

Text bases Sentiment Analysis

Rustom Bhesania¹, Aryan Mehta², Deepam Maurya³ and Sumita Nainan⁴

¹⁻³Student, NMIMS University/Electronics and Telecommunication Engineering Department, Mumbai, India
Email: {rustombhesania@rediffmail.com, am499702@gmail.com, maurya.deepam@gmail.com}

⁴Assistant Professor, NMIMS University/Electronics and Telecommunication Engineering Department, Mumbai, India
Email: sumitaa.nainan@gmail.com

Abstract— Sentiment Analysis is one of the most growing fields of work in Natural Language Processing. Sentiment The analysis is used in determining the sentiment of any given text which can be positive, negative, and neutral. The proposed work includes the selection of databases, data acquisition, model building, and deploying the model. The databases selected were Airline tweets and Google Play App Reviews. Then the databases were pre-processed by techniques such as tokenization, lemmatization, and TF-IDF. Later, the pre-processed data are fed into the model. The LSTM (Long Short Term Memory) and BERT (Bidirectional Encoder Representations from Transformers) model was implemented. The LSTM model was trained and tested on Airline Tweets Sentiment Dataset whereas the BERT model was trained on Google Play App Reviews dataset and tested on a random user input text. Later, it was known the BERT model was performing better than the LSTM Model, so this model was selected and then deployed to form the Web Application programming interface (API).

Index Terms— tokenization, lemmatization, TF-IDF, LSTM, BERT and API.

I. INTRODUCTION

Sentiment analysis is a process in which a trained model understands the sentiment of the input text. The sentiment refers to the emotion of the text provided by the user. The most common sentiments are negative, neutral and positive. According to the requirement, the model can be trained to give sentiments based on emotions such as sad, happy, angry, etc. This field of area can have various applications where the customer's opinion can be taken, and the company can improve the product using these views. Another application where sentiment analysis can be used when we want to notice the emotions of the person and what is the speaker going through? Is the speaker happy or not? So overall, sentiment analysis has a great future and more work can be done for the same.

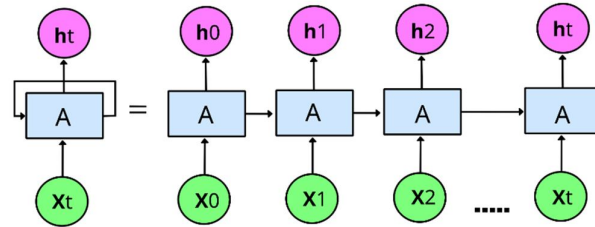
II. METHODOLOGY

A. Databases

1. *Airline Tweets Sentiment Dataset*: The Airline Tweets Sentiment Dataset consisted of 15,000 reviews and 15 columns. The dataset consisted of labeled column where for each sentence the sentiment was already given.
2. *Google Play App Review Dataset*: The dataset consisted of 16,000 reviews approximately with 11 columns. In this dataset, there was no labeled column representing the sentiment for each of the reviews.

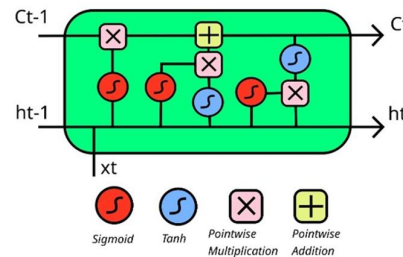
B. RNN

RNN is a Recurrent Neural Network which is made for the basic sequence-to-sequence modelling. RNN is just a set of neural networks with feedback neurons. RNN computes the given input with respect to its timestamp. Ex. If you have sequence of n words, RNN will have n times its internal states. It is difficult to calculate gradient because if the sentence is long the previous state of the neuron is lost (also called long term dependency problem) because of exploding and vanishing gradient.



C. LSTM

Long Short Term Memory (LSTM) are a special kind of RNN which is designed to solve the learning of long-term dependency because it has forget gate, input and output gate, which are nothing but nodes with nonlinearity functions.



1. **Forget Gate:** If there is no correlation between the current and past sequence the forget gate will remove the current sequence. In forget gate nonlinearity (sigmoid) is applied to the current state and the current state, the output of sigmoid ranges between 0-1, if closer to 1 then we keep, if 0 then we forget.
2. **Input gate:** The previous hidden state and the current state are passed to a sigmoid function which results the values between 0-1. Then the same previous hidden and current is passed to tanh function which will transform the values -1 to 1. Then we multiply the sigmoid output with the tanh output so the sigmoid will decide which values are to be kept from the tanh function. This is how we update the cell state.
3. **Output Gate:** The output gate will decide the next hidden states. First, we pass our previous hidden state to a sigmoid function, then the newly modified output cell state from the input gate is again passed through a "tanh" function, then we multiply the output to the sigmoid and the tanh, thus this decides which information is to be carried over to the next hidden state.

D. Problems with LSTM

1. Computation complexity is high because there are multiple uses of sigmoid and tanh functions in a single LSTM cell.
 2. Transfer learning cannot be used because we need to know all the previous states. Therefore, to call a new problem statement we must train a LSTM network from scratch.
 3. LSTM is not truly bidirectional in nature because the sequence is passed from left-right and then right-left
- All these challenges are solved using Transformers.

III. IMPLEMENTATION OF LSTM MODEL

A. Data Exploration

As mentioned earlier, the dataset used for the RNN Model comprised of Sentiment Airline Tweets. So, the dataset consisted of 15 columns and around 15,000 reviews. Out of the 15 columns, only two columns were kept

1) Sentiment 2) Text and rest were discarded. Since the “sentiment” column was a labeled one i.e., it already had the sentiment specified for the given text, a histogram plot was implemented and it was known that there were three sentiment classes: Positive, Negative, and Neutral. The negative class had the most amounts of reviews while the positive class had the least number of reviews.

B. Preprocessing

The data fed to the model should be in such a form that the training of the model should be efficient and understandable. To satisfy the above requirements, the preprocessing is done. The texts had punctuations and tags starting with “@” which were replaced with space. After doing the required process, the texts were then converted to lowercase and the texts were given as the input to the Tokenizer. The tokenizer basically does the task of assigning numbers to each word in each text on a given range. For example, the user would provide a value of 5000, the tokenizer will assign a number to each word from 0 to 5000. Once each word is assigned a number then every text is made equal in length with the help of padding. Padding works by adding zeroes to the end of the text according to a specified length. Thus, after padding, all the texts were made of equal length. Finally, the preprocessed data was ready which can be fed to the model.

C. Creating the model

Several layers made up the LSTM model. The first layer was the embedding layer which was the first hidden layer and required three arguments.

1. *Input dimension*: It is the range or the size of the vocabulary which was set during the preprocessing process while tokenizer was being used.
2. *Output dimension*: This is the size of the vector space in which words will be embedded. It defines the size of the output vectors from this layer for each word. The proposed work utilized the value 256 as the output dimension.
3. *Input length*: This is the length of input texts which was set for a fixed length value by the `pad_sequences` method.

The next layer is the Dropout which basically avoids the overfitting of the model. It basically works by dropping out some units which help in the better transformation of the input. After the dropout layer, there was stacked LSTM architecture consisting of the two LSTM Layers. The First LSTM layer required four arguments:

1. *Output dimension*: The vector space which is defined in the embedding layer is used and for our case 256 will be the output dimension.
2. *Return Sequences*: As the name suggests it returns some hidden units which can be used by the next LSTM layer. This can be ensured when the value set is True for this argument. Also, when the value set is True, it satisfies the condition of the consecutive LSTM layer which is to have only three-dimensional inputs.
3. *Dropout*: As mentioned earlier, it helps to drop some units for the linear transformation of the input units.
4. *Recurrent Dropout*: Fraction of the units to drop for the linear transformation of the recurrent state.

The second LSTM Layer requires three arguments: ‘Output Dimension’, ‘Dropout’, ‘Recurrent Dropout’. In the end, a dense layer is added to the model. The purpose of the dense layer is to operate on the given input (i.e., output from the previous layer is fed to the model) and return the output. For the purposed model, it requires two arguments: ‘Input Dimension’, ‘Activation Function’—The activation function used was SoftMax. Finally, the model was compiled with the following arguments:

Loss: It basically helps in how well the model has learned the dataset. Categorical cross-entropy was used as a loss function for the model.

Optimizer: Consists of the optimizer name which is used during the compiling phase. The Adam optimizer was chosen for the purposed work.

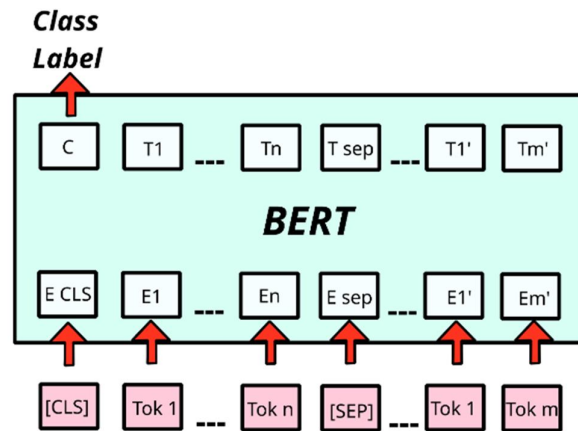
Metrics: A metric is a function that is used to judge the performance of your model. The metric chosen was Accuracy.

D. Training the model

To make it easy for the model to understand the dataset, one hot encoding was used on the sentiment column (i.e., labeled column) for the three classes-Negative, Neutral and Positive. For example: If the sentence is neutral, the one-hot encoded value would be [0,0,1] where 1 represents the neutral class and the rest of the two signify the negative and positive classes, respectively. These values are combined are then combined with the preprocessed data which form the input to the model. Further, the data was split into training and testing purposes. Later the model was trained on the training data with a batch size of the 32 and epoch value of 50, respectively. After running the model for an epoch value of 50, accuracy of 99.3 % was obtained.

E. BERT

A language model for Google AI called BERT uses pre-trained and fine-tuned models to create state of the art models for variety of tasks, this included sentiment analysis, image recognition, document classification, classifying biomedical texts, and many more. BERT stands for Bidirectional encoder representations from Transformers. It is based on Transformers Architecture. It is pre-trained of a larger corpus data from Wikipedia of 2500 million words and 800 million words of Brown Book Corpus. BERT has many subtypes which is specific for each task, suppose for Monolingual Corpora M-BERT pretrained model is used and fine-tuned model of code switching is used. In a regular bidirectional LSTM, the sequence starts from left-right and right-left, so in true sense it is not bidirectional. But in BERT, transformer architecture is used in which the entire sequence is read at once. In transformers there are two mechanism, encoder and decoder. The encoder is used to read the input and decoder is used to decode the information into desired output. BERT is a language model therefore only encoder is used. There are 4 types of BERT pretrained-model: BERT-base-uncased, BERT-large-uncased, BERT-large-cased. These models are pretrained on English Language using masked language modelling (MLM). Masked language modelling takes a sentence and randomly masks 15% of the input and then the model runs through the whole sentence and predicts the masked words. Next Sentence Prediction (NSP), during pretraining the model concatenates the two masked sentences. Sometimes the two sentences are following to each other and sometimes not. The model also predicts if the next sentence is the continuation of the previous sentence or not.



IV. CONCLUSIONS

The goal of the paper was to understand different types of language sequence-sequence models such as RNN, LSTM, Transformers, BERT. We hope that this paper has helped you to understand the basic concepts of the language models.

REFERENCES

- [1] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. (2020). HuggingFace's Transformers: State-of-the-art Natural Language Processing.
- [2] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. (2017). Attention Is All You Need.
- [3] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. (2019). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding.
- [4] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. (2014). Sequence to Sequence Learning with Neural Networks.
- [5] Sherstinsky, A. 2020. Fundamentals of Recurrent Neural Network (RNN) and Long Short-Term Memory (LSTM) network. *Physica D: Nonlinear Phenomena*, 404, p.132306.