Sentimental Analysis Of Tweets By- TEAM SHADY

Asutosh Mohapatra [1], Jaya Sridhar NK [2], Aravind Kashyap KR [3], Parshuram Kumar Gupta [4], Naveen Kumar Vaegae [5]
[1] asutosh.mohapatra2017@vitstudent.ac.in, [2] jayasridhar.nk2017@vitstudent.ac.in,
[3] kr.aravindkashyap2017@vitstudent.ac.in, [4] parshuramkumar.gupta2017@vitstudent.ac.in, [5] vegenaveen@vit.ac.in
School of Electronics Engineering, Vellore Institute of Technology, Vellore 632014, India

Abstract:

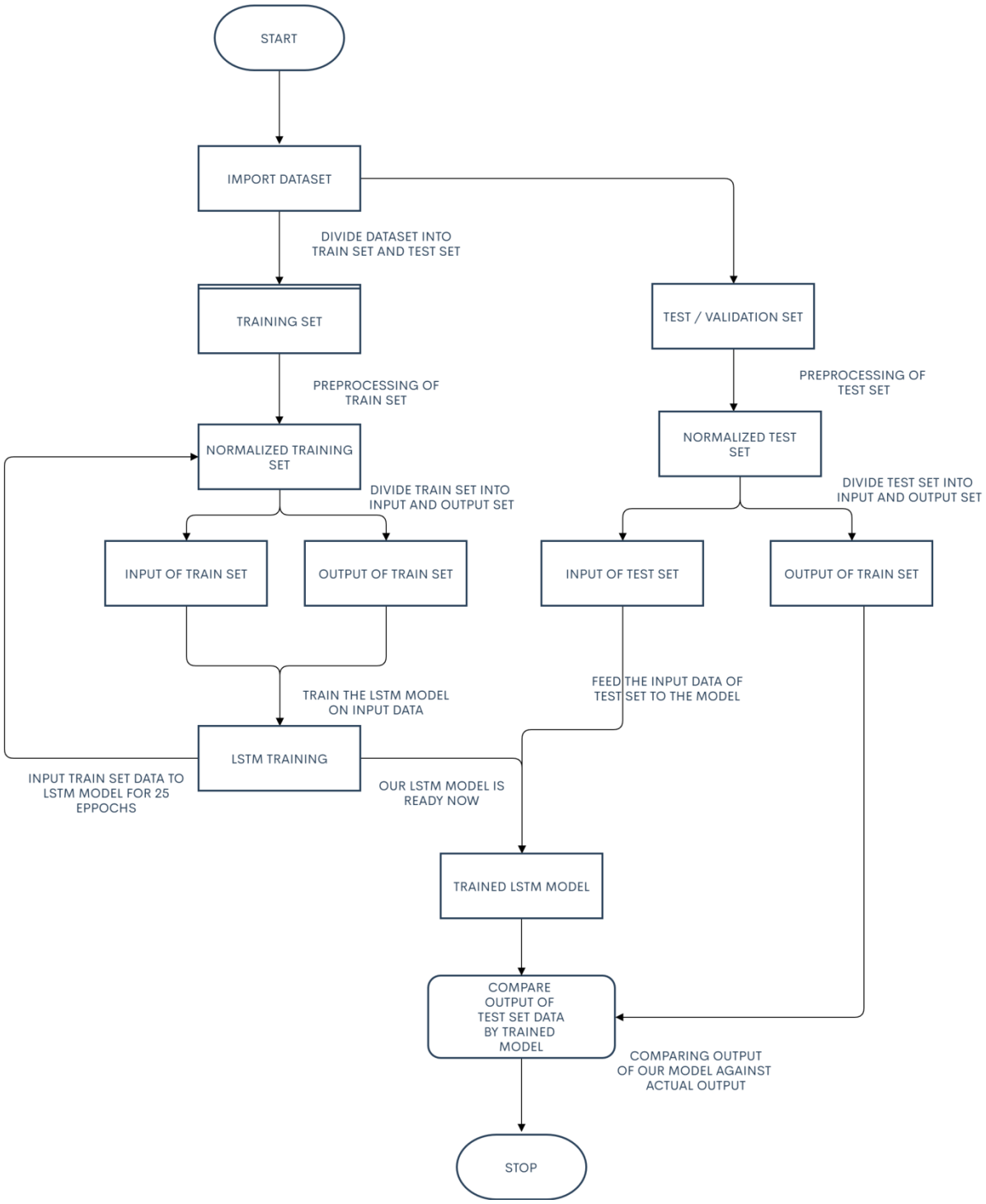
As social media have received more attention nowadays, both public and private opinion about a wide variety of subjects are expressed and spread continually via numerous social media. Twitter is one of the social media that is gaining popularity. Twitter offers organizations a fast and effective way to analyze customers' perspectives toward the critical to success in the market place. Twitter have become major platforms of information exchange and communication between users, with tweets as the common information carrier. As an old saying has it, an image is worth a thousand words. The image tweet is a great example of multimodal sentiment. In Developing a program for sentiment analysis is an approach to be used to computationally measure customers' perceptions. This project reports on the design of a sentiment analysis, extracting a vast number of tweets. Prototyping is used in this development. Results classify customers' perspective via tweets into positive and negative, which is represented in a confusion matrix.

Keywords: Tweets, LTSM, ReLU, Training, Testing

1. Introduction:

Sentiment Analysis is also known as “Opinion Mining”. The task of sentiment classification of tweets is notoriously difficult due to both the brevity of the form and the common use of nonstandard spellings or slang terms. While many studies have described sentiment classification systems with incredibly high levels of accuracy, most have not been tested on Twitter data. Two-way sentiment analysis is a task that many machine learning systems have generally performed very well on. With recent advances in deep neural architectures, it is not uncommon to see ML systems achieve over 95% accuracy when classifying text such as IMDB movie reviews or Amazon.com product reviews. However, few of these systems were extensively trained and tested on data from Twitter, a social media platform where users communicate through 140-character posts called "tweets". Tweets pose an interesting natural language processing challenge because they can be very difficult to classify due to their limited length, and because Twitter users often employ unconventional spelling and grammar to convey their messages.

2. Proposed Method



At first, the whole dataset is processed. Subsequently, the data set is divided into 2 sets i.e. train and test sets respectively. This is done because the LSTM model should be trained with train set data and then check the performance of the model with test set data. The accuracy and loss of the model is represented using confusions matrix.

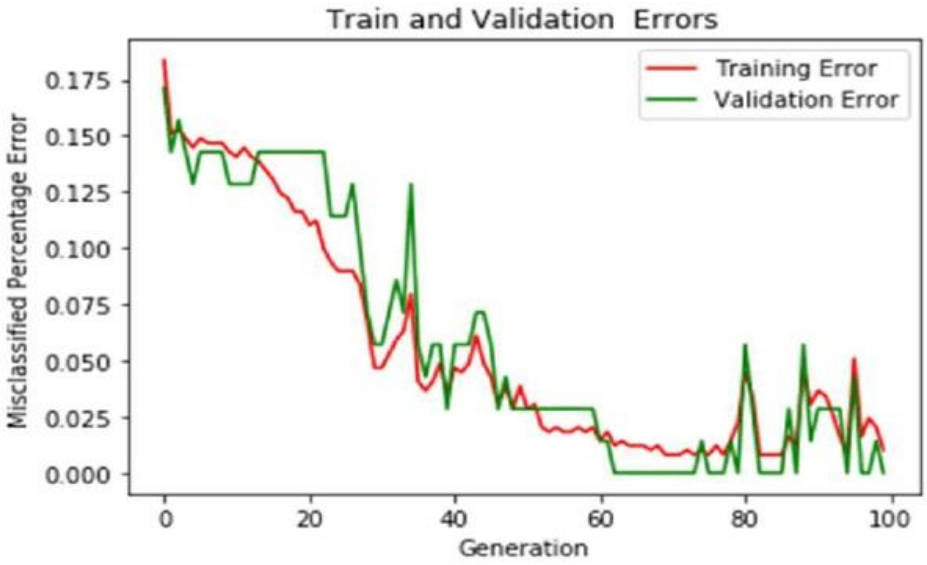
Any special characters or numbers present in the tweet were removed and for every tweet we set the maximum length i.e. the maximum features as 4000. Then we used porter stemmer to make similar words one word. For eg- word like ‘his’, ‘her’, ‘he’, ’she’, ’we’, ‘they’ or ‘make’, ’making’, ‘made’ were converted into one word. Then the tweets were tokenized, i.e. one unique number is given to every word. Then we padded the tweets to maintain tweets of same length if the length of tweets is less than 4000.

First, the dataset is divided into training and validation sets in the ratio 2:1. Then the training set data is sent through an LSTM layer. After LSTM layer there is a dense output layer consisting of single neuron. We used RELU activation function in LSTM layer and Softmax activation function in output layer. Dropouts have been used in LSTM layer to avoid overfitting and batch normalization after LSTM layer to avoid gradient explosion or vanishing problem. To compute loss we have used categorical cross entropy function and used Adams optimizer to optimize parameters during back propagation. We used batch size of 32 i.e. weights will get updated after sending 32 tweets from train set.

3. Database and performance metrics

Language : Python
Libraries : TensorFlow, Keras, Pandas
Platform : Google Colab
Dataset : Twitter tweets dataset from kaggle

Epoch	Loss	Accuracy
1	0.1849	0.9391
2	0.1076	0.9638
3	0.0842	0.9720
4	0.0642	0.9725
5	0.0502	0.9823
6	0.0405	0.9862
7	0.0332	0.9888
8	0.0262	0.9911
9	0.0240	0.9914
10	0.0206	0.9929



4.

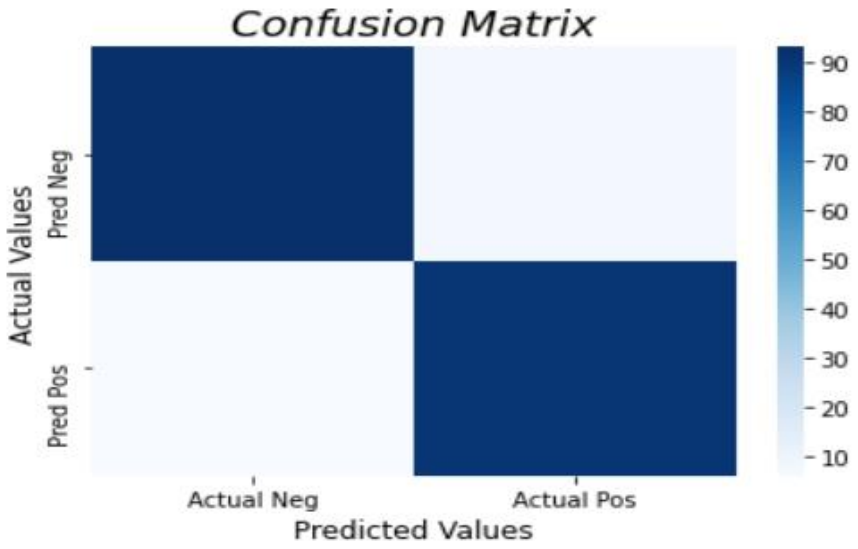
Results

We have split the data into 2 parts The training set had 70% of the whole dataset and validation set has 30% of whole dataset. We have performed 10 epochs and kept batch size as 32. As result we have obtained training set accuracy of 99.29% and loss of 2.06% and we got validation set accuracy of 95.31% and validation loss of 27.74%.

```
batch_size = 32
model.fit(X_train, Y_train, epochs = 10, batch_size=batch_size,validation_data = (X_test,Y_test))
```

```
Epoch 1/10
700/700 [==============================] - 113s 161ms/step - loss: 0.1849 - accuracy: 0.9391 - val_loss: 0.1459 - val_accuracy: 0.9483
Epoch 2/10
700/700 [==============================] - 109s 156ms/step - loss: 0.1076 - accuracy: 0.9638 - val_loss: 0.1306 - val_accuracy: 0.9565
Epoch 3/10
700/700 [==============================] - 107s 153ms/step - loss: 0.0842 - accuracy: 0.9720 - val_loss: 0.1415 - val_accuracy: 0.9570
Epoch 4/10
700/700 [==============================] - 107s 153ms/step - loss: 0.0642 - accuracy: 0.9775 - val_loss: 0.1650 - val_accuracy: 0.9587
Epoch 5/10
700/700 [==============================] - 109s 156ms/step - loss: 0.0502 - accuracy: 0.9823 - val_loss: 0.1656 - val_accuracy: 0.9547
Epoch 6/10
700/700 [==============================] - 109s 156ms/step - loss: 0.0405 - accuracy: 0.9862 - val_loss: 0.1963 - val_accuracy: 0.9559
Epoch 7/10
700/700 [==============================] - 110s 157ms/step - loss: 0.0332 - accuracy: 0.9888 - val_loss: 0.2073 - val_accuracy: 0.9560
Epoch 8/10
700/700 [==============================] - 110s 158ms/step - loss: 0.0262 - accuracy: 0.9911 - val_loss: 0.2560 - val_accuracy: 0.9540
Epoch 9/10
700/700 [==============================] - 111s 158ms/step - loss: 0.0240 - accuracy: 0.9914 - val_loss: 0.2654 - val_accuracy: 0.9502
Epoch 10/10
700/700 [==============================] - 110s 157ms/step - loss: 0.0206 - accuracy: 0.9929 - val_loss: 0.2774 - val_accuracy: 0.9531
<tensorflow.python.keras.callbacks.History at 0x7f020cad0f28>
```

We have also tried to evaluate our model performance using confusion matrix that shows the model has predicted most of the positive and negative tweets correctly.



As the actual negative and predicted negative parts and actual and predicted positive have higher color intensity it means our model has successfully predicted tweet sentiments. There are fewer tweets for which model has given opposite output.

4. Discussions

In the research paper "Text Sentiment Analysis Based on Long Short-Term Memory", the authors have carried out research work on sentiment analysis on movie reviews similar to our work on Tweets. They have categorized each review into 3 parts positive, negative and neutral. They have used LSTM model. Their dataset was also not so large . First they have vectorized input text data then divide into training and test sets. They have trained the model on the train set data and checked accuracy using test set data.

A similar Journal paper "Spam SMS Filtering using Recurrent Neural Network and Long Short Term Memory", the authors have carried out similar work as done in previous paper. Here the authors have tried to figure out if a SMS is spam or not, using LSTM. Their methodology is also similar i.e. adapted by authors of previous journal. They have compared their LSTM model with Naïve Bayes and SVM models and found that accuracy with LSTM is higher than those two models.

5. Conclusion

We have designed nearly 20-25 models and also tried out different preprocessing steps. In the end we got very good accuracy for this dataset using model with a single LTSM layer of 512 units and a dictionary size of 4000 and pre padding of sequences.

We got an train set accuracy of 99% and 95% train set accuracy. The confusion matrix shows that the accuracy of detecting positive tweets correctly is lesser in percentage than accuracy of detecting negative tweets, this is because the dataset is not a properly balanced one. It has very large number of negative tweets than positive tweets. In supervised learning projects datasets of equal number of entries of different types is preferred. Still with this dataset we have managed to get very good accuracy. There is still variance issue is there. In order to solve that we will definitely need a better dataset.

We would like to conclude by saying that lstm models are very powerful ones as this type of dataset is quite complicated to process but only single layer of lstm layer got us very good result.

REFERENCES

[1] Dan Li, Jiang Qian, Text Sentiment Analysis Based on Long Short-Term Memory, 2016 First IEEE International Conference on Computer Communication and the Internet (ICCCI)

[2] J. Bollen, H. Mao, and A. Pepe, Modeling public mood and emotion: Twitter sentiment and socio-economic phenomena. In ICWSM, 2011.

[3] D. Borth, T. Chen, R. Ji, and S.-F. Chang. Sentibank: large-scale ontology and classifiers for detecting sentiment and emotions in visual content. In ACM MM, pages 459–460. ACM, 2013.

[4] Arijit Chandra, Sunil Kumar Khatri, Spam SMS Filtering using Recurrent Neural Network and Long Short Term Memory, 2019 4th International Conference on Information Systems and Computer Networks (ISCON).

[5] D. Borth, R. Ji, T. Chen, T. M. Breuel, and S. Chang. Large-scale visual sentiment ontology and detectors using adjective noun pairs. In ACM MM, pages 223–232, 2013.

[6] D. Cao, R. Ji, D. Lin, and S. Li. A cross-media public sentiment analysis system for microblog. Multimedia Systems, pages 1–8, 2014.

[7] X. Chen and C. Lawrence Zitnick. Mind's eye: A recurrent visual representation for image caption generation. In CVPR, June 2015.

[8] D. C. Cires¸an, U. Meier, J. Masci, L. M. Gambardella, and J. Schmidhuber. Flexible, high performance convolutional neural networks for image classification. In IJCAI, pages 1237–1242, 2011

Personal Contribution-:

I helped my team in building the optimized long-short-term-memory model mainly. I have coded the model and done research work about how to develop an optimized model and also coordinated and merged works done by my teammates. Developing a model was never easy job. As I have to build a model and run it that takes around 30-50 minutes depending on the number of layers added. I have initially just used regular dropouts for normalization then again after doing some research work I have added recurrent dropout that helped in developing a optimized model.

Initially while developing the model I noticed that number of positive tweets are way lesser than the number of negative tweets so I thought that using a deeper model will give better result. I have tried with different models with at-least 2 to 4 hidden LSTM layers that gave good training accuracy and validation accuracy also but while creating the confusion matrix it showed that it is not detecting a single positive tweets correctly which led me to change and try different models quite some time.

Finally, when I tried with a single LSTM layer with 512 units I got very good results. This taught me how powerful these layers are. Detecting such tweets is not a easy task at all because generally when we do supervised machine learning projects we train the model with equal number of data of all verities otherwise the model won't learn properly. But here in this project the number of positive tweets are not even 10% of the negative tweets, but I developed a model that gave very good accuracy for this task also.

We have tried to create a uniform dataset just by keeping the same number of negative tweets as the number of positive tweets present in the dataset. We ran several models with this dataset also the accuracy was very bad for both training and validation set.

We also shuffled the dataset that resulted in increasing the accuracy. So we drew a conclusion from this that while performing machine learning tasks we need to shuffling a dataset is always a good decision.

What I learnt from this is, whenever we do a deep learning project always start with one hidden layers and change the units and observe how well the model is performing. Always try to have a uniform dataset and shuffling the dataset is a good practice. While developing a deep learning project we need to be patient and while one model is in the training state we need to create a new model without deleting the previous model so that wastage of time is minimal.

I would like to conclude by saying that acting as a coordinator and merging work done by my team mates in to one single file gave me understanding of various different things like

padding and the importance of dictionary size. I have also changed dictionary size quite some times from 200 to 4000. I have observed that as I increased the vocab size and changed type of padding the accuracy of our model increased and loss decreased. This taught me that preprocessing the data perfectly can give way better results for a same model and it is always a good practice to fine tune the data and again and again before changing the model. Moreover, this projected was very much time taking and taught me a lot of new thing that I was unaware of previously.