LOVELY PROFESSIONAL UNIVERSITY

INT248 PROJECT REPORT

on

Twitter Comment Classifier

Project Link - https://github.com/Nimagna123/TwitterCommentClassifier

Submitted by

Nimagna Jain

Registration No : 1181079

Programme Name ; Btech. CSE

Under the Guidance of

Ankita Wadhawan

School of Computer Science & Engineering

Lovely Professional University, Phagwara

ACKNOWLEDGEMENT

I would like to express my special thanks of gratitude to my teacher Antika Wadhawan who gave me the golden opportunity to do this wonderful project on the topic Twitter Sentiment Analysis, which also helped me in doing a lot of Research and I came to know about so many new things I am really thankful to them.

Secondly I would also like to thank my parents and friends who helped me a lot in finalizing this project within the limited time frame.

Abstract

In the social media, Twitter has been the most engaging platform for a long time and many companies, political personalities, celebrities have their presence in twitter which makes it a great place for having discussions by millions of users about various topics ranging from laws passed by the parliament to the new movie releases. This makes it interesting to analyse the sentiments of the tweets to see if a tweet by a certain user has positive or negative impact on the community. Going a step further, we can also find which words of the tweet contribute to the sentiment. The analysis of public reaction can be easily done using the sentiment analysis and the keyword extraction of the tweets.

This project report presents the sentiment analysis of tweets using various deep learning algorithms and compare their performance using different metrics. The dataset that is considered encompasses a broad set of tweets which are classified into 3 different types of sentiments, namely, positive, negative and neutral.

1 Introduction

Every day, on an average, 500 million tweets are being sent on the social networking site, Twitter, relating to a myriad number of topics. Because of this, a huge dataset of tweets can be generated on a daily basis which has the valuable information relating to the public opinion on different topics. Using methods in natural language processing, we can break down the tweets into different public sentiment like positive, neutral and negative. We can also construct more detailed sentiments like strongly positive, weakly, positive etc which captures the sentiment in fine details. The analysis will explain the number of people who are reacting to a topic positively or negatively and this can help understand why the public opinion is inclined in such a manner. With such analysis, it will enable the companies to fix their products to better suit the majority of the public, help politicians take correct stance on a sensitive topic and so on.

In other words, we can describe the sentiment analysis as a part of computational linguistics which focuses on finding the polarity of an opinion/comment or in this case, a tweet. It can capture different emotions exhibited by the person by analysing the structure of the words. There are various tools present in the Natural Language Processing which enables us to input the text in ”readable” format to various algorithms. This allows us to use the text in various machine learning tasks. The sentiment analysis is an example of text classification and the tweet extraction is same as keyword extraction.

2 Overview of Neural Network Methods

The complex architecture of the neural networks enables us to evaluate the relationship between the input text and the output analysis. We have different architectures for the neural networks which can be employed for a certain task but we are focusing on the popular architectures (which are listed below) for our analysis. Neural network architectures can be used for solving various problems in machine learning and in this report we focus on the case of multi-classification problem.

2.1 Components of a neural network

The artificial neural networks are composed of several nodes called neurons which behave similar to the neurons in the human brain. We have several layers of neurons which are connected to each other to make a network. The first layer is called the input layer and the last layer is called the output layer. The layers between the input and output layers are called hidden layers.

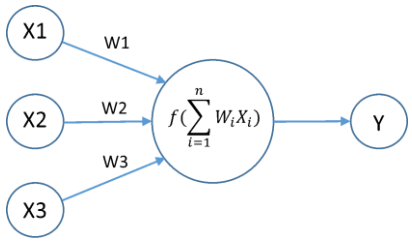

Figure 1: Neuron illustration

Figure 1 shows how a single neuron (in the hidden layer) works in the network. The neuron has input from various neurons in the input layer with weights attached to those values, which are summed up before applying an activation function. Then, the output of this neuron becomes the input to the next layer. The output of a neuron is determined by the activation function.

There are different activation functions like binary step function (which activates the neuron only if the value is above a certain threshold), linear activation function (the output is proportional to the input) and non-linear activation functions. Non-linear activation functions are widely used in the neural network architectures. Some of them are listed below:

1. Sigmoid
2. Tanh
3. ReLU (Rectified Linear Unit)
4. Selu
5. Leaky ReLU
6. Softmax
7. elu (Exponential Linear Unit)

A loss function is used to measure the difference between the model prediction and the actual value. Our aim is to decrease the loss as much as possible in order to get better predictions that are close to the actual value. During the training, the neural network tries to find the weights which reduce the loss function as a whole. The weights are modified by using optimization technique to decide by how much value a weight should be changed. The weights are calculated until we run out of the epochs or the algorithm converges to a minimum loss. There are several optimization techniques that can be used with neutral networks, some of them are shown below:

1. Stochastic Gradient Descent (SGD)

2. Adam
3. RMSprop
4. AdaDelta

2.2 Multilayer Perceptron (MLP)
This is a very basic neural network architecture which consists of multiple hidden layers and it is a type of feed forward network where all layers are fully connected.

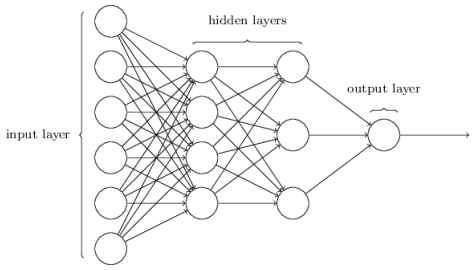

Figure 2: Multilayer perceptron

As shown in figure 2, MLP consists of an input layer where the data is fed into the neural network. We can have a series of hidden layers with each layer having a set of neurons with an activation function. The output layer shows the interpretation of the model with the input data. If it is a case of a multi label problem, then the output neurons should be same as the number of unique labels

2.3 Convolutional neural network (CNN)

Initially convolutional neural networks are developed for image processing and image classification but they can be employed on text classification tasks also. The network uses convolution and pooling as two main operations which function as feature extractors for the model. The output from convolution and pooling is connected to a fully connected multilayer perceptron. The final output layer of CNN is similar to that of an MLP.
Convolution is a process by which we select a filter or a kernel of a predefined size and move this filter along the text values (the text in the sentence are converted into number using an embedding technique which is explained in the following sections) and multiply the values corresponding to the filter values. Then, all the values in the filter are added to make it as the first feature in the result array. The filter is moved along the text values depending on the stride length (it determines how many steps the filter can take everytime it moves). The result array obtained is called as the convoluted feature. Every time the filter is moving along the text values, it is picking up the ngrams (as shown in figure 4) and size of the filter and ngrams obtained is important for the text classification output.
Pooling is a technique used to downsample the convoluted features. We use a similar filter but the operation performed using the filter is different than that of convolution. Different operations can be performed like max-pooling (picking the max value from the filter), average-pooling etc depending on the task. The main purpose of the pooling technique is to reduce the size of the convoluted feature. Padding is used if we want to keep the size of the array same as the input, then ”same padding” makes sure that the size of the array after convolution process is the same.

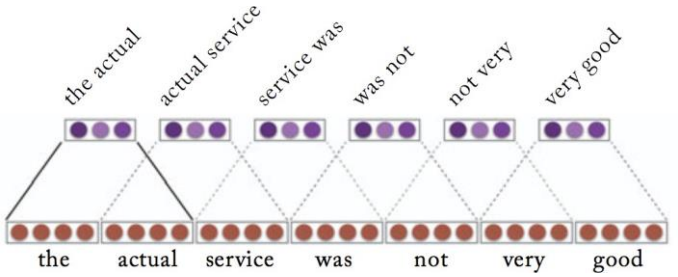

Figure 3: 1D convolution for text

Figure 4 shows the convolution process for the sentence in 1 dimension. Here, the filter is chosen with the size 2 so the words captured are of length 2 (in the top layer). The input text is represented as a vector embedding of 4 dimensions and after the convolution, the dimension is reduced to 3.

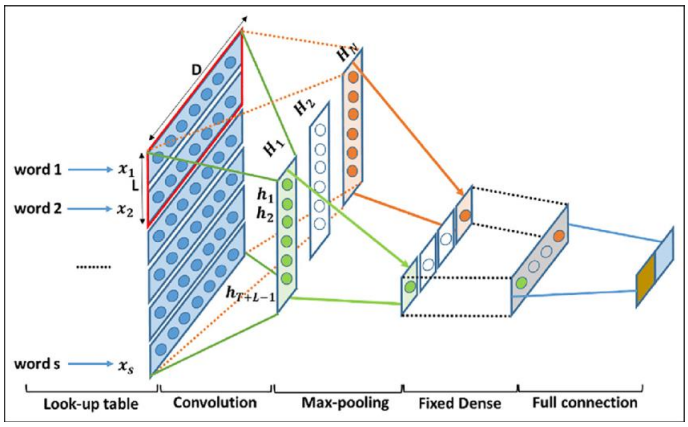

Figure 4: CNN for text classification

2.4 Recurrent neural network (RNN)

Recurrent neural networks are recurrent for a certain number of timesteps and work in a way such that the input from the previous time step is also added to the current input.

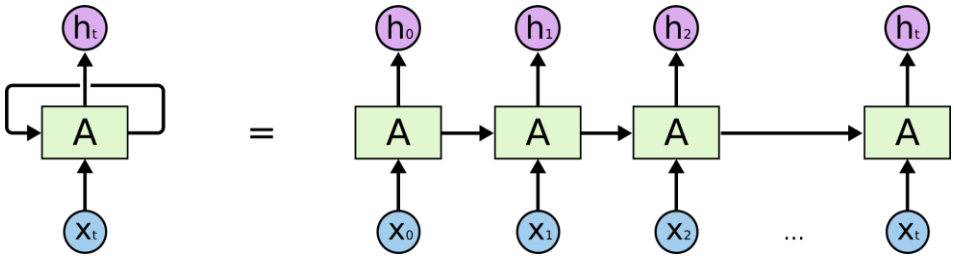

Figure 5: RNN illustration

Figure 5 shows how the input is fed into the network in a loop. This way the network memorizes the sequence of the words in the sentence. This method is useful for preserving the sentence structure where the the context of the words is taken into account while training. During back propagation through time, we face the problem of vanishing gradient where the gradient becomes smaller and smaller which finally has very less effect in modifying the weights. This problem is fixed with the LSTM network.

2.5 Long short-term memory (LSTM)

Standard RNNs were unable to learn long-term dependencies as the gap between two time steps becomes large. To address this issue, LSTM was first introduced in and reemerged as a successful architecture since Ilya et al [9] obtained remarkable performance in statistical machine translation. Many researchers have done lot of work in publishing the different variants of LSTM which were proposed, we have adopted the standard architecture [10] from this renowned work.

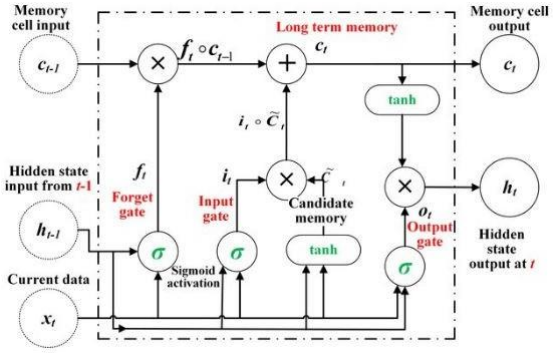

Figure 6: LSTM cell illustration

The LSTM architecture has a range of repeated modules for each time step as in a standard RNN. In Figure 6 at each time step, the output of the module is controlled by a set of gates in Rd as a function of the old hidden state ht1 and the input at the current time step xt: the forget gate ft, the input gate it, and the output gate ot. These gates collectively decide how to update the current memory cell ct and the current hidden state ht.

3 Overview of Natural Language Processing methods

In this section, we explain the various techniques used in natural language processing which helpsfor tasks like text classification. Text classification is a task where our objective is to classify a given text into different classes.

3.1 Text Preprocessing

Text preprocessing is a task where we convert the text into simple format that is more convenient toprocess. It involves removing unwanted words and characters, modifying some words etc. Althoughwe can use the text without preprocessing for the purpose of classification, but studies have shown that the algorithm perform bettwe with some forms of simple preprocessing like lower casing the text and tokenization. [15]. For our experimentation, we used the following preprocessing techniques:

1. Lower casing the entire text
2. Removing URLs and emails which are not helpful when doing text classification
3. Removing numbers
4. Removing white spaces
5. Removing punctuations and special characters

There are more advanced methods of preprocessing like replacing the contractions with full form(Ex. the word ”couldn’t” is replaced with ”could not”). But we have not employed such methodsin our experiments.

3.2 Text Vectorization

The input to the neural networks should be in numerical format. Hence, the text is converted into numerical representation and this process is called text vectorization. There are some methods whichhelp us with text vectorization and they are discussed below:

1. Bag of Words:
   This is a simple and straightforward method for numerical representation of text. Firstly, we build a dictionary of all the words present in the given sentences and for each sentence we generate a vector (which is same size as the number of words in the dictionary) where if the words present in the sentence are denoted by 1 and 0 otherwise. This technique is not useful inthe case of large text because of the high dimensionality of the vector generated. The empiricalperformance of the bag of words is explained in this paper [18].
2. Word embeddings:
   Word embedding is a method where each word in a sentence is mapped to a vector of *d* dimen- sions such that the words which have close semantic similarity are represented closer to each other in the latent space. This is an advanced technique than the bag of words which carries more context of the sentence in a low dimensional space.

   Embedding layer is a method by which we can find the word embeddings while the neural network is being trained. The text has to be input in one-hot vector format where a numberis assigned to every word in the sentence (similar to bag of words).

4 Experiments

4.1 Dataset

The dataset is taken from the kaggle competition ”Tweet sentiment extraction” [6]. It consists of27481 tweets which are denoted as ”Full text” for the sake of our experiments. It also consists a col- umn having text that strongly supports the sentiment of the tweet. This part is denoted as ”Selectedtext” in our experiments. Finally, every tweet is labeled with a sentiment of being positive, nega-tive or neutral. Our analysis is divided into 3 cases which are mentioned in the following subsections.

4.1.1 Exploratory Data Analysis

As explained earlier the dataset is labeled with different sentiments. In this section the distributionof the different sentiment classes is shown in the figure 10

Figure 7: Sentiment Labels in Training and Rest of the Data

The dataset is split into train, validation and test sets. The train set is used in the training phase of the neural network (70% of the data is included in the train set and the rest 30% is split into validation and test set equally) and validation set is used to tune the hyperparameters. Finally, the test set is used to analyse the performance of the final model as shown in figure 11

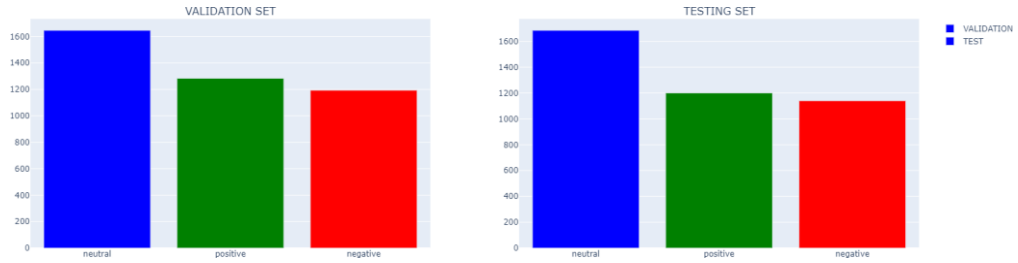

Figure 8: Sentiment Labels in Validation and Test Set

The performance metrics used are accuracy score and F1 scores. Accuracy scores are importantwhen we are looking for true positives and true negatives while, F1 scores are important in the caseof false positives and false negatives. Also, F1 scores become important in the case of imbalanced dataset. Our tweet dataset is slightly imbalanced but not much with neutral class is dominating over other classes in our dataset. We verified if this slight class imbalancy in the dataset affects theaccuracy and F1 scores obtained during sentiment analysis.

For our experiments, it is also important to understand the distribution of total number of words present in every tweet in the entire dataset. The graph below shows the distribution for the numberof words in the tweets separately for each sentiment: (figure 12)

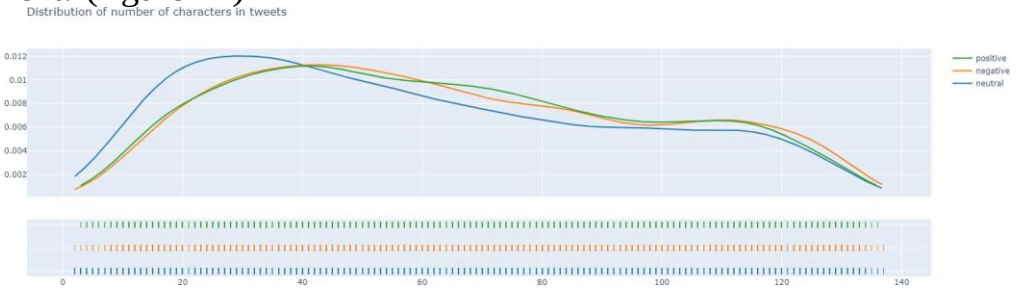

Figure 9: Distribution of number of Characters in tweets

We observed that the maximum tweet length is 32 and minimum is 1. We have taken this into account while performing the experiments.

Basically we are dealing with the textual data in for classification of the sentiment labels. We need to understand the most frequent words that help in text classification. This can be done in the form of word cloud representation. Word clouds are important to extract the top frequent words in a dataset. In the figure 13 we picked 3 random tweets (one for each sentiment) from the dataset andillustrated their word cloud representations.

Figure 10: Word Cloud Representation on Full Text

4.2 Experimental setup

Following 4 algorithms are used for performing the experiment and finding the best algorithm out of them:

1. Multilayer Perceptron (MLP)
2. Convolutional neural network (CNN)
3. Recurrent neural network (RNN)
4. Long Short-Term Memory (LSTM)

4.3 Results

	Acc. Score	F1 score	Time(seconds)
MLP	71.61	71.50	41.98
CNN	69.27	69.25	89.01
RNN	37.64	31.14	1870.85
LSTM	40.88	23.07	183.01

Performance metrics for Sentiment analysis

5 Optimal hyperparameters

The optimal hyperparameters are found for every neural network method which is shown in the following table. The hyperparameters used for pretrained version and the embedding layer version are almost same with 1 or 2 changes which are shown in the table. Most of the algorithms perform well with RMSprop as optimizer and with low batch sizes 16 and 8. (The accuracy scores shown here

are for Case 1 pretrained version)

	Activation	Optimizer	Batch size	Epochs	Acc. Scores
MLP	Relu	RMSprop	32	5	71.61
CNN	elu	RMSprop	8	5	69.27
RNN	Tanh	RMSprop	8	5	37.64
LSTM	Tanh	adam	16	5	40.88

Optimal hyperparameters

6 Discussions and Future work

The sentiment analysis gave us good results and the performance of the neural network models wereas expected. Transformers gave the best results out of all the models. The neural networks were able to perform better on the selected text (which is a shorter form of the tweet containing the wordsthat best describe the sentiment). By this, one idea would be to extract the important words from the tweets using tweet extraction methods and then use those words as input for sentiment analysis.

For the future work, we plan to:

1. Explore more models like RCNN, CNN-LSTM which are mix of convolution and recurrent networks
2. Experiment with transformer models and also other state of the art text classification models.

7 Conclusion

From our experiments, we conclude that:

1. MLP give the best performance of all the neural network models
2. We observed that running the model of more number of epochs give good training scores andit keeps on increasing with increase in epochs but after a certain number of epochs (like 10 epochs) the validation scores are effected. This can mean that the model is starting to overfit by running it more number of times.
3. For the tweet dataset, 8 or 16 batch size is optimal for better scores at the test time. As we increase the batch size, the training accuracy drops, which can be a case of adding bias to the model or the model is underfitting the data points. Most of the neural network methods perform better on low batch sizes.
4. RMSprop optimizer was the best among the optimizers we tested (relu, elu, selu, Adadelta and Adam) which worked well with the tweet dataset.

8 References

[1] Bert tweet extraction. https://www.kaggle.com/yutanakamura/dear-pytorch-lovers-bert-

transformers-lightning.

[2] Bert, xlnet and roberta code. https://github.com/jinudaniel/ amazon-fine-food-reviews.

[3] Convolutional neural networks for text classification. http://www.davidsbatista.net/blog/ 2018/03/31/SentenceClassificationConvNets/.

[4] Deep-dive into bidirectional lstm. https://www.i2tutorials.com/ deep-dive-into-bidirectional-lstm/.

[5] Multilayer perceptron. http://naviglinlp.blogspot.com/2015/05/ lecture-8-neural-network-word.html.

[6] Tweet sentiment extraction dataset. https://www.kaggle.com/c/ tweet-sentiment-extraction/data.

[7] Understanding rnn. https://medium.com/x8-the-ai-community/ understanding-recurrent-neural-networks-in-6-minutes-967ab51b94fe.

[8] J. Devlin, M. Chang, K. Lee, and K. Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805, 2018.

[9] A. K. I. S. Geoffrey E Hinton, Nitish Srivastava and R. R. Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. 2012.

[10] S. Hochreiter and J. Schmidhuber. Long short-term memory. 143:1735–1780, 1997.

[11] A. Jacovi, O. S. Shalom, and Y. Goldberg. Understanding convolutional neural networks fortext classification. *arXiv preprint arXiv:1809.08037*, 2018.