particularly task B. The accuracy in Task A compares with the 35th best team, whereas for Task B it compares with the 5th best team [2].

6. CONCLUSION

This project performed Sentiment Analysis and Topic Based Sentiment Analysis on Tweets based on SemEval 2017 competition. The approach taken was to use character level CNN and LSTM which seems to have done quite well, although not exceptional. But it is still promising, because the fundamental idea to learn from building blocks of text is proven to be strong. There is scope for future improvement, by tweaking the hyperparameters around further. The number of filters in Convolution layer and the LSTM hidden units can be tweaked.It would also be worth trying L2/L1 regularisation to obtain better results, as in [13]. The fact that the approach compares well with the participants of SemEVal 2017 competition also shows it is a step in the right direction.

7. REFERENCES

[1] Anastasia Giachanou and Fabio Crestani, “Like it or not: A survey of twitter sentiment analysis methods,” *ACM Computing Surveys*, vol. 49, pp. 1–41, 06 2016.

[2] Sara Rosenthal, Noura Farra, and Preslav Nakov, “SemEval-2017 task 4: Sentiment analysis in twitter,” in *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, Vancouver, Canada, Aug. 2017, pp. 502–518, Association for Computational Linguistics.

[3] Xiang Zhang and Yann LeCun, “Text understanding from scratch,” 2015.

[4] Alec Go, Richa Bhayani, and Lei Huang, “Twitter sentiment classification using distant supervision,” *Processing*, vol. 150, 01 2009.

[5] Xingyou Wang, Weijie Jiang, and Zhiyong Luo, “Combination of convolutional and recurrent neural network for sentiment analysis of short texts,” in *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, Osaka, Japan, Dec. 2016, pp. 2428–2437, The COLING 2016 Organizing Committee.

[6] Rafal Jozefowicz, Oriol Vinyals, Mike Schuster, Noam Shazeer, and Yonghui Wu, “Exploring the limits of language modeling,” *arXiv preprint arXiv:1602.02410*, 2016.

[7] Jônatas Wehrmann, Willian Becker, Henry Cagnini, and Rodrigo Barros, “A character-based convolutional neural network for language-agnostic twitter sentiment analysis,” 05 2017, pp. 2384–2391.

[8] Mike Schuster and Kuldip Paliwal, “Bidirectional recurrent neural networks,” *Signal Processing, IEEE Transactions on*, vol. 45, pp. 2673 – 2681, 12 1997.

[9] F. Long, K. Zhou, and W. Ou, “Sentiment analysis of text based on bidirectional lstm with multi-head attention,” *IEEE Access*, vol. 7, pp. 141960–141969, 2019.

[10] Mathieu Cliche, “BB_twtr at SemEval-2017 task 4: Twitter sentiment analysis with CNNs and LSTMs,” in *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, Vancouver, Canada, Aug. 2017, pp. 573–580, Association for Computational Linguistics.

[11] Diederik P Kingma and Jimmy Ba, “Adam: A method for stochastic optimization,” *arXiv preprint arXiv:1412.6980*, 2014.

[12] Vitaly Bushaev, “Adam-latest trends in deep learning optimization.,” Oct 2018.

[13] Andrew Y. Ng, “Feature selection, l1 vs. l2 regularization, and rotational invariance,” in *Proceedings of the Twenty-First International Conference on Machine Learning*, New York, NY, USA, 2004, ICML ’04, p. 78, Association for Computing Machinery.