International Journal of Emerging Trends & Technology in Computer Science (IJETTCS)
Web Site: www.ijettcs.org Email: editor@ijettcs.org
Volume 5, Issue 2, March - April 2016 **ISSN 2278-6856**

Machine Learning Algorithm For Sentimental Analysis of Twitter Feeds

[1]Mr. Mane Mayur R. , [2]Mr. Kalambate Akshay R., [3]Mr. Rane Zilu Ramkrishna, [4]Prof. Gamare P. S.

[1]B. E. Computer, Department of Computer Engineering, RMCET, Ambav
Mumbai University

[2]B. E. Computer, Department of Computer Engineering, RMCET, Ambav
Mumbai University

[3]B. E. Computer, Department of Computer Engineering, RMCET, Ambav
Mumbai University

[4]Assistant Professor, Department of Computer Engineering, RMCET, Ambav
Mumbai University

Abstract
A sentiment can be defined as a personal positive or negative feeling. Opinion mining is the computational technique for extracting data, classifying it then understanding, and assessing the opinions expressed in various contents. Huge amount of data is generated daily on various social networking sites. Millions of people are posting their likes, dislikes, comments about anything daily on social networking sites. This paper discusses an approach where a publicized stream of tweets from the Twitter microblogging site are preprocessed and classified based on their emotional content as positive, negative and neutral and algorithm which is used to classify these sentiments. Algorithm performance is improved by reducing words in tweet to their root form through mechanism of pre-processing before passing them to sentiment analyzer. Hence, the algorithm classifies tweets as neutral, positive or negative with respect to a query term. This is very useful for the companies and other organizations who want to know the people's opinion about their products or the customers who want to get the feedback from others about product before purchase or also for election exit polls.

I. INTRODUCTION

People are now using new forms of communication. Some of them area micro-blogging sites and text messaging services. These technologies have emerged and become more popular. While there is no limit to the range and size of information conveyed by tweets and texts, often these short messages are used to share ideas, opinions and sentiments that people have about what is going on in the world around them.
Sentiment analysis is a procedure where the dataset consists of emotions, attitudes or assessment which takes into account the way a human thinks. In a sentence, trying to understand the positive and the negative aspect is a very difficult task. The data features used to classify the sentences should have a very strong adjective in order to summarize the review.

This paper focuses on designing algorithm with an acceptable performance that could be used to classify tweets based on the expressed sentiment as neutral or positive or negative. Even though there have been a number of researches which have attempted to classify tweets based on the sentiment stemming from various aspects, an accuracy of 80 per cent or more has never been achieved. Therefore, this area attracts many researches which focus on improving the accuracy and performance of classification techniques. Sentiment analysis in Twitter is a different paradigm compared to other researches that attempt sentiment analysis through machine learning. This is due to constraints which are present in identifying the sentiments expressed in tweets. Due the limitation of characters, people frequently use shortened forms and abbreviations which could bring different interpretations in different contexts.

The implemented algorithm analyses the tweets and classifies them to positive, negative sentiments. But what it does additionally is- it classifies the tweets into neutral sentiment as well, if there is some neutral sentiment.

The rest of the paper is organized as follows: In section II, algorithm processing is discussed. Section III gives sample analysis of a random tweet; section IV presents conclusion.

2.ALGORITHM

A.DATA

Raw Tweets

This dataset is the collection of tweets which are extracted using flume. These tweets are in the unstructured data format. The dataset is full of many non-English words along with technical substitutions. This dataset needs to be prepossessed for the effective analysis. This part is done in Preprocessing module.

Dictionaries

The algorithm maintains dictionaries which are effectively the training sets for the classification purpose. The training set is backbone of the algorithm- more strong the training set, more accurate is the analysis of sentiments.

The dictionaries we've implemented are basically yaml files. It is a markup format used for building up dictionary. There are six main dictionaries built for algorithm. They are:

positive.yml, negative.yml, neutral.yml, inv.yml, inc.yml and dec.yml.

PRE-PROCESSING

Due to the varying and unpredictable nature of language used in tweets, it is likely that preprocessing mechanism [3] could be used to standardize certain tokens of tweets. It is highly likely that most tweets contain some form of acronym, grammatical or spelling mistakes, colloquialisms and slangs; incorporated into due to the 10,000-character limit imposed by Twitter on tweets.

The quality of the data affects the results and therefore in order to improve the result, the raw data is pre-processed. It deals with the preparation that removes the repeated words and punctuations and improves the efficiency of analysis algorithms.

The pre-processing module extracts the relevant content from the tweets while leaving out the irrelevant ones. The techniques applied in this paper are used commonly in information retrieval applications specifically in sentiment analysis in micro-blogging. The collected data is passed through a series of pre-processors.

Some of the pre-processing steps that have been carried out are explained below.:

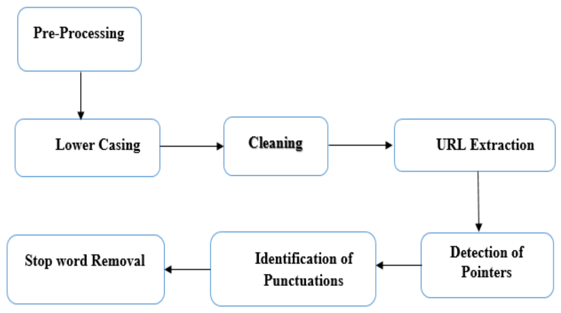

Figure 2: Pre-processing Module

Cleaning

By using a list of cut off patterns, we omit contact addresses and formatting in order to extract only the textual components, smileys,

Lower Casing

It is important to have the entire word in a consistent case when classifying texts in order to guarantee that all tokens map to the corresponding feature irrespective of casing. This is extremely important for this research work as it is very common to find irregular casing(such as "TwITteRseNtlMeNtanAIYsiS") in micro-blogs.

URL Extraction

Many tweets contain URLs in order to share more content than what can be given in the limited-character post. The content in the URL might provide supplementary knowledge regarding the emotion a user trying to express, however it would be far too expensive to crawl URLs for their content. In order to trim down the feature size during training, all URLs in the training tweets have been replaced with an equivalence class <URL>. This could considerably reduce feature size.

Detection of Pointers (usernames and hashtags)

In Twitter, posts can point to other users with the use of anL @ token in front of a username. And users tag tweets pertaining to a category in twitter, using #. Again, to avoid explosion of features, we abstract it to a constant symbol <USER> and <HASHTAG>. This replacement of usemames and hashtags reduce the feature size by a large margin.

Identification of Punctuations

In micro-blogging space, it is common to use excessive punctuation in order to stay away from proper grammar and to communicate emotion more easily. The punctuations can also give insight to the polarity of the message. For example, exclamation marks are used to express powerful emphasis which are usually polar messages [13]. In this step, irrelevant punctuations marks were removed by replacing<PUNCT> to avoid redundant feature in the training set.

Stop word Removal

Words without a deeper meaning, such as the, is, of, are named stop words and can thus be removed. We use a list of stop words.

TOKENIZATION

We segment tweets by splitting it by spaces and punctuation marks, and form a bag of words. That is each tweet is split into sentences and single words named tokens.

PART-OF-SPEECH TAGGING

One of the earliest steps in the processing of natural language text is part of speech (POS) tagging. Usually this is a sentence-based process and given a sentence formed of a sequence of words, part-of-speech tagging tries to label (tag) each word with its correct part of speech.

The mechanism of classifying words into their parts of speech and labeling them accordingly is known as part-of-speech tagging, POS-tagging, or simply tagging. Parts of speech are also known as word classes or lexical categories. The collection of tags used for a particular task is known as a tag set.

A part-of-speech tagger, or POS-tagger, processes a sequence of words, and attaches a part of speech tag to each word:

```
>>> text = word_tokenize("And now for something completely different")

>>> nltk.pos_tag(text)

[('And', 'CC'), ('now', 'RB'), ('for', 'IN'), ('something', 'NN'),

('completely', 'RB'), ('different', 'JJ')]
```

Here we see that and is CC, a coordinating conjunction; now and completely are RB, or adverbs; for is IN, a preposition; something is NN, a noun; and different is JJ, an adjective.

DICTIONARY TAGGING

Dictionary tagging is implemented by dict_tagger module. The sole purpose of dictionary tagging is to lookup for the arrival of tokenised word in the predefined training sets.

Initially, it opens all the dictionaries and keep them ready for mapping. The lookup of the extracted token is then followed in every opened dictionary. As soon as the token is found in either of the dictionaries, the tagger notifies the presence of the token in corresponding dictionary. After successfully tagging, it returns tagged sentences.

Lemmatization

In computational linguistics, stemming refers to the process that reduces inflected words to their stem. This process is carried out by the lemmatiser() module. The lemmatisation results in the drawing out a derived word to it' s root form. This is an important step in NLP.

```
downcase=[i.replace(i,wnl.lemmatize(i,'a')) for i in nouns]
```

Calculation of Sentence Score

In this module, following dictionaries serve the most important role of calculating sentiment score:- inc.yml, dec.ml and inv.yml.

If a tagged token is found to be of positive sentiment, the score is added multiplicatively. On contrast, if the token is found as of negative sentiment, the score is subtracted divisionally. If word which changes the polarity of a sentence are encountered, then the token is looked up in inv.yml dictionary and the score is inverted multiplicatively.

```
if 'inc' in previous_tags:

    token_score *= 2.0

elif 'dec' in previous_tags:

token_score /= 2.0

elif 'inv' in previous_tags:

token_score *= -1.0
```

Storing of Result

Finally, overall sentiment score of a tweet is calculated and is stored in a separate output file. This file is an input for the Graph Rendering module, which renders an infographic format of sentiment analysis.

3.RESULT

Consider a sample tweet to be analyzed

This project is just an biggest piece of work!!

The first step carried out is the tokenisation of the sentence.

[['This', 'project', ʻis', 'just', 'an', ‘biggest’, ‘piece’, ‘of’, ‘work', '!'], [ʻ!']]

The tokenized words are stored in a list in python.

The second step is to convert the tokens to possible root word by lemmatizer() function:

[['This', 'project', ‘is', 'just’, ‘an’, u’big', 'piece', 'of', 'work', '!'], [ʻ!ʼ]]

In next step, every token undergoes part of Speech tagging by POSTagger module:

```
[('This', 'This', ['DT']),

('project','project', ['NN']),

('is', 'is', ['VBZ']),

('just', 'just', ['RB']),

('an', 'an', ['DT']),

('biggest', u'big', ['JJ']),

('piece', 'piece', ['NN']),

('of', 'of', ['IN']),

('work', 'work', ['NN']),

('!', '!', ['.'])],
[('!', '!', ['.'])]]
```

Then the POSTagged tokens are looked up in dictionaries or training set to find out it’s sentiment.

```
[('This', 'This', ['DT']),

('project', 'project', ['NN']),

('is', 'is', ['VBZ']),

('just', 'just', ['RB']),

('an', 'an', ['DT']),

(u'big', u'big', ['positive', 'JJ']),

('piece', 'piece', ['NN']),

('of', 'of', ['IN']),

('work', 'work', ['NN']),

('!', '!', ['.'])],

[('!', '!', ['.'])]]
```

Once the tokens are successfully classified by the algorithm, a sentiment score is calculated for whole sentence:

```
score = sentiment_score(dict_tagged_sentences)
```

The value of a score is among -1, 0 and 1. Accordingly the newly arrived token is added to the corresponding dictionary.

Finally, the overall result for the score is stored to a text file name Result-Tweets.txt.

```
open('Result'+x+'.txt','a').write(str(score)+'\n')
```

This file serves as an input to the graph rendering module.

4.CONCLUSION

The tweets received for the analysis are in the unstructured format, which proves difficulty to analyse, as they contain URLs, non-English words and other non-essential parts. Thus the unstructured tweets are preprocessed with the help of Preprocessing module, to obtain clean tweets. The implemented algorithm has it’s backbone on training set. The accuracy of analysis is thus depend on the sttronger training set. Importatnt modification is an introduction of ‘Neutral’ class, for classifying neutral sentiments. Thus, Preprocessing, training set and Neutral class are responsible for improving performance and accuracy of algorithm

REFERENCES

[1]. Ana C.E.S. Lima, “Automatic Sentiment Analysis of Twitter Messages”, 2012

[2]. Geetika Gautam, “Sentiment Analysis of Twitter Data Using Machine Learning Approaches and Semantic Analysis”

[3]. BalakrishnanGokulakrishnan, “Opinion Mining and Sentiment Analysis on a Twitter Data Stream”, 2012