Twitter Sentiment Analysis

Ashit Neema   
 Computer Science Department  
 Utah State University  
 Logan Utah U.S.A.  
ashu.neema28@gmail.com

Kartik Saini  
Computer Science Department  
 Utah State University  
 Logan Utah U.S.A.  
kartik.saini@usu.edu

Vedang Gharat  
Computer Science Department  
 Utah State University  
 Logan Utah U.S.A.  
 vedanggharat98@gmail.com

ABSTRACT

Sentiment analysis over Twitter offers organization a fast and effective way to monitor the publics’ feelings towards there own organization and the organization with whom they compete their competitors. To assess the performance of sentiment analysis methods over Twitter a small set of evaluation datasets have been released in the last few years. We present an overview of eight publicly available and manually annotated evaluation datasets for Twitter sentiment analysis. Based on this review, we show that a common limitation of most of these datasets, when assessing sentiment analysis at target (entity) level, is the lack of distinctive sentiment annotations among the tweets and the entities contained in them. For example, the tweet “I love iPhone, but I hate iPad” can be annotated with a mixed sentiment label, but the entity iPhone within this tweet should be annotated with a positive sentiment label. Aiming to overcome this limitation, and to complement current evaluation of datasets, a new way of evaluating dataset where tweets and targets (entities) are annotated individually and therefore may present different sentiment labels. This provides us a comparative study of the various datasets along several dimensions including: total number of tweets, vocabulary size and sparsity. We also investigate the pair-wise correlation among these dimensions as well as their correlations to the sentiment classification performance on different datasets.

**INTRODUCTION**

Nowadays, the age of Internet has changed the way people express their views, opinions. It is now mainly done through blog posts, online forums, product review websites, social media ,etc. Nowadays, millions of people are using social network sites like Facebook, Twitter, Google Plus, etc. to express their emotions, opinion and share views about their daily lives. Through the online communities, we get an interactive media where consumers inform and influence others through forums. Social media is generating a large volume of sentiment rich data in the form of tweets, status updates, blog posts, comments, reviews, etc. Moreover, social media provides an opportunity for businesses by giving a platform to connect with their customers for advertising. People mostly depend upon user generated content over online to a great extent for decision making. For e.g. if someone wants to buy a product or wants to use any service, then they firstly look up its reviews online, discuss about it on social media before taking a decision. The amount of content generated by users is too vast for a normal user to analyze. So there is a need to automate this, various sentiment analysis techniques are widely used. Sentiment analysis tells user whether the information about the product is satisfactory or not before they buy it. Marketing firms use this data to understand about their products or services in such a way that it can be offered as per the user’s requirements. Text Information retrieval techniques mainly focus on processing, searching or analyzing the factual data present. Facts have an objective component but, there are some other text contents which express subjective characteristics. These contents are mainly opinions, sentiments, appraisals, attitudes, and emotions, which form the core of Sentiment Analysis. It offers many challenging opportunities to develop new applications, mainly due to the huge growth of available information on online sources like blogs and social networks. For example, recommendations of items proposed by a recommendation system can be predicted by taking into account considerations such as positive or negative opinions about those items by making use of sentiment analysis.

**DATA PREPROCESSING**

The preprocessing of the data is an essential step as it makes the raw text ready for mining that is it becomes easier to extract information from the text and apply model to it in our case RNN model. If we skip the step then there is a high chance that you are going to get error in the output and also the output data will be less accurate and inconsistent. The objective of this step is to remove the data which is not required such as punctuations, special characters, numbers and terms which do not carry any weightage in context to predict the text data as negative or positive or neutral.

Basic preprocessing need to clear out the unwanted data

The approach used in preprocessing is explained in detailed below:

* ****Remove Numbers****

It is a common tactic to remove numbers from text, because they do not contain any sentiment. However, some researchers argue that keeping the numbers may improve classification effectiveness.

* ****Replace Repetitions of Punctuation****

 We distinguish three punctuation signs, whose repetitions concern us. These are the exclamation, question, and stop marks. The use of these punctuation marks signals the existence of intense emotion. If we find more than one in a row, we replace it with a representative tag. For example the token ‘???’ will be replace with ‘multiQuestionMark’.

* ****Handling Capitalized Words****

 Same as before, capitalized words may imply intense emotion, so we detect all the words that are longer than two characters with all of their characters capitalized. We prefix them with ‘ALL\_CAPS\_’ did, so they can be identified in machine learning.

* ****Lowercasing****

One of the most common pre-processing techniques is to lowercase all words. By doing so, many words are merged and the dimensionality of the problem is reduced.

* ****Replace Slang and Abbreviations****

Social media users usually write in an informal way and their texts contain a lot of slang and abbreviations. These words, in order to be interpreted correctly, have to be replaced to impute their meaning. We manually constructed a lookup table consisting of 290 such words and their replacements. Some examples are the words ‘ty’, ‘qq’ and ‘omg’, which respectively mean and replaced by ‘thank you’, ‘crying’, and ‘oh my god’.

* ****Replace Elongated Words****

Elongated is a word when it contains a character that is repeating more than two times, like the word ‘shoulda’. It is important to replace words like this with their source words, so they can be merged. Otherwise, the classifier will treat them as different words, and probably the elongated ones will be ignored because of their low frequency of occurrence. Detecting and replacing elongated words have been examined by researchers before.

* ****Replace Contractions****

One technique that can be used in pre-process is the replacement of contractions, i.e. words like ‘won’t’ and ‘don’t’, that will be replaced with ‘will not’ and ‘do not’, respectively.

* ****Replace Negations with Antonyms****

It is an approach that has not been used by many researchers and is presented in . We search in each sentence for the word ‘not’ and then, we check if the next word has an antonym. If yes, we replace both words with the antonym. For example, the phrase ‘not good’ will be replaced with the word ‘bad’, using WordNet .

* ****Handling Negations****

When text analysis is performed in a word level, it is very challenging to handle negation. One method that is widely used by researchers is the detection of words that imply negation and the addition of the prefix ‘NOT\_’ in every word after them until the first punctuation mark.

* ****Remove Stop-words****

Stop-words are function words with high frequency of presence across all sentences. It is considered needless to analyze them, because they do not contain much useful information. The set of these words is not completely predefined and it can be changed by removing or adding more to it, depending on the application. In our implementation, we used the standard stop-words provided by NLTK .

* ****Stemming****

It is the process of removing the endings of the words in order to detect their root form. By doing so, many words are merged and the dimensionality is reduced. It is a widely used method that generally provides good results; we used the Porter Stemmer .

* ****Lemmatizing****

Another method of merging many words to one is Lemmatization. In this method, we remove the endings of the words in order to detect their lemmas, i.e. their root forms in a dictionary.

* ****Replace URLs and User Mentions****

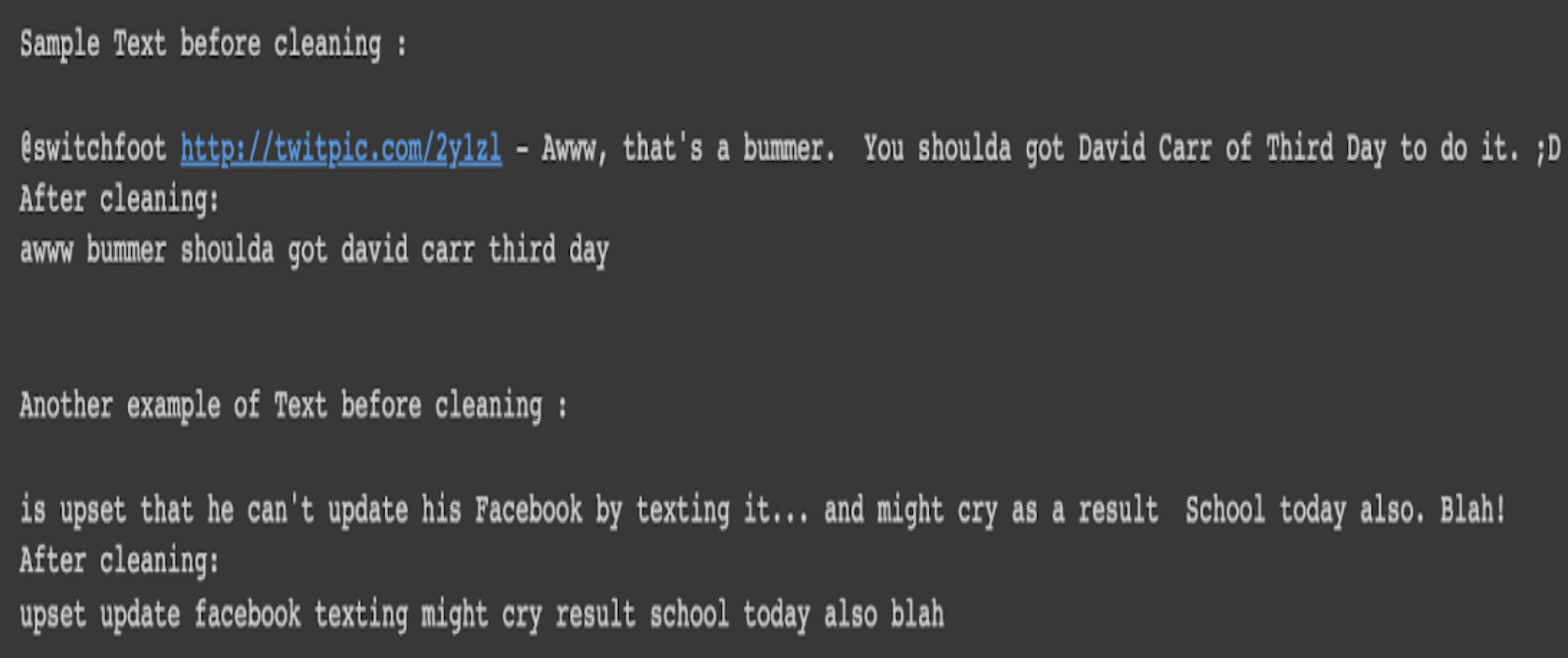
In Twitter texts, almost every sentence contains a URL and a user mention. Their presence does not contain any sentiment and one approach is to replace them in pre-processing with tags as did. We used the tags ‘URL’ and ‘AT\_USER’.

* ****Spelling** **Correction****

It is very common in informal texts for users to make spelling errors that might make classification harder. By using tools that automatically correct these errors, it is possible to improve classification effectiveness. While no corrector is perfect, they have some —usually high— accuracy of success.

* ****Remove Punctuation****

In many works, it is common to remove punctuation signs in pre-processing. However, many times the presence of punctuation marks denotes the existence of some sentiment. For example, an exclamation mark may mean an intense positive or negative sentiment. So if we remove them we might decrease the accuracy of classification.



**FEATURE EXTRACTION**

The preprocessed dataset has many distinctive properties. In the feature extraction method, we extract the aspects from the processed dataset. Later this aspect are used to compute the positive, neutral and negative aspects in a sentence which is useful for determining the opinion of the individuals using models like Machine learning techniques require representing the key features of text or documents for processing. These key features are considered as feature vectors which are used for the classification task. Some examples features that have been reported in literature are

* Words And Their Frequencies: Unigrams, bigrams and n-gram models with their frequency counts are considered as features. There has been more research on using word presence rather than frequencies to better describe this feature. showed better results by using presence instead of frequencies.
* Parts Of Speech Tags

Parts of speech like adjectives, adverbs and some groups of verbs and nouns are good indicators of subjectivity and sentiment. We can generate syntactic dependency patterns by parsing or dependency trees

* Opinion Words And Phrases

Apart from specific words, some phrases and idioms which convey sentiments can be used as features. e.g. cost someone an arm and leg.

* Position Of Terms The position of a term with in a text can affect on how much the term makes difference in overall sentiment of the text.
* Negation

Negation is an important but difficult feature to interpret. The presence of a negation usually changes the polarity of the opinion.

* Syntax

Syntactic patterns like collocations are used as features to learn subjectivity patterns by many of the researchers.

* Applying Feature SelectionModel

Word2Vec Feature SelectionModel

Word embedding is a way used for depicting the words in the form of vectors. The aim of word embedding is to redefine the high- dimensional word characteristics into lowdimensional feature vectors by preserving the corpus contextual similarity.

1. The advantages of using word embedding’s over Bag of words (BOW) or TF-IDF are:
2. Dimensionality reduction - a significant reduction in the no. of features required to build a model.
3. It captures the meanings of the words, semantic relationships and the different types of contexts the words are utilized in sentences.

* Word2Vec Embedding’s

Word2Vec algorithm is a combination of 2 Techniques namely continuous bag of words (CBOW) and Skip-gram model. Both the techniques are shallow neural networks used for mapping a word/words to a target variable which can also be a word or a set of words. Also, these techniques learn weights of words which are represented in the form of word vector representations.

There are two types of Word2Vec model designs are used:

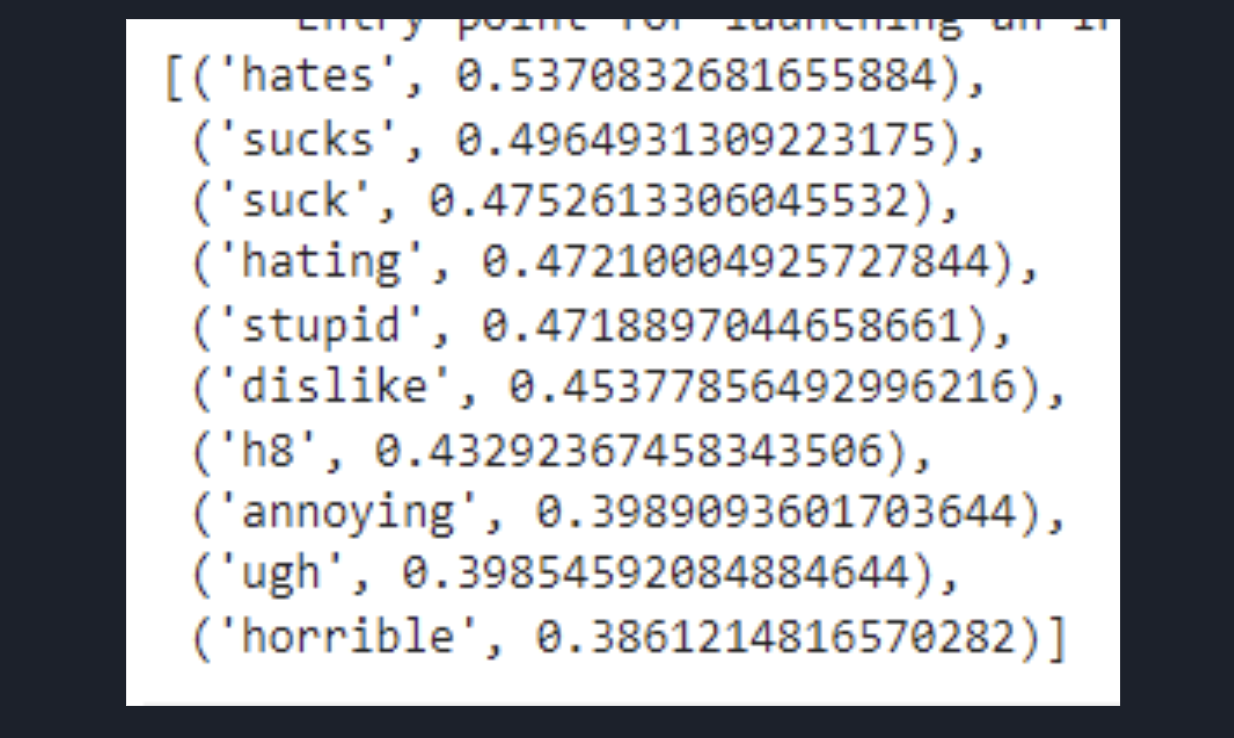
1. Continuous Bag-of-Words model (CBOW)
2. Skip-Gram model

CBOW tends to predict the likelihood of a word given context it is being utilized in and the context can be a single adjacent word or a group of surrounding words. The Skip gram model works in a reverse manner, it tries to predict the context for a given word.

The Skip-gram model is being used and has the following advantages:

Advantage 1: Skip-gram captures 2 semantics for a word that is it will have 2 vector representations of ‘Apple’. One can be representing the fruit and the other representing the technology company.

Advantage 2: Generally, the Skip-gram with negative subsampling outperforms Continuous bag of words.



* Training and Testing Set

Training a Word2Vec model on our data is very important in order to obtain vector representations for all the unique words present in our corpus. There is one more option of using pre-trained word vectors instead of training our own model. However, in this paper, we will be training our own vectors since the size of the pre-trained word vectors is generally huge.

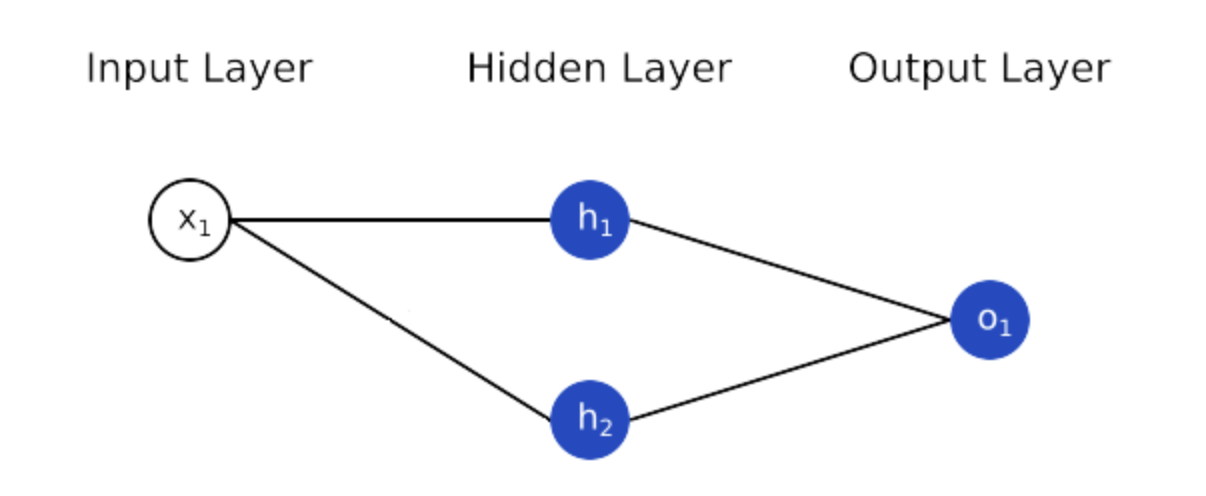
One can see from the training data that the Word2Vec model does a good job of finding the most similar words for a given word. But how is it able to do so? That’s because it has learned vectors for every unique word in our data and it uses cosine similarity to find out the most similar vectors.

* Preparing Vectors for Tweets

Since our data contains tweets and not just words, one would have to figure out a way to use the word vectors from the Word2Vec model to create vector representation for an entire tweet. There is a simple solution to this problem, that is by simply taking the mean of all the word vectors present in the tweet. The length of the resultant vector will be the same, i.e. 200. Thereby, repeating the same process for all the tweets in our data and obtain their vectors. Now that one has 200 word2vec features for our data.

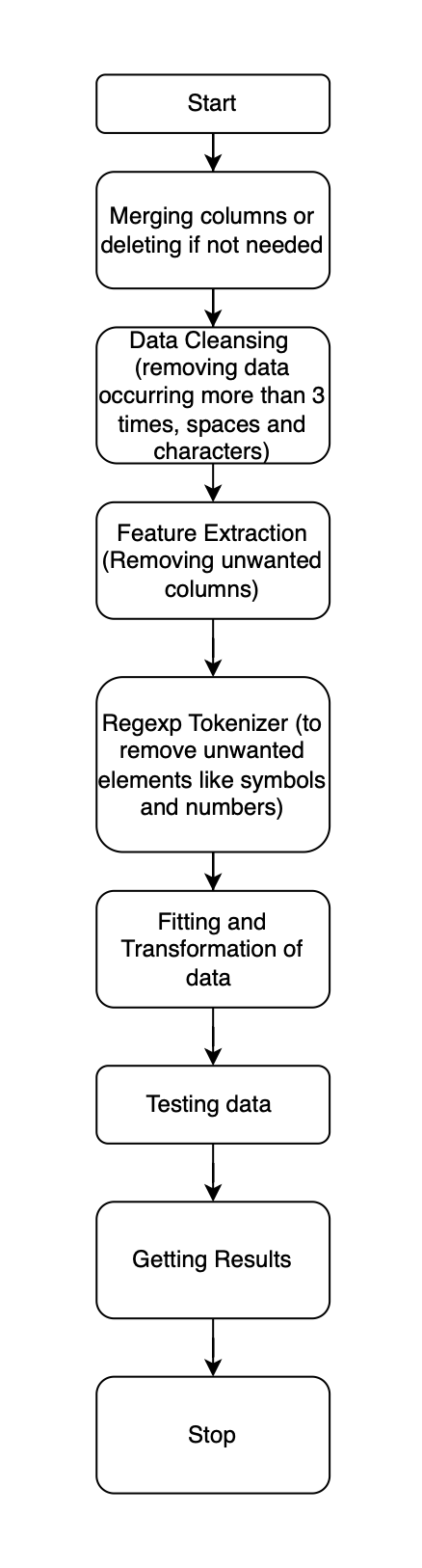
**MODEL**

* Sequential Model API, lets you create a model layer by layer for most problems. It’s straightforward (just a simple list of layers), but it’s limited to single-input, single-output stacks of layers.
* The Sequential API is a framework for creating models based on instances of the sequential() class. The model has one input variable, a hidden layer with two neurons, and an output layer with one binary output. Additional layers can be created and added to the model



* Word embeddings provide a dense representation of words and their relative meanings. They are an improvement over sparse representations used in simpler bag of word model representations. Word embeddings can be learned from text data and reused among projects. They can also be learned as part of fitting a neural network on text data.
* Dropout is a technique where randomly selected neurons are ignored during training. They are “dropped-out” randomly. This means that their contribution to the activation of downstream neurons is temporally removed on the forward pass and any weight updates are not applied to the neuron on the backward pass.
* As a neural network learns, neuron weights settle into their context within the network. Weights of neurons are tuned for specific features providing some specialization. Neighboring neurons become to rely on this specialization, which if taken too far can result in a fragile model too specialized to the training data. This reliant on context for a neuron during training is referred to complex co-adaptations.
* You can imagine that if neurons are randomly dropped out of the network during training, that other neurons will have to step in and handle the representation required to make predictions for the missing neurons. This is believed to result in multiple independent internal representations being learned by the network.
* The effect is that the network becomes less sensitive to the specific weights of neurons. This in turn results in a network that is capable of better generalization and is less likely to overfit the training data.
* The first layer is the Embedded layer that uses 32 length vectors to represent each word. The next layer is the LSTM layer with 100 memory units (smart neurons). Finally, because this is a classification problem we use a Dense output layer with a single neuron and a sigmoid activation function to make 0 or 1 predictions for the two classes (good and bad) in the problem.
* Because it is a binary classification problem, log loss is used as the loss function (binary cross entropy in Keras). The efficient ADAM optimization algorithm is used. The model is fit for only 2 epochs because it quickly overfits the problem. A large batch size of 64 reviews is used to space out weight updates.

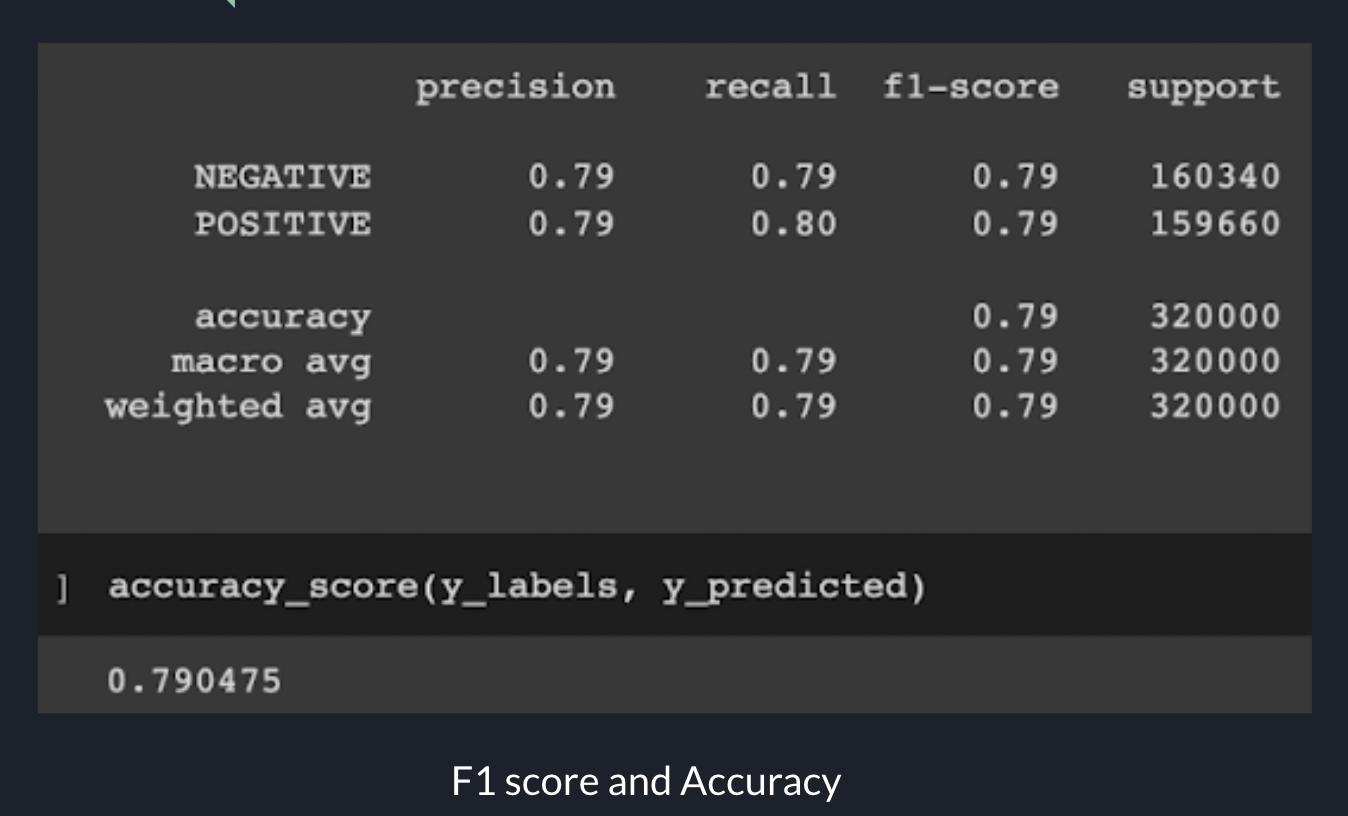
**FLOW DIAGRAM**

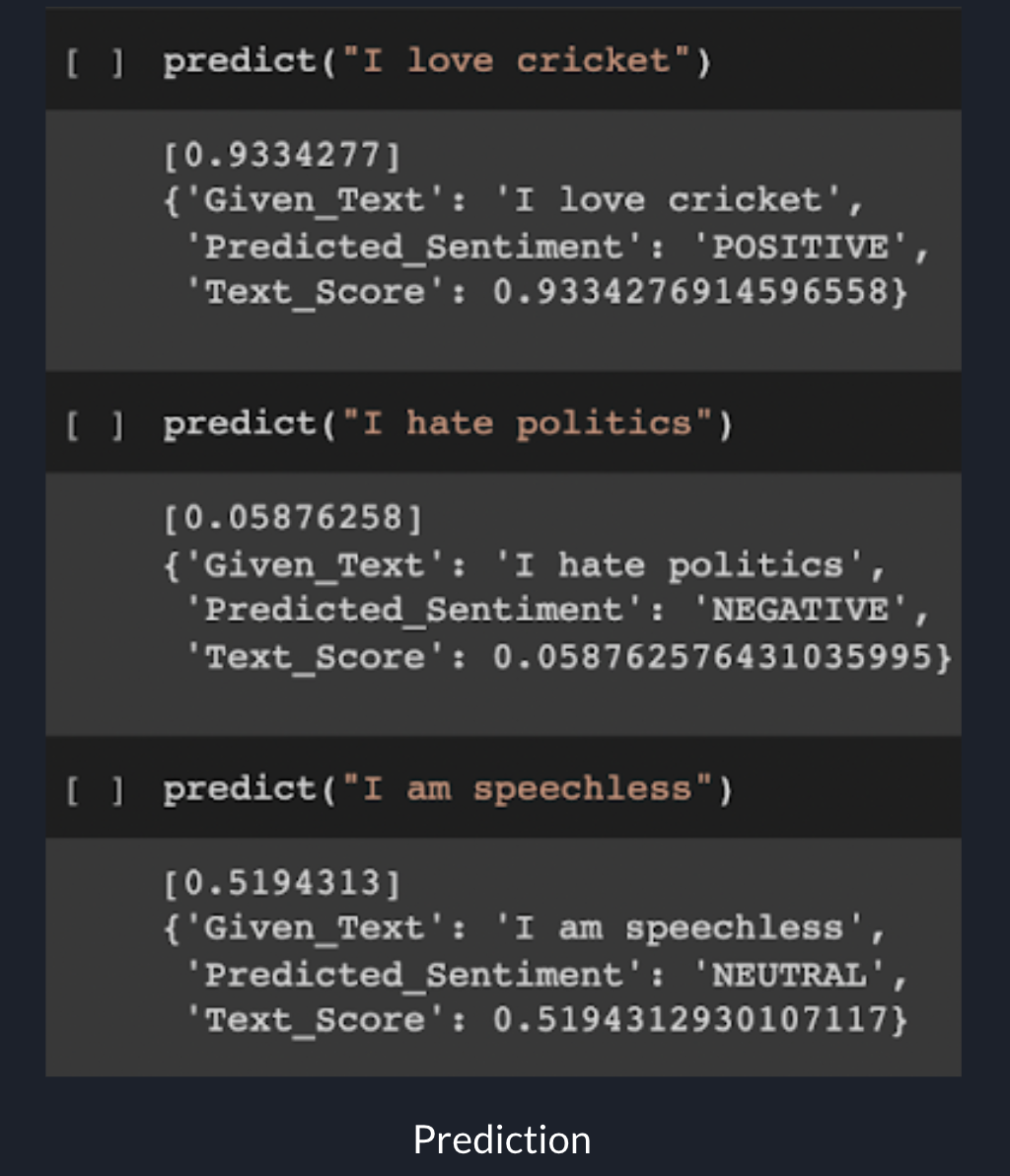


**ANALYSIS AND RESULT**

* Dataset The dataset has a Train set of 1280000 tweets and the test set has 320000 tweets. In the training data, the data has been classified based on the polarity of tweets into negative and positive then they were assigned 0 and 1 respectively. It has been seen that there are 12890 are positive tweets and 5795 are negative tweets. The tweets used for training have been gathered through twitter developer API using the Hashtags related to Indian politics since 2019 excluding the retweets. The testing data is the unclassified set of tweets used for testing purposes to fit into the Machine Learning model.
* Evaluation Metrics F1 score is being used as the evaluation metric. It is the weighted average of Precision and Recall. Therefore, this score takes both false positives and false negatives into account. It is suitable for uneven class distributionproblems.
* The important components of F1 score are:

1. True Positives (TP) - These are the correctly predicted positive values which means that the value of the actual class is yes and the value of the predicted class is also yes.
2. True Negatives (TN) - These are the correctly predicted negative values which mean that the value of the actual class is no and the value of the predicted class is also no.
3. False Positives (FP) – When actual class is no and predicted class is yes.
4. False Negatives (FN) – When actual class is yes but predicted class in no.
5. Precision = TP/TP+FP
6. Recall = TP/TP+FN
7. F1 Score = 2(Recall Precision) / (Recall + Precision)





**CONCLUSION**

Sentiment analysis (also known as opinion mining) is one of the many applications of Natural Language Processing. A set of methods and techniques used for extracting subjective information from text or speech, such as opinions or attitudes. In simple terms, it involves classifying a piece of text as positive and negative.

This paper introduces the combination of the Word2Vec and Keras model for performing sentiment analysis on twitter data. Initially, created a twitter developer API and using the details of the API such as Consumer Key (API Key), Consumer Secret (API Secret), Access Token, Access Token Secret extracted tweets using the searchTwitter function and collected 1600000 tweets. Classified data into positive and negative tweets using the inbuilt function. Then performed data cleaning and data pre- processing on the classified data. Later, applied the Word2Vec feature selection model for extracting features from cleaned tweets. Finally, used these feature sets to build a model for sentiment analysis using the Keras Sequential model. Our approach achieves 80.8% for Bag-of-Words, 81.2% for TF-IDF and 83.9% for Word2Vec. This shows the importance of word embedding in dealing with NLP problems, especially in sentiment analysis. Further, Parts-of- Speech tagging, word2vec with various combinations of machine learning algorithms can be analyzed in future work to improve the accuracy of sentiment analysis for large scale real-time social media data on various platforms.

REFERENCES

[1] https://medium.com/analytics-vidhya/maths-behind-word2vec-explained-38d74f32726b

[2] Sten Andler. 1979. Predicate path expressions. In *Proceedings of the 6th. ACM SIGACT-SIGPLAN Symposium on Principles of Programming Languages (POPL '79)*. ACM Press, New York, NY, 226-236. DOI:https://doi.org/10.1145/567752.567774

[3] Ian Editor (Ed.). 2007. *The title of book one* (1st. ed.). The name of the series one, Vol. 9. University of Chicago Press, Chicago. DOI:https://doi.org/10.1007/3-540-09237-4.

[4] David Kosiur. 2001. *Understanding Policy-Based Networking* (2nd. ed.). Wiley, New York, NY..

Conference Name:ACM Woodstock conference

Conference Short Name:WOODSTOCK’18

Conference Location:El Paso, Texas USA

ISBN:978-1-4503-0000-0/18/06

Year:2018

Date:June

Copyright Year:2018

Copyright Statement:rightsretained

DOI:10.1145/1234567890

RRH: F. Surname et al.

Price:$15.00