Twitter Sentiment Analysis

Rishab

DOS: 18.07.2020

Abstract- We know that social media usage is increasing rapidly with every passing day. Social Media is increasingly used by people all around to express their feelings and opinions in the form of text messages. Detecting Sentiment has a broad usage e.g. identifying anxiety or depression of individuals or their groups and thus giving an idea of mood of the community so that one can take any action for the well-being of the mentioned. This project is carried out using the fundamentals of Natural Language Processing techniques.

Keywords- Deep Learning, Machine Learning, Python, Social Media, Sentiment Analysis

I. INTRODUCTION

Twitter serves as a mean for individuals to express their thoughts or feelings about different subjects. These emotions are used in various analytics for better understanding of humans. In this project we aim to do sentiment analysis of various tweets using machine learning and deep learning techniques and apply various data preprocessing steps which serve as input to those models. Sentimental analysis has many applications for different domains for example in businesses to get feedbacks for products by which companies can learn user's feedback and reviews on social medias.

In the following sections we will discuss the data source and preprocessing we did, followed by various machine learning techniques tested and compared for the analysis.

II. PREVIOUS WORK

There has been a lot of work regarding in this topic of analysis previously. Earlier there were machine learning models which were once state of the art, converting text to numerical features were done by vectorizers like TF-IDF and Bag of Words. These were good techniques back then and performed well with models like Naïve Bayes Classifiers and Support Vector Machines with modifications in kernel. Recently, many states of the art techniques have been developed since the introduction of Neural Networks in text classification tasks. Neural Networks captured the non-linearity present in the data and hence made it possible to construct complex decision boundaries. With a recent further introduction of attention models and transformers like BERT, UNIVERSAL SENTENCE ENCODER etc., it is now possible to capture the context better than ever as compared to the standard neural network techniques.

III. DATA PROCESSING AND MODEL SELECTION

The observations on reviewing the data are listed in the following subsections:

A. Data and its structure

While exploring the data we found several interesting properties and patterns which were present in the data.

The training data was present as a text file with comma separated fields. The fields comprised of a unique '*tweet_id*', the corresponding '*tweet_text'* which had the text whose sentiment we needed to predict and a '*sentiment*' field which was the label associated with the data. There were a total of 21,465 training data points.

The data provided had a lot of interesting properties, each tweet had several forms of non-plain text items like emojis, mentions, hashtags, URLs etc. We need to preprocess these items so that the text remaining is of similar value and the models don't confuse with unwanted characters.

- The tweets contained emojis(as unicode characters in the form of '\uxxxx') and URLs, emails, mentions(beginning with @ and followed by a twitter handle). These needed to be removed. We accomplished this task by parsing through regular expressions and replacing each occurrence of the above with a blank. These types of elements are not useful in understanding the sentiment of the person, as a mention is a neutral element, it has no value which contributes to the sentiment. The emojis however are used in various ways in daily life, people can be sarcastic and put a random emoji which is can be no way be related to the tweet sentiment and can prove to be ambiguous if we use it, so it is better that it is removed.
- One of the most frequently occurring elements present in the tweet are hashtags. These hashtags can sometime summarize the whole tweet by using some strong words or feelings which may govern the sentiment. We tried to remove the "hash" from the hashtags and use the remaining word. However, the words were not of any meaning altogether as most of the hashtags consisted of two or more words concatenated together. We tried to break the words using a library called 'ekphrasis' but it didn't help much. So, we decided to remove the hashtags in the end.
- Numbers: Any form of Numerical Data, example 1200 oranges or 34 bottles etc, won't help in our analysis in any way, so the numerical data were also removed.

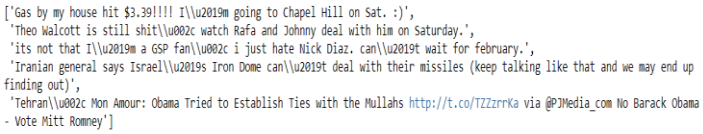
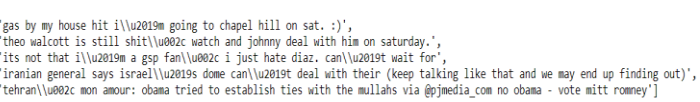


Figure 1: Example of 5 data samples before(above) and after cleaning(below)



B. Model Selection

There were various models which we implemented and compared for this task. The models comprised of both machine learning and deep learning models. The different models we tried are shown below:

i. Support Vector Machines: Initially after data cleaning we used vectorizers both Count Vectorizers (a simple Bag of Words) and TF-IDF vectorizer. After applying the vectorizers we got vectors of size 26,874 in both cases. Naturally both of them were quite sparse. So, we also applied PCA to reduce dimension so as to preserve 99% of the variance explained by the vectorized data. After applying PCA, the vector size reduced to 3135. This means more than 85% of the features contributed to less than 1% of the variance. Now, for applying the machine learning model, we used Support Vector Machine with an 'RBF' kernel, since Radial Basis Kernels have a VC dimension of infinity, we thought it was quite suitable for applying to the non-linearly separable twitter data. The performance with other models is shown in the consequent section of this paper.

ii. Naïve Bayes Classifier: With the same vectorizers as the Support Vector Machines, we tried Multinomial Naïve Bayes Classifiers provided by scikit-learn, we also used Complement Naïve Bayes, which said to be better than Multinomial Naïve Bayes and our predictions verified this statement as well. In both the cases we kept the smoothing factor as 1. Naïve Bayes Classifiers are probabilistic models, and since TF-IDF vectorizers use probabilities for calculations, we tried this model as well.

iii. Convolution Neural Networks with Bidirectional LSTM's and Universal Sentence Encoder: We tried to use Universal Sentence Encoders as the Embedding Layer for this model. The Universal Sentence Encoder is provided by Google and converts the words to 512 dimensional vectors.

Figure 2: The model architecture is shown.

Model: "sequential_1"

Layer (type)	Output Shape	Param #
embedding (Embedding)	(None, 300, 200)	4898400
bidirectional (Bidirectional	(None, 300, 100)	100400
conv1d (Conv1D)	(None, 293, 64)	51264
max_pooling1d (MaxPooling1D)	(None, 146, 64)	0
dropout (Dropout)	(None, 146, 64)	0
conv1d_1 (Conv1D)	(None, 139, 32)	16416
max_pooling1d_1 (MaxPooling1	(None, 69, 32)	0
dropout_1 (Dropout)	(None, 69, 32)	0
dense (Dense)	(None, 69, 32)	1056
dropout_2 (Dropout)	(None, 69, 32)	0
dense_1 (Dense)	(None, 69, 16)	528
global_max_pooling1d (Global	(None, 16)	0
dense_2 (Dense)	(None, 3)	51

Total params: 5,068,115
Trainable params: 169,715
Non-trainable params: 4,898,400

iv. Convolution Neural Networks with Bidirectional LSTM's with GLoVE Embeddings: Just like the previous model but with architectural changes and a different type of embeddings known as GLoVE. The GLoVE embeddings are provided by Stanford University. They are available in many different dimensions, the most commonly used are 100d and 200d embeddings. Stanford provides an exclusive Twitter GLoVE embedding, made for twitter data, using twitter data. We used 200d embedding vectors for our project. **Figure 3:** The model architecture we used with this embedding is shown below.

Model: "sequential"

Layer (type)	Output Shape	Param #
conv1d (Conv1D)	(None, 512, 64)	256
conv1d_1 (Conv1D)	(None, 512, 96)	18528
max_pooling1d (MaxPooling1D)	(None, 102, 96)	0
bidirectional (Bidirectional	(None, 1024)	2494464
dense (Dense)	(None, 512)	524800
dropout (Dropout)	(None, 512)	0
dense_1 (Dense)	(None, 320)	164160
dropout_1 (Dropout)	(None, 320)	0
dense_2 (Dense)	(None, 128)	41088
dense_3 (Dense)	(None, 3)	387

Total params: 3,243,683
Trainable params: 3,243,683
Non-trainable params: 0

v. BERT: BERT stands for Bidirectional Encoder Representations from Transformers. It is a state-of-the-art deep learning language model developed by researchers at Google. It has proven to be very efficient in many complex NLP tasks like Named Entity Recognition, Question and Answering systems, Chatbots, Text Classification etc. Almost in every case it has outperformed many machine learning and deep learning models by a large score/factor. It was first presented in the paper "*BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*" in October 2018. It is derived from the class of self-attention and transformer models which were also introduced by Google in the paper "*Attention*

is All You Need" in 2017. We tried this model in our task as well, and we got good results as expected.

IV. OBSERVATIONS

After training and testing these models, we saw several observations, text data is nonlinear in nature, so creating linear boundaries is quite difficult for classification, and hence it was certainly difficult for all the models to achieve a very high accuracy, below we list the observations we found in our analysis. We split our dataset for training as follows:

			tweet_id	tweet_text
sentiment	**label**	**data_type**		
negative	**1**	**train**	3048	3048
		val	339	339
neutral	**2**	**train**	8113	8113
		val	901	901
positive	**0**	**train**	8157	8157
		val	907	907

Figure 4: Distribution split of dataset for training.

- Convolution Neural Networks with Bidirectional LSTM's with GLoVE Embeddings: The loss and accuracy plot for both training and validation set for this model is shown below.

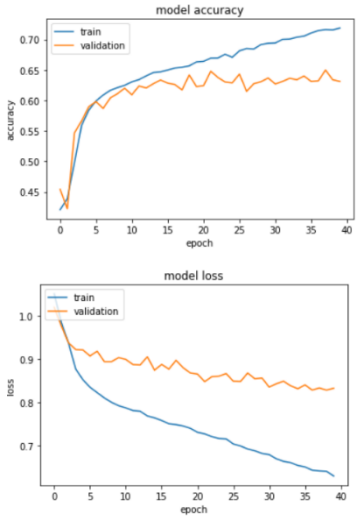
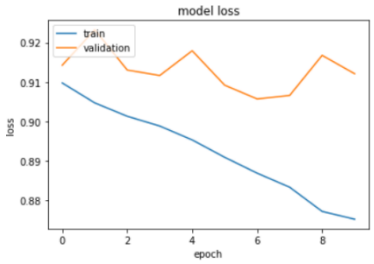


Figure 5: Accuracy and Loss plot for Neural Net with GLoVE embeddings.

- Convolution Neural Networks with Bidirectional LSTM's and Universal Sentence Encoder: The loss and accuracy plot for both training and validation set for this model is shown below.



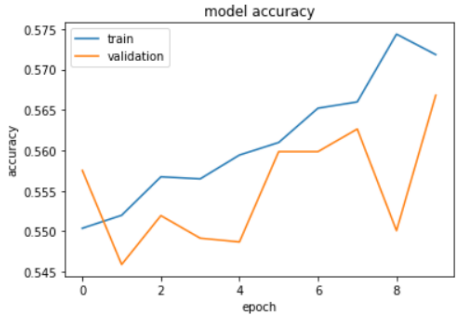


Figure 6: Accuracy and Loss plot for Neural Net with Universal Sentence Encoder embeddings

- BERT: We used BERT model with the help of transformers provided by the "hugging face" package. It contains several pretrained language models like BERT itself. It provides a comfortable API for the use of transformers in various NLP tasks. It is supported by both Tensorflow and PyTorch, but we chose PyTorch for this model because of its dynamic computational graphs. It has a stronger GPU support and has several packages written in C language as well. Moreover, BERT is a very deep neural network, so it requires lots of time and computations to train the model. We used NVIDIA GeForce RTX 2060 for this project. It took around 87 minutes for training the model on the GPU for only 4 epochs. We trained the model for only 4 epochs because it was found that on training the model for more than that it began to overfit.

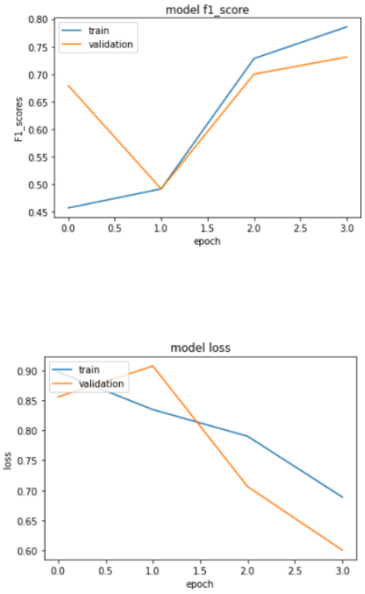


Figure 7: Weighted F1_score and Loss plot for BERT model.

RESULTS

Model Name	F1-score
Support Vector Machines(Linear Kernel)	0.525
Support Vector Machines(RBF Kernel)	0.645
Naïve Bayes Classifier (ComplementNB)	0.6112
Naïve Bayes Classifier (MultinomialNB)	0.592
Convolution Neural Networks with Bidirectional LSTM's and Universal Sentence Encoder	0.575
Convolution Neural Networks with Bidirectional LSTM's with GLoVE Embeddings	0.6635
BERT	0.7501

Figure 8: F1-scores of various models for comparison.

We see that BERT outperforms all the models with a large gap in weighted F1-score. This boost in performance can be attributed to its self-attention. Self-attention models are continuously changing the way we configure our NLP tasks.

V. CONCLUSION AND FUTURE WORKS

In this technical paper, we discussed the importance of social network analysis and its applications in different areas. We focused on Twitter as and have implemented the python program to implement sentimental analysis. Upon analysis we concluded that, state-of-the-art attention and transformer model BERT by Google outperformed all the other models. There might have been effect on the scores due to randomness present in the models, but BERT outperforms each and everyone of them by a great factor. We hope to improve the models by trying much more complex architectures and also use them with different parameters.

.

REFERENCES

[1] Vishal A. Kharde, "Sentiment Analysis of Twitter Data: A Survey of Techniques," in *International Journal of Computer Applications (0975 – 8887) Volume 139 – No.11, April 2016*.

[2] Hamid Bagheri, *Sentiment analysis of twitter data*.

[3] Haddi, E., Liu, X., & Shi, Y. (2013). The role of text pre-processing in sentiment analysis. Procedia Computer Science, 17, 26-32."

[4] R. Parikh and M. Movassate, "Sentiment Analysis of User-GeneratedTwitter Updates using Various Classi_cation Techniques",CS224N Final Report, 2009

[5] Dmitry Davidov, Ari Rappoport." Enhanced Sentiment Learning Using Twitter Hashtags and Smileys". Coling 2010: Poster Volumepages 241{249, Beijing, August 2010

[6] How to Run Text Classification Using Support Vector Machines, Naive Bayes, and Python, https://aiiseasy.com/2019/06/09/text-classification-svm-naive-bayes-python/

[7] Working with Text Data. https://scikit-learn.org/stable/tutorial/text_analytics/working_with_text_data.html

[8] BERT https://huggingface.co/transformers/model_doc/bert.html

[9] Multi-Class Text Classification with LSTM https://towardsdatascience.com/multi-class-text-classification-with-lstm-1590bee1bd17

[10] Text Classification, Part I - Convolutional Networks, https://richliao.github.io/supervised/classification/2016/11/26/textclassifier-convolutional/

[11] Use-cases of Google's Universal Sentence Encoder in Production, https://towardsdatascience.com/use-cases-of-googles-universal-sentence-encoder-in-production-dd5aaab4fc15