International Journal of Latest Engineering and Management Research (IJLEMR)
ISSN: 2455-4847
www.ijlemr.com || Volume 07 – Issue 12 || December 2022 || PP. 41-46

Real-time Sentiment Analysis with Tweepy

Swati Malik
Assistant Professor,
Department of Electronics and Communication Engineering,
Maharaja Surajmal Institute of Technology, New Delhi

Abstract: Sentiment analysis also popularly known as Opinion Mining is a way of extracting and analyzing the sentiments, opinions, perception, attitude etc., of a person towards various products, topics, and services. This paper is based on devising a technique of real-time sentiment analysis on live tweets using a pre-trained score based machine learning model. To achieve the above operations Regex cleaning was used and polarity of the sentiment for the cleaned tweet was assigned using a pre-trained machine learning text blob model. In this paper we have used polarity to classify the sentiment of the tweet into different categories of varied positivity and negativity.
Keywords: Sentiment, machine learning, classification, tweets

1. Introduction

Sentiment analysis is a task of Natural Language Processing (NLP) which focuses on extracting opinions and sentiments from the text [1, 2]. The novel opinion mining techniques have incorporated the data from text and other methods like visual data [3,4]. Sentiment analysis and affective computing are the prime components for the development of Artificial Intelligence. Also, they tend to have great potential when applied in various systems and domains. Text classification problem [6-8] can be considered as a task of sentiment analysis as this process involves numerous operations that end up in classifying positive or negative sentiment. Although, it seems to be quite simple process but it considers many subtasks of NLP such as subjectivity detection and sarcasm [9, 10]. Also, the text is not always present in newspapers or books [11, 12] and can contain many idiomatic expressions, orthographic mistakes or abbreviations.

Nowadays, Sentiment analysis has found its place not only among the researchers, but also in governments, companies and organizations.The excessive use of social media platforms has made the web universal and most important source of information. Millions of people express their sentiments and opinions in blogs, forums, wikis, social media handles and other web resources [13-15]. Those opinions hold importance in our daily lives and hence we need to analyze this data to monitor public opinion like, twitter posts were used to predict election results [16]. This paper is based on devising a technique of real-time sentiment analysis on live tweets using a pre-trained score based machine learning model. To achieve the above operations Regex cleaning was used and polarity of the sentiment for the cleaned tweet was assigned using a pre-trained machine learning text blob model. In this paper we have used polarity to classify the sentiment of the tweet into different categories of varied positivity and negativity.

2. Proposed Technique

In order to perform sentiment analysis, data manipulation is required via a processing chain. On this matter, in the early stage of the project, a support module was developed. The support module, referred from now on as *the pipeline*, had to be capable of integrating and testing the following components:

1. Data Gathering Modules
2. Data Filtering Modules
3. Association Rules Modules
4. Sentiment Classification Modules

Together, the pipeline and the aforementioned components form the engine of the system. One of the early objectives of this paper was developing a working model of the engine. As a consequence, the user interface was produced in later development stages. The architectural overview is illustrated in Figure 1 The following subsections will cover in detail the design and the requirements gathering for each component of the system.

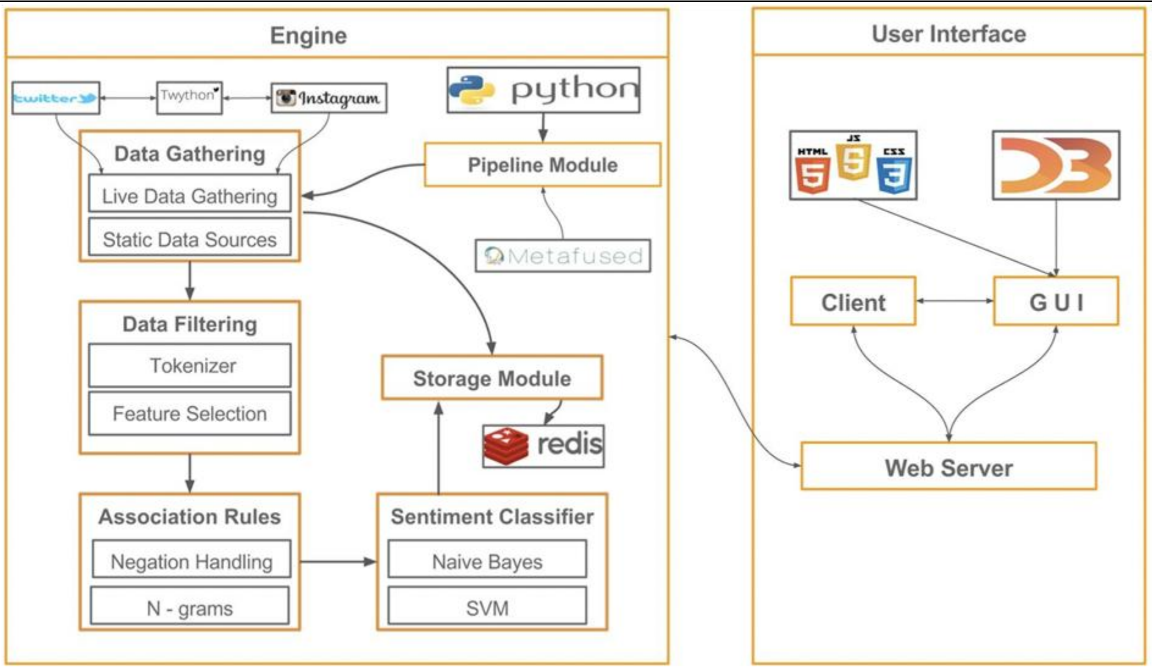


Figure 1 Architecture Overview

The design and development stages were completed in small iterations. This is a core part in the agile methodology which was extensively applied throughout this work. The time available for development was divided into smaller chunks resembling the iterations, each having a deadline and a set of tasks to be completed. Furthermore, GitHub was used as a version control platform. At the end of each iteration a new branch containing the tasks done was created. If the code passed the tests, the current branch was merged with the master branch. If the code did not pass the tests, the code remained unmerged until errors / conflicts were resolved. To help with the progress tracking,**JIRA**, a third party software offering an implementation of the “Tasks Board” agile practice, was used.

Requirements, both functional and non-functional, were gathered at different stages during the development process, contrary to a classical up front approach. However, defining a base set of requirements and expected behavior was necessary to begin development. As such, the engine had to be scalable, allowing integration of new modules without affecting its performance. In addition, a standard structure of the processed data was defined, using JSON (JavaScript Object Notation) formatting, where data objects were stored using attribute-value pairs.

A non-functional requirement prior to the development stage, was researching the state of the art in sentiment analysis. From this perspective the accuracy of the algorithms used for classification, the flexibility of implementing heuristics on such algorithms, and the time required for implementation and testing were assessed. For example, a system using Neural Networks requires more time for implementation compared to one where Naïve Bayes is used. In addition, Neural Network algorithms require in depth understanding in order to perform heuristics, while a probabilistic model like the Naïve Bayes algorithm is more flexible.

In order to achieve the real time behavior desired, a reliable Twitter stream software had to be used in the development of the engine, as Twitter does not provide a native Python API. In these regards, a *“pure Python wrapper for Twitter API”*- Tweepy was employed. As there are a number of Python wrappers available for Twitter’s API (e.g: Tweepy, Twitter Search, Birdy, etc.), the feature which made Tweepy most appealing was the ability of supporting a multi-threaded implementation. This way, the application supports more than one user at a time, and also to manage multiple requests from the same user without having to run multiple instances of the engine.

Furthermore, the engine has to support both the real time component, and the long-term sentiment analysis component. Consequently, Redis was the first choice, as it proved to be an efficient solution for both temporary and permanent storage, being particularly useful at caching operations in the long-term component.

The pipeline had to provide an environment ready to process data, while integrating the other modules. Furthermore, the pipeline had to be flexible so that it could easily adapt data streams. One common issue when processing data streams with auxiliary use of data structures, is that the random access memory space available might be exceeded. This can lead to system failures which are least desirable. Relevant to this issue, was a built-in feature of Python: functions generators. Instead of returning a set of values stored in a data structure, a function can be written so that it generates the values as they are needed by other components in the processing chain.

While using generators presents a set of advantages, there is one drawback to be mentioned. Once data was generated for usage it can only be called once. Not as bad as it sounds, this obliged the code to be written in a careful manner by avoiding redundancies, auxiliary variables and data structures, in order to preserve the efficient memory use. Another important feature of the pipeline is to ensure the suitability of an object for the purpose it was initially designed for. From this perspective, the pipeline had to serve data structures from one module to another, while ensuring that those objects preserved their properties. The end result of the pipeline is a class which once instantiated serves as support for the other modules in the processing chain.

2.1 Data Gathering

The main data source used throughout the project is the online social networking service, Twitter. Two data gathering modules were implemented using Tweepy. One component was designed to be used for the real time interface. Once a request is placed, it will further send the request to Tweepy, which in turn will reply with a continuous stream of tweets.

The other component collects data within a pre-established interval of time. As marketing is concerned with development and implementation of a promotional strategy, the time factor had to be carefully considered when performing analysis. Ten minutes' worth of data on a campaign might not justify the success rate of it. Considering the amount of time available for the development of the project, waiting an entire week to perform analysis over a set of data was not feasible either. In order to meet the requirements, while having enough time to test and adjust the sentiment analysis process, the second module was used to store data within the following intervals: ten minutes, one hour, and two days. This way, adjustments of other components is possible in an efficient amount of time, while the amount of data analysed will deliver an accurate success estimation of a campaign. Furthermore, an auxiliary module was developed to format the labelled data obtained from a third party source - "Sentiment140".

2.2 Data Filtering I

For data filtering one module was built, which performs tokenization splitting textual input in tokens (words). The tokenization module implements auxiliary methods to detect and discard tweets which are not entirely made out of English characters (e.g.: Japanese, Hiragana, Katakana, etc.). However, tweets containing other symbols (e.g.: mentions marked with the symbol "@" and hashtags, marked with the symbol "#") are filtered in a separate JSON field.

2.3 Data filtering II

The above mention technique does not result in higher accuracy rates. Later in the development, as a solution to improve the accuracy of the classifier, another filtering module was build to be used in the training stage of the classification. The problem identified was the large number of tokens produced by the tokenization module, which equalled almost 1 million. This made the model to be computationally expensive, reducing the speed performance of the algorithm. In addition, two approaches were considered:

- Using Porter's Stemming algorithm to perform word reductions.
- Excluding tokens with low sentiment value

While Porter's algorithm reduced the corpus size to 60% of its initial size after tokenization, this was not sufficient, as a significant number of the tokens left, held no particular sentiment value. Verifying manually which tokens were prevalent in positive or negative contexts was not a solution, due to the high amount of manual work. An interesting new heuristics using the pareto principle (explained in Chapter 2) was applied. Looking at the tokens distribution of the training model Figure 2, it can be noticed that words such as "I" and "the" have the highest occurrence throughout the whole input data. At the opposite end, are words which albeit, they might present some sort of sentiment value, the number of total occurrences in the corpus is low; consequently the impact over the module is almost irrelevant.

However, when separated from their context, those words do not hold any sentiment value. By contrast, words situated on the curve's slope in the figure, appeared to have a strong sentiment value (e.g.: "love", "hate", "amazing" etc.). Following the pareto principle, the corpus was cut to 20% of its size, so that only the tokens with a high sentiment value were kept. A manual inspection was required to determine where tokens with low sentiment value start to appear in the distribution, in order to determine the cut points. As a result, the final size of the model was reduced down to 40.000 tokens. Consequently, the accuracy was improved by 7% as a result of noise (unwanted data) removal.

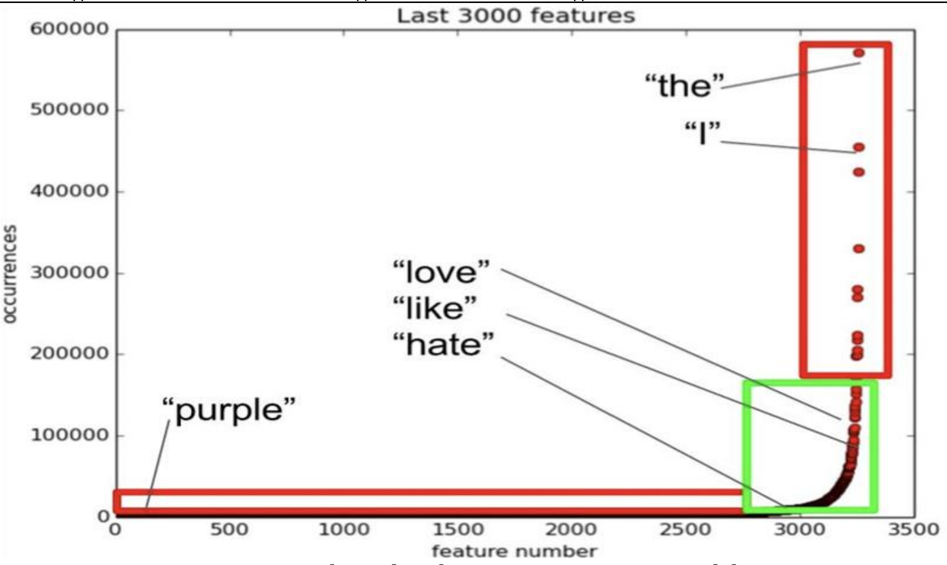


Figure 2 Tokens distribution - Naïve Bayes model

2.4 Association Rules

Another common improvement in computational linguistics is the use of n-grams for which a module was created. However, implementing n-grams in the form of bigrams and trigrams, without considering only relevant associations, increases the size of the corpus and also adds noise to it. As such, associations between nouns, adjectives and verbs were done using a third party software, polyglot, which offered a part of speech tagging tool.

3. Result Analysis

In the presented section results of the proposed technique have been presented. Once data was acquired and filtered, the next step was the implementation of the classification algorithm, the main module of the engine. The module implemented the Naïve Bayes algorithm, which consists of two phases: training and testing. In the training phase, labelled data acquired from Sentiment140 was used. As such, 1.2 million tweets out of the 1.6 million size of the corpus was used in training. The output is a model consisting of token-value pairs, where the values are: number of occurrences over the entire dataset, number of occurrences in positive labelled tweets, and number of occurrences in negative labelled tweets. This is exemplified in Table 1.

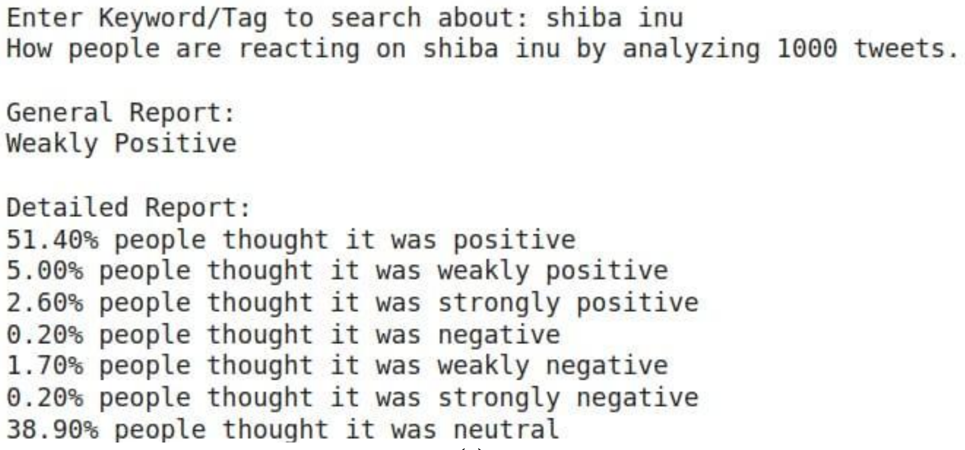
Table 1: Sample of the model obtained in the training phase of the classification

Token	Positive counts	Negative counts	Total counts
more	11426	11320	22746
fleet	16	61	77
whole	17	12	29
woods	97	75	172
spiders	33	88	121
like	1087	542	1629
woody	26	15	41
loving	931	12	943
sigh	8	105	113

In the testing phase, unseen labelled data and the probabilistic model produced in the training phase were used to measure the accuracy of the algorithm. Consequently, the algorithm produced a classification accuracy of 71%. For the development stage when it was implemented, such accuracy was expected, yet further improvements were required.

As the data set used in training had labels only for positive and negative tweets, yet the goal is to distinguish between the three: positive, negative, neutral, further heuristic was required. In addition, a sentiment evaluation scale: -1 (negative) to 1 (positive) was introduced, where tweets of which sentiment score was in the range of: [-0.05, 0.05] were considered neutral. The limits of the range were established after a manual inspection of the tweets analysed.

As an improvement bigrams were added which shows that the overall accuracy of the classifier grew by 2.9% up **to 80.9%.** In addition to n-grams, a separate module was build to handle negation. A classical approach in specialized literature is to add artificial words: *"if a word x is preceded by a negation word (e.g: not, don't, no), then rather than considering this as an occurrence of the feature x, a new feature (token) NOT x is created"*. For example after performing negation handling the sentence: "I do not like this new brand" will result in the following representation: "I do not NOT_like, NOT_this, NOT_new, NOT_brand". The advantage of this feature is that the plain occurrence and the negated occurrence of the word are now distinguishable in the corpus. In addition, the accuracy of the classifier was improved by 1.8% up to **82.71%** overall accuracy. Figure 3 shows the data visualization of the proposed technique.

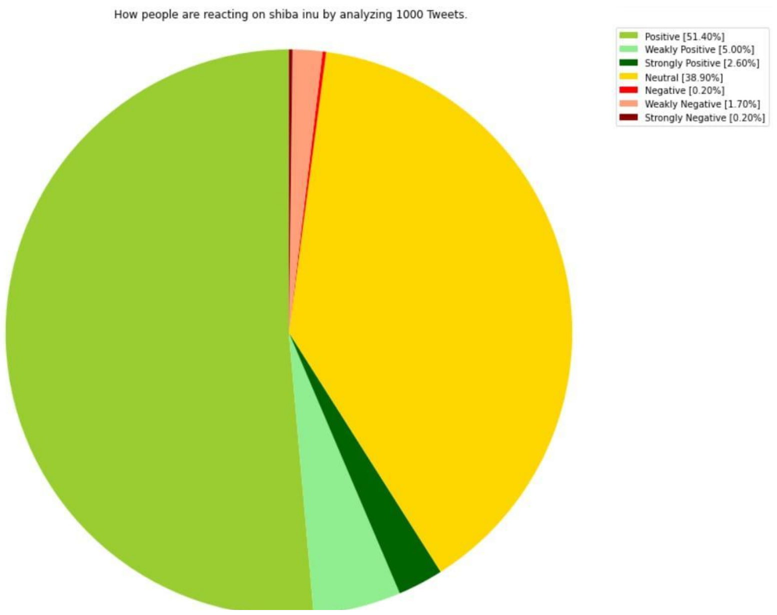


```
Enter Keyword/Tag to search about: shiba inu
How people are reacting on shiba inu by analyzing 1000 tweets.

General Report:
Weakly Positive

Detailed Report:
51.40% people thought it was positive
5.00% people thought it was weakly positive
2.60% people thought it was strongly positive
0.20% people thought it was negative
1.70% people thought it was weakly negative
0.20% people thought it was strongly negative
38.90% people thought it was neutral
```

(a)



(b)

Figure 3 (a) Detailed opinion analysis on the keyword shibainu (b) pictorial visualization of people's reaction on the keyword shibainu

4. Conclusion

This paper presented a low cost and efficient technique for real-time sentiment analysis using Tweepy. This method can also be used in an integrated way to make a publicly accessible tool with high-level abstraction. This method finds application as a data mining tool and also can be used to study the research market trends.

References

[1] B. Liu, Sentiment Analysis, 2015, pp. 1–367, https://doi.org/10.1017/ CBO9781139084789.
[2] I. Chaturvedi, E. Cambria, R.E. Welsch, F. Herrera, Distinguishing between facts and opinions for sentiment analysis: Survey and challenges, Inf. Fusion. 44 (2018) 65–77, https://doi.org/10.1016/j.inffus.2017.12.006.
[3] S. Poria, E. Cambria, R. Bajpai, A. Hussain, A review of affective computing: From unimodal analysis to multimodal fusion, Inf. Fusion. 37 (2017) 98–125, https://doi.org/10.1016/j.inffus.2017.02.003.
[4] J.F. Sánchez-Rada, C.A. Iglesias, Social context in sentiment analysis: For- mal definition, overview of current trends and framework for comparison, Inf. Fusion. 52 (2019) 344–356, https://doi.org/10.1016/j.inffus.2019.05. 003.
[5] E. Cambria, Affective computing and sentiment analysis, IEEE Intell. Syst. 31 (2016) 102–107, https://doi.org/10.1109/MIS.2016.31.
[6] B. Schuller, A.E.D. Mousa, V. Vryniotis, Sentiment analysis and opinion mining: On optimal parameters and performances, Wiley Interdiscip. Rev. Data Min. Knowl. Discov. 5 (2015) 255–263, https://doi.org/10.1002/ widm.1159.
[7] Y. Choi, H. Lee, Data properties and the performance of sentiment classification for electronic commerce applications, Inf. Syst. Front. 19 (2017) 993–1012, https://doi.org/10.1007/s10796-017-9741-7.
[8] S. Ju, S. Li, Active Learning on Sentiment Classification By Selecting Both Words and Documents, 2013, pp. 49–57, https://doi.org/10.1007/978-3- 642- 36337- 5_6.
[9] E. Cambria, D. Das, S. Bandyopadhyay, A. Feraco, eds., A Practical Guide To Sentiment Analysis, 5, 2017, pp. 1–196, https://doi.org/10.1007/978- 3- 319- 55394- 8.
[10] A. Valdivia, M.V. Luzón, E. Cambria, F. Herrera, Consensus vote models for detecting and filtering neutrality in sentiment analysis, Inf. Fusion. 44 (2018) 126–135, https://doi.org/10.1016/j.inffus.2018.03.007.
[11] M. Erritali, A. Beni-Hssane, M. Birjali, Y. Madani, An approach of semantic similarity measure between documents based on big data, Int. J. Electr. Comput. Eng 6 (2016) 2454–2461, https://doi.org/10.11591/ijece.v6i5. pp2454- 2461.
[12] M. Birjali, A. Beni-Hssane, M. Erritali, Measuring documents similarity in large corpus using MapReduce algorithm, in: 2016 5th Int. Conf. Multimed. Comput. Syst, IEEE, 2016, pp. 24–28, https://doi.org/10.1109/ ICMCS.2016.7905587.
[13] F.J. Ramírez-Tinoco, G. Alor-Hernández, J.L. Sánchez-Cervantes, B.A. Olivares-Zepahua, L. Rodríguez-Mazahua, A Brief Review on the Use of Sentiment Analysis Approaches in Social Networks, 2018, pp. 263–273, https://doi.org/10.1007/978- 3- 319- 69341- 5_24.
[14] S. Mukherjee, Sentiment analysis of reviews, in: Encycl. Soc. Netw. Anal. Min, Springer New York, New York, NY, 2017, pp. 1–10, https://doi.org/ 10.1007/978- 1- 4614- 7163- 9_110169- 1.
[15] M. Rushdi Saleh, M.T. Martín-Valdivia, A. Montejo-Ráez, L.A. UreñaLópez, Experiments with SVM to classify opinions in different domains, Expert Syst. Appl. 38 (2011) 14799–14804, https://doi.org/10.1016/j.eswa.2011. 05.070.
[16] B. O'Connor, R. Balasubramanyan, B.R. Routledge, N. A. Smith, from tweets to polls: Linking text sentiment to public opinion time series, in: W.W. Cohen, S. Gosling (Eds.), Proc. Fourth Int. Conf. Weblogs Soc. Media, The AAAI Press, 2010, pp. 1–8.