Twitter Sentiment Classification

Pratik Pundlik Sayanekar
Student No: 20200161
Department of Computer Science
University College Dublin
pratik.sayanekar@ucdconnect.ie

Abstract

In this paper, we propose several approaches to do sentimental analysis on Twitter data. This research is mainly focused on the implementation of deep learning algorithms in Keras including textual data preprocessing techniques such as word embedding and tokenization. Detailed explanations starting from preprocessing strategies to a theoretical approach to understanding deep neural networks such as CNN, LSTM, and Bidirectional LSTM have been discussed in this research. Finally, an appropriate combination of LSTM, BiLSTM, and CNN model along with Word2Vec embedding resulted as an efficient approach to do sentiment analysis of Twitter data.

1 Introduction

In the past decade, social media have become one of the valuable sources of data in various forms as text, images, videos, and voices. Twitter is one of the social media platforms famous for microblogging where user's posts and interactions are known as tweets. Every day millions of people use this platform to posts tonnes of tweets about what they are doing, what is happening around them etc. This explosion of data has led researchers to develop new approaches in Machine Learning and Deep Learning to analyze this data. Nowadays with the advancement of higher computational power and astounding research in Machine Learning and Deep Learning is helping one to create robust algorithms to fetch insights from this huge amount of data.

With increasing popularity of twitter, the sentiment analysis of tweets becoming more attractive. Sentiment Analysis is branch of natural language processing that analyses text to identify, extract and quantify the subjective information. For twitter sentiment analysis we have the tweets classified into 'Happy', 'Angry', and 'Disappointed'. So the task here is to analyze the text of tweets and precisely predicting the target class. This is a multiclass classification problem. In this paper, we discuss the approaches that can be used to do a sentimental analysis of tweets effectively.

2 Related Work

Research in Deep Learning and Machine Learning encouraged by high computational power has had great success in the past few years. Today language modeling has become a trending research topic in NLP that solves numerous problems such as sentiment analysis, text summarization, text auto-completion, and many more. This section presents a brief review of the work done in sentiment analysis and prediction particularly in regards to Twitter data.

2.1 Target-dependent Twitter Sentiment Classification

This paper compares the traditional state-of-the-art approach of target-independent and target-dependent sentiment analysis. The target-independent approach may assign irrelevant sentiments to

COMP47650 Deep Learning - Final Project Report, (UCD 2020-2021), Dublin, Ireland.

the given target. Moreover, sentiment classification is done considering only the text of tweets; they ignore its context (i.e., related tweets and retweets). The authors also proposed strategies to improve target-dependent Twitter sentiment classification such as incorporating target-dependent features and considering related tweets. [2]

2.2 Learning Sentiment-Specific Word Embedding for Twitter Sentiment Classification

This paper illustrates the importance of word embedding for twitter sentiment classification. Most of the word embedding algorithms focuses on the syntactic context of words over sentiment. For example, good and bad are of similar syntactic context but the sentiment polarity is the extreme opposite. The authors of this paper have addressed this issue by learning sentiment-specific word embedding that encodes by considering sentiment information of continuos words. [3]

2.3 A Combined CNN and LSTM Model for Arabic Sentiment Analysis

In this paper, the authors investigated the benefits of integrating CNN's capability of selecting good features and LSTM's ability to learn sequential data and achieved improved accuracy for Arabic sentiment analysis on different datasets. This approach has been validated in areas such as image classification, voice recognition, language translation, and other natural Language Processing (NLP) tasks. The authors have also discussed the challenges in Sentiment classification for short text messages from Twitter. [4]

3 Experimental Setup

3.1 Dataset

The dataset is available on kaggle as mentioned in the link below.

`https://www.kaggle.com/kosweet/cleaned-emotion-extraction-dataset-from-twitter`

The dataset contains 3 columns i.e. 'Emotion' as target variable (sentiment), 'Content' as preprocessed tweets, and 'Original Content' as raw tweets. The dataset contains 9,12,495 tweets divided into three classes of emotions as **34% of Disappointed** and **33% of Happy** and the remaining **33% of Angry** sentiments. We can infer that this is a balanced dataset and multiclass classification problem.

3.2 Preprocessing

Preprocessing is an iterative approach and the strategies vary as per data. This section discusses the extraction of textual features and several text cleaning approaches.

3.2.1 Text Pre-processing: Manual Approach

Understanding the data is an important step in preprocessing as this determines the strategy to preprocess. Visualizing the textual data for maximum occurrences of words, punctuation, and emojis of each class directs the further plan of action. For the provided dataset, in the case of Emojis, there is a pattern in usage that has been observed with disappointed, happy, and angry tweets, so keeping emojis proved to be the better choice. As per applied preprocessing, emojis can be identified with the '_' (underscore) character in between words.

As the dataset is related to tweets and it may contain URLs, hashtags, emojis, smileys, and some specific words such as username and Twitter-related acronyms (RT, FAV), etc. All these words except Hashtags and Emojis are unnecessary to predict the correct sentiments. So in a manual approach, all such words are handled using regular expression and string utilities provided in Python. Few patterns are observed in the dataset such as @ characters in a username, so as per the inconsistency in data and semantics, all such words were filtered out. Removing single-character words is also one of the proven strategies that help in minimizing the word count and ultimately dimension of the embedding matrix. Textual features of preprocessed data, such as distribution of word count *(recorded as 122)* are one of the prime features that are used to decide the length of padding and dimensions of the embedding matrix.

3.2.2 Text Pre-processing using Built-in Libraries

Python has several built-in libraries specific for text processing and twitter as mentioned below,

- **tweet-preprocessor**
  Tweet preprocessor is an efficient library used for cleaning, tokenization, and parsing features such as URLs, Hashtags, Mentions, Reserved words (RT, FAV), Emojis, Smileys. This can be done manually as well using regular expression and string utilities but there are few shortcomings for example RT character and extra spaces that need to be handled explicitly. Such robust libraries are specially designed to preprocess tweets. [6]
- **gensim**
  gensim.parsing.preprocessing is another efficient library to handle stop words. The main advantage is that one can apply this to directly raw text before tokenization as this is taken care of by gensim to make it more robust.[7]
- **WordNetLemmatizer**
  Stemming and Lemmatization are similar text cleaning strategies that convert the words into their base form. The main difference is lemmatization considers the context of text while stemming just removes the last few characters. This is implemented using NLTK's WordNetLemmatizer with POS tags for verbs and nouns, due to unfavourable results this approach has been commented in implementation.

Appropriate usage of both manual text preprocessing and built-in libraries gives better results. Preprocessing improves the quality of data exponentially as it reduces the corpus size dramatically. This has been demonstrated in the *'Data_Preprocessing.ipynb'* python notebook in detail.

3.3 Models

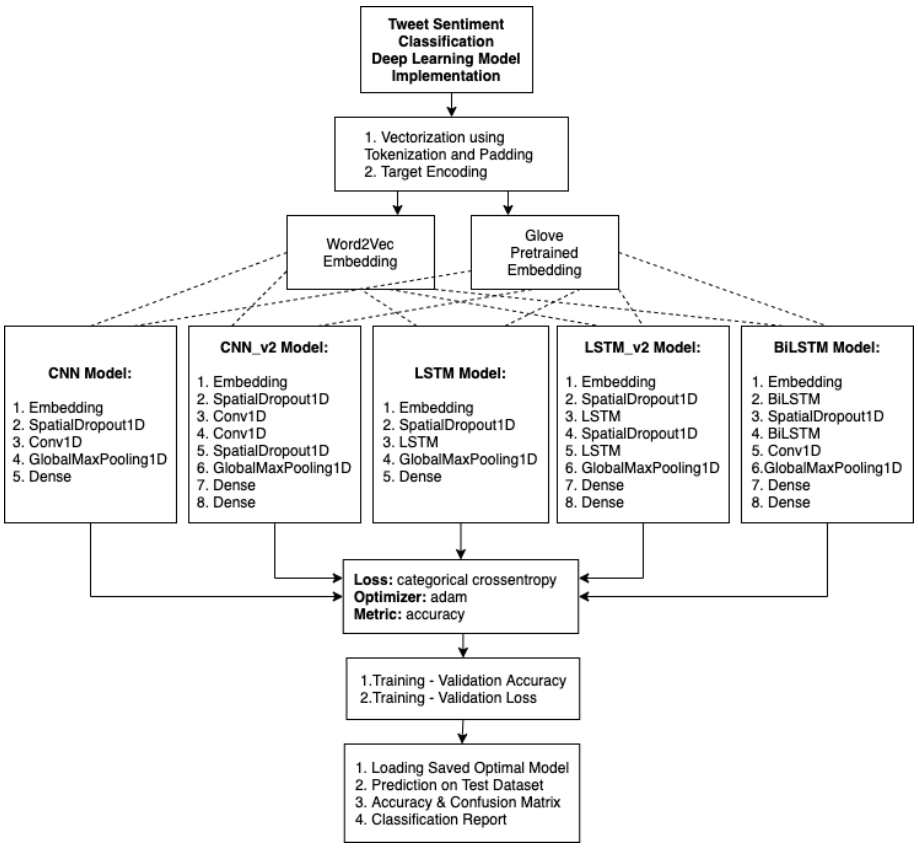

Figure 1: Deep Learning Model Implementation Overview

3.4 Proposed Algorithm

3.4.1 Tokenization Padding

Tokenization is a part of preprocessing but this is one of the primary steps which transform the textual input into a vectorized version that can be provided to any learning algorithm directly or can be transformed into a numerical matrix. This vectorization is achieved by converting each text into a sequence of integers by considering word count or techniques such as term frequency and inverse document frequency. This can be implemented in python using a dedicated API provided by TensorFlow as 'tf.keras.preprocessing.text.Tokenizer'.[11] The integer sequence is further padded as per the maximum length of characters in each tweet to maintain the same dimension of vector.[12]

3.4.2 Word Embedding: Word2Vec and Glove

Word2Vec is a shallow deep learning model used to create word embeddings. It is based on a linear relationship between the words. for example, king - man + women = Queen. We have used this model along with the tokenizer provided by Keras to filter out unique words.

The gloVe is an unsupervised learning algorithm by Stanford, can be used as an alternative to Word2Vec for obtaining vector representations for words. This algorithm is trained on a wide range of global datasets and provides vectors of multiple dimensions as 50, 100, 150, 200, 300, etc. For the Twitter dataset, after preprocessing the max word count is 122, so we have integrated a trained gloVe with a 100-dimensional vector.

The embedding matrix is generated by integrating Word2Vec or Glove vector with a tokenized version of train data. This matrix is further passed to the embedding layer of a neural network as weights. The gloVe is recommended for small datasets as it helps to generalize quickly but as we have millions of tweets Word2Vec model is also giving impressive results.

3.4.3 Combination of BiLSTM + CNN

BiLSTM is another variant of LSTM and the specialized RNN model. Like LSTM, it also has the ability to store information of recurrent layers, moreover, it feeds the data to the algorithms in both directional once from start to end as well as from end to start. So the output is 2 times LSTM that doubles the dimensional as described in the model summary part of a notebook.

CNN is known for sequential data as uses a convolution strategy to learn from data and end up learning complex pattern in the data. The size of the kernel that is 5(*minimum word count in tweets*), decides the length of the sequence resulted in a convoluted matrix.

BiLSTM + CNN gives the advantage bidirectional Information of recurrent layers and one-dimensional sequential convolution strategy resulted impressively with an accuracy of 90%. The detailed implementation is provided in the BiLSTM.py file.

3.5 Baseline Approach

3.5.1 CNN Model

1-dimensional convolution layer takes the additional hyperparameters such as the number of filters and the number of words as kernel size to be convoluted along with provided batch size, and the number of epochs. The architecture of this model implemented using Keras demonstrates series of layers including the first embedding layer that accepts the tokenized input and embeds it into low-dimensional vectors described as max_len = 122. For gloVe, the max_len is 100 as we are using a 100-dimensional gloVe vectorizer. The dropout layer regularizes the model. Dropout is trained at 0.3, 0.5, and 0.8 as disabling rates. The trends of validation and training loss and accuracy show no overfitting and the model giving similar results on the test dataset. The CNN layer performs convolutions over input as specified filters and word count as max_len. The Max-pooling layer converts the result of the convolutional layer into a long feature vector, and the dense layer with softmax as activation function is used to output the results in the form of probabilities into the specified number of classes as per the input, as this is the multiclass problem there are 3 classes to be predicted.

3.5.2 LSTM model

The main challenge of RNN, to store information of recurrent layers is solved by LSTM with backpropagation. LSTM i.e. Long short-term memory is a special RNN that is processing the information of recurrent layers in one direction with gradient descent manner.The architecture of this model implemented using Keras demonstrates series of layers including the first embedding layer same as the CNN baseline model. then dropout layer as regularization. The LSTM learns the sequence data. As this is a multiclass classification problem, softmax is used as the activation function in the final layer to output the result as probabilities.

The detailed implementation in *Deep_Learning_models.ipynb* notebooks versions and respective classes python classes.

3.6 Hyper Parameter Tuning

Hyperparameter tuning is one of the crucial steps in building efficient deep neural networks. Basically, it selects the best parameters among the list of parameter choices. In Keras, this could be achieved by implementing KerasClassifier to create a deep learning model(Neural Network) and RandomizedSearchCV to iterate through the specified list of parameters. Due to lack of computational power, this is not implemented in the provided notebook, however, hyper-parameter tuning is achieved manually by applying the following strategies.

3.6.1 Dropout Regularization

Dropout is considered a computationally inexpensive regularization strategy. It is implemented using the SpatialDropout1D layer provided by Keras. Rather than dropping individual elements, it drops the entire 1D feature map. Dropout value affects the validation and training loss convergence. Dropout does not regularize activations for strongly correlated frames. SpatialDropout1D supports independence between feature maps and it is recommended for sequential data. [13]

3.6.2 Early Stopping

Early stopping is one of the strategies used to avoid overfitting. It monitors the generalization error and stops the iterative approach of learning if generalization error starts to degrade. This can be achieved by the callback functionality provided in Keras. Callbacks allow us to perform any operations on various stages of learning. ReduceLROnPlateau is also one of the similar operations where it monitors and reduces the learning rate when a specified metric has stopped improving. In the implementation, we have considered a validation loss as an evaluation metric for both early stoppings and reduce the learning rate on the plateau.

3.6.3 Optimization Strategies: Adam

Adam optimization is a stochastic gradient descent method that combination of both adaptive gradient descent with momentum algorithm (AdaGrad) and root mean square propagation (RMSProp), It handles sparse gradient on noisy data. Adam optimizer is integrated with default parameters in the notebook. According to Kingma et al., 2014, the method is "computationally efficient, has little memory requirement, invariant to diagonal rescaling of gradients, and is well suited for problems that are large in terms of data/parameters". [14][15]

4 Results

Accuracy: As this is a multiclass classification problem, the accuracy is the accumulation of all three classes, i.e 'Happy', 'Angry', and 'Disappointed'. The accuracy of the 'Angry' class is comparatively low for all models. The dataset is closely balanced but the training and testing examples of 'Angry' class are less as compared to other two classes. This depicts that the model can easily generalize 'Happy' and 'Disappointed' tweets but in case of Angry most tweets wrongly predicted tweets are classified as 'disappointed' which illustrates that model accuracy can be improved by providing some additional distinct features of the 'angry' class. Accuracy and F1 score of each classes are as described below.

	Dropout	Angry	Disappointed	Happy	Accuracy
CNN	0.3	0.87	0.87	0.96	0.90
	0.5	0.87	0.88	0.96	0.90
	0.8	0.86	0.87	0.96	0.89
CNN v2	0.3	0.87	0.88	0.96	0.90
	0.5	0.87	0.87	0.96	0.90
	0.8	0.83	0.83	0.93	0.86
LSTM	0.3	0.87	0.88	0.96	0.90
	0.5	0.87	0.88	0.96	0.90
	0.8	0.85	0.85	0.95	0.88
LSTM v2	0.3	0.87	0.88	0.96	0.90
	0.5	0.87	0.87	0.96	0.90
	0.8	0.76	0.75	0.91	0.81
BiLSTM + CNN	0.3	0.87	0.88	0.96	0.90
	0.5	0.87	0.88	0.96	0.90
	0.8	0.87	0.88	0.96	0.90

Table 1: F1 Score of target classes and overall Accuracy.

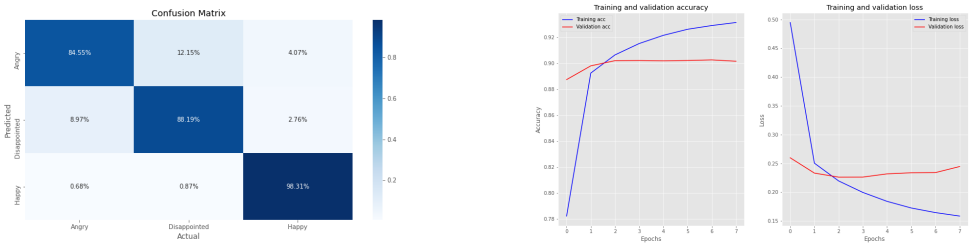

Figure 2: Performance of BiLSTM Network at dropout 0.3 with gloVe vectorization

Word2Vec Vs Pretrained word embedding: Pre-trained word embedding is also a better option to generate an embedding matrix that can be passed to the embedding layer as weights. This is recommended for a smaller dataset as pre-trained embedding helps the model to generalize very easily but as the twitter dataset is large enough, this was avoided initially but in the later stage of exploration, we analyzed how pre-trained embedding improves the accuracy. Above figure shows the validation and training loss trend along with confusion matrix when trained at dropout value as 0.3.

5 Future Work

The proposed model, a combination of BiLSTM and CNN performs better as compared to simple CNN and LSTM models. In terms of future work, it would be exciting to explore the similar task on which this model performs well. The bidirectional approach of context conditioning used in BiLSTM is becoming state-of-the-art in various continuous language modeling tasks. With the limited access of Google Collab and the unavailability of GPU, we couldn't explore much hyperparameter tuning. Checking the model performance with several values of parameters would be a great learning experience and help to understand the behavior of different types of layers used in models more thoroughly. This project and research helped to understand that language modeling is evolving domain and various language representation models are coming into the market. BERT, Bidirectional Encoder Representations from a transformer is also one of the novel models which can be used for a wide range of language modeling tasks. Recently, Google's XLNet outperformed BERT on major tasks of NLP such as text classification and sentiment analysis as well as advance NLP tasks. XLNet's permutation language modeling technique is proven to be efficient to generate information in forward and backward direction just like BiLSTM. This paper discusses basic models and as research goes forward, it would be an exciting opportunity to implement such state-of-the-art models for twitter dataset.

6 Conclusion

In this paper, we propose the novel approach of multiclass tweet sentiment classification that achieves state-of-art performance and 90% accuracy. From preprocessing to neural network modeling, several approaches, their pros, and cons are discussed in this paper in detail. For preprocessing tweet preprocessor and gensim's stopword removal APIs are proven to be effective along with some manual cleaning as per requirement. Textual data after applying preprocessing strategies described in this paper is as efficient as provided cleaned data. Neural network modeling is implemented in Keras. The sequential model and diverse availability of layers in Keras infrastructure are studied thoroughly. Considering CNN and LSTM as baseline architecture, one more complex network of each is designed which also improves the accuracy to some extent. This tremendously helped to understand Keras's architecture and the contribution of each layer. Improvisation in technique such as embedding is achieved by implementing vectorization using Word2Vec or pre-trained models such as Glove makes the model more efficient. A bidirectional version of LSTM i.e BiLSTM along with CNN is the final proposed model. The accuracy of all models is comparatively the same with a difference of 1% to 2%, execution time wise, BiLSTM and LSTM take more time to train as compared to CNN.

Acknowledgments

We here acknowledge that this project has been completed by the author on his own and all the references to the research papers and online forums have been thoroughly mentioned. All the citations related to implementation such as several websites and Kaggle notebooks have been provided. I would like to thank Professors and all TAs for their continuous guidance throughout this project.

References

[1] Liu, B. (2012) Sentiment Analysis and Opinion Mining. Synthesis Lectures on Human Language Technologies. Morgan Claypool, San Rafael

[2] Jiang, L., Yu, M., Zhou, M., Liu, X. and Zhao, T. Jiang, L. et al. (2011) "Target-dependent Twitter Sentiment Classification", ACL Anthology. *Available at: https://www.aclweb.org/anthology/P11-1016/*

[3] Tang, D., Wei, F., Yang, N., Zhou, M., Liu, T. and Qin, B. Tang, D. et al. (2014) "Learning Sentiment-Specific Word Embedding for Twitter Sentiment Classification", *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. doi: 10.3115/v1/p14-1146.

[4] Alayba, A. M., Palade, V., England, M. and Iqbal, R. Alayba, A. et al. (2018) "A Combined CNN and LSTM Model for Arabic Sentiment Analysis", *Lecture Notes in Computer Science, pp. 179-191*. doi: 10.1007/978-3-319-99740-7_12.

[5] https://www.kaggle.com/kosweet/cleaned-emotion-extraction-dataset-from-twitter

[6] https://pypi.org/project/tweet-preprocessor/

[7] https://radimrehurek.com/gensim/parsing/preprocessing.html

[8] https://github.com/RaRe-Technologies/gensim/wiki/Migrating-from-Gensim-3.x-to-4

[9] Hochreiter, S. and Schmidhuber, J. Hochreiter, S. and Schmidhuber, J. (1997) "Long Short-Term Memory", Neural Computation, 9(8), pp. 1735-1780. doi: 10.1162/neco.1997.9.8.1735.

[10] https://colah.github.io/posts/2015-08-Understanding-LSTMs/

[11] https://www.tensorflow.org/api_docs/python/tf/keras/preprocessing/text/Tokenizer

[12] https://www.tensorflow.org/api_docs/python/tf/keras/preprocessing/sequence/pad_sequences

[13] https://keras.io/api/layers/regularization_layers/spatial_dropout1d/

[14] Kingma, D. P. and Ba, J. Kingma, D. and Ba, J. (2014) Adam: A Method for Stochastic Optimization

[15] https://www.tensorflow.org/api_docs/python/tf/keras/optimizers/Adam

[16] https://nlp.stanford.edu/projects/glove/

[17] Devlin, J., Chang, M., Lee, K. and Toutanova, K. Devlin, J. et al. (2018) BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding,