IJCRT.ORG ISSN : 2320-2882

INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS (IJCRT)

An International Open Access, Peer-reviewed, Refereed Journal

Implementation and designing a framework for detecting fake news in social media

1Kavita N. Jadhav, 2Dr. Avinash S. Kapse
1Student, 2Head of Department & Assistant Professor
1Anuradha Engineering college, chikhli, dist buldana,
2Anuradha Engineering college

Abstract- It is always better to verify the news and then make it circulation in the social media. But every time it is not possible the people and social media platform failed to verify the post and due to which many aspects of disturbances in public occurred. And after considering the above problem the proposed system helps to rectify this problem by designing a framework which helps to detect the fake news content with due respect to the dataset available. Considering the above reflection on the project's planning, it can be concluded that the scope of the project was met within the timeline set. As such, the desired classification accuracy of 80% or more is reached. The reporting is performed in real time and the graphical user interface offers an intuitive, yet comprehensive user interaction. Overall, the project was a good opportunity to further hone my programming skills and expand my knowledge in the fields of natural language processing and machine learning. By designing and developing this project, my overall knowledge and skills in computer science were significantly improved. As such, a better understanding of the machine learning field was gained, while implementing the classification algorithm and the subsequent heuristics to improve its accuracy. In addition, fundamental knowledge in the field of natural language processing was acquired.

Keywords: Classification, Computer, Intuitive, Programming, Processing

1. Introduction

Sentiment Analysis is a term that you must have heard if you have been in the Tech field long enough. It is the process of predicting whether a piece of information (i.e. text, most commonly) indicates a positive, negative or neutral sentiment on the topic. In this article, we will go through making a Python program that analyzes the sentiment of tweets on a particular topic. The user will be able to input a keyword and get the sentiment on it based on the latest 100 tweets that contain the input keyword. Also known as "Opinion Mining", *Sentiment Analysis* refers to the use of Natural Language Processing to determine the attitude, opinions and emotions of a speaker, writer, or other subject within an online mention. *Essentially, it is the process of determining whether a piece of writing is positive or negative. This is also called the Polarity of the content.* As humans, we are able to classify text into positive/negative subconsciously. For example, the sentence "The kid had a gorgeous smile on his face", will most likely give us a positive sentiment. In layman's terms, we kind of arrive to such conclusion by examining the words and averaging out the positives and the negatives. For instance, the words "gorgeous" and "smile" are more likely to be positive, while words like "the", "kid" and "face" are

really neutral. Therefore, the overall sentiment of the sentence is likely to be positive. A common use for this technology comes from its deployment in the social media space to discover how people feel about certain topics, particularly through users' word-of-mouth in textual posts, or in the context of Twitter, their *tweets*.

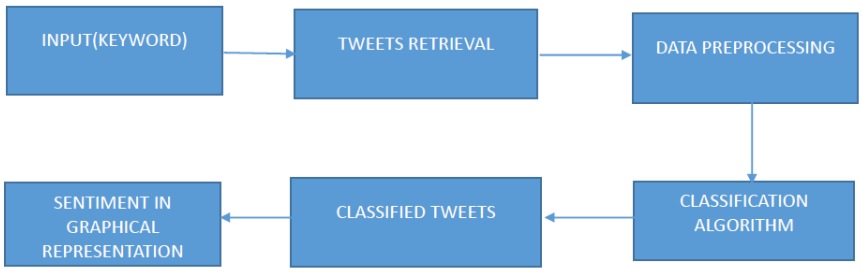

Fig. 1 Flow of the concept

2. System Architecture

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 the project 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 3.1. The following subsections will cover in detail the design and the requirements gathering for each component of the system.

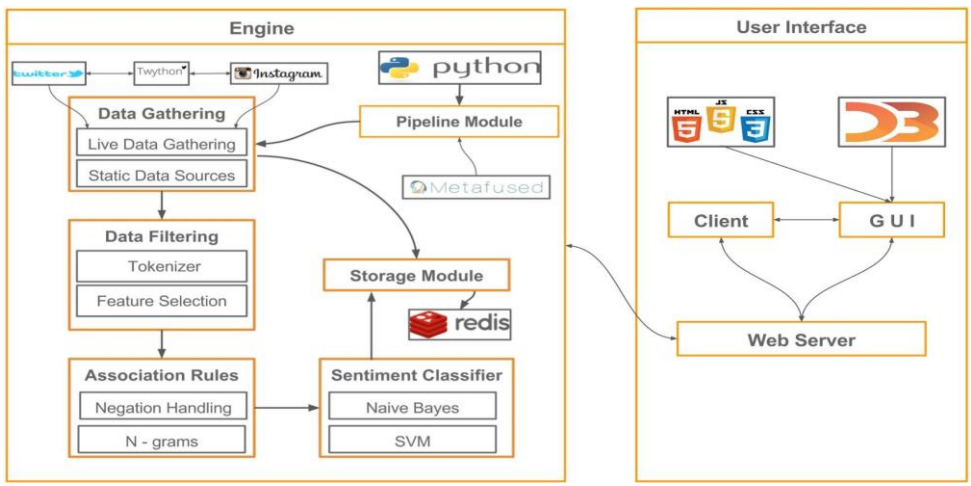

Fig. 2 Architecture overview

3. Methodology

The project's design and development stages were completed in small iterations. This is a core part in the agile methodology which was extensively applied throughout the project. 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.

No.	Functional Requirements	Priority	Hours	Development Stage
1	Train the main classification algorithm	Very High	15-20	Early
2	Output and store the model after the training phase of the classification algorithm	Very High	5-10	Early
3	Test the classification algorithm proposed	Very High	15-20	Early
4	Clean the noise from tweets retrieved and address orthography issues	High	10-15	Middle
5	Retrieve a stream of tweets (domain specific for specified topics) for the long-term components within intervals of : 10 minutes, 2 hours, 2 days	Very High	10-15	Middle
6	Store the acquired data depending on the component in which the engine is used	High	10-15	Middle to Late
7	Train an additional machine learning algorithm for comparison with the main algorithm	Low	12-15	Late

Table 1: Engine - functional requirements

4. Implementation

4.1 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.

4.2 Classification - Naïve Bayes Algorithm

Once data was acquired and filtered, the next step was the implementation of the classification algorithm, the main module of the engine. As outlined in Section 3, 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 4.1.

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

Table 2: Sample of the model obtained in the training phase of the classification

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.The following subsection describes the implementation of heuristics to improve the performance and the classification accuracy of the algorithm.

4.3 Data filtering II

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 equaled almost 1 million. This made the model to be computationally expensive, reducing the speed performance of the algorithm. In addition, two approaches were considered:

1. Using Porter's Stemming algorithm to perform word reductions.
2. 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 4.3, 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.

4.4 Graphical User Interface

As described in Chapter 3, the graphical user interface was implemented for the real time component. In order to add interactivity to the system, the language of choice to build the front-end was JavaScript. Consequently, a third party library - D3JS, was used to build interactive charts. Albeit there is a variety of templates already available within the library, an implementation from scratch was necessary to meet the requirements of the system. As such, the visual component was challenging due to the lack of previous experience in JavaScript (more details in the Conclusion chapter). Captures of the graphical user interface are presented in Appendix B.

5. Result

Screenshots

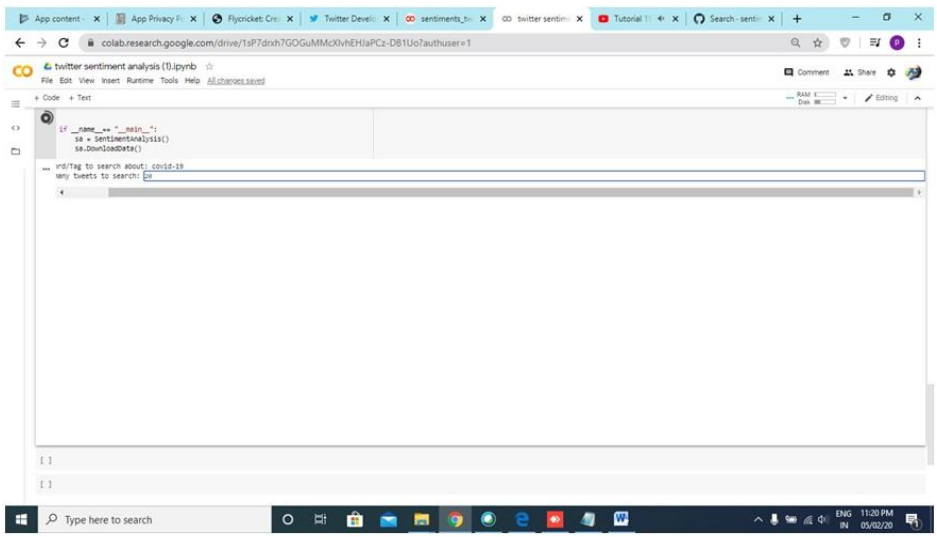

Fig.3 Importing the package

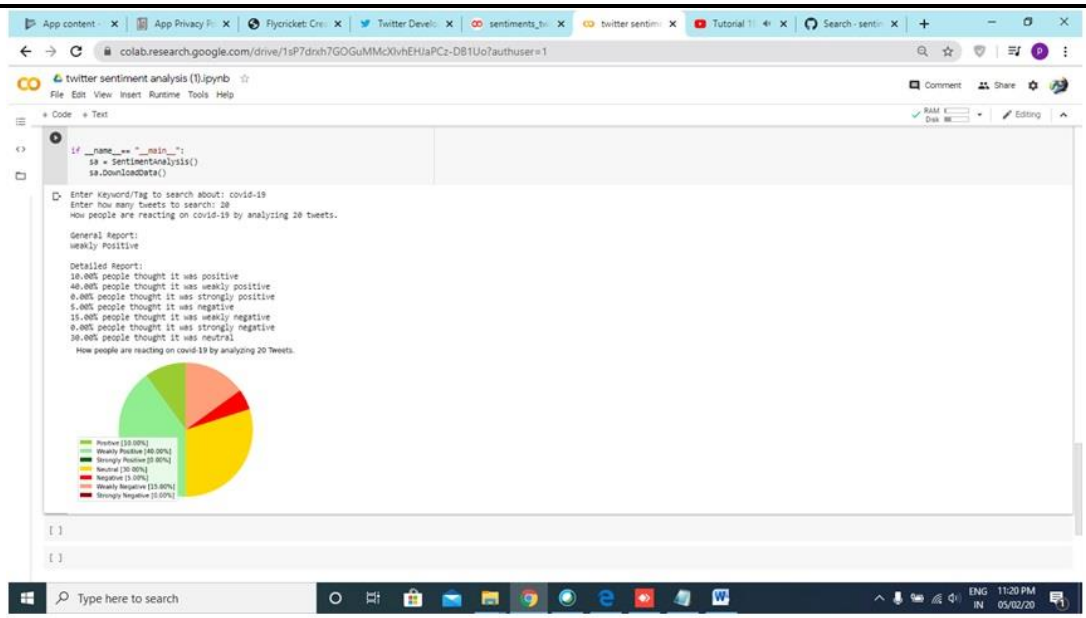

Fig.4. Retrieving the tweets and category of the tweets according to polarity as positive, negative and neural

6. Conclusion

Considering the above reflection on the project's planning, it can be concluded that the scope of the project was met within the timeline set. As such, the desired classification accuracy of 80% or more is reached. The reporting is performed in real time and the graphical user interface offers an intuitive, yet comprehensive user interaction. Overall, the project was a good opportunity to further hone my programming skills and expand my knowledge in the fields of natural language processing and machine learning. By designing and developing this project, my overall knowledge and skills in computer science were significantly improved. As such, a better understanding of the machine learning field was gained, while implementing the classification algorithm and the subsequent heuristics to improve its accuracy. In addition, fundamental knowledge in the field of natural language processing was acquired.

REFERENCES

[1] SisiraNeti, S. (2011). SOCIAL MEDIA AND ITS ROLE IN MARKETING. 1st ed.[ebook]International Journal of Enterprise Computing and Business Systems. Available at: http://www.ijecbs.com/July2011/13.pdf [Accessed 13 Apr. 2016].

[2] Shatkay, H. and Craven, M. (2012). Mining the biomedical literature. Cambridge, Mass.: MIT Press.Nlp.stanford.edu. (2016). Stemming and lemmatization. [online] Available at: http://nlp.stanford.edu/IR-book/html/htmledition/stemming-and-lemmatization-1.html [Accessed 17 Apr. 2016].

[3] Busino, G. (ed) (1965) Oeuvres complètes. Vilfredo Pareto 1848-1923. Geneva: Droz.Russell, Stuart, Norvig, Peter (2003) [1995]. Artificial Intelligence: A Modern Approach (2nd ed.). Prentice Hall. ISBN 978-0137903955.

[4] Konstantinova, N. (2014). Machine learning explained in simple words - Natalia Konstantinova.

[online] Nkonst.com. Available at:http://nkonst.com/machine-learning-explained-simple-words/ [Accessed 18 Apr. 2016].

[5] Miskovic, V. (2001). Application of inductive machine learning in data mining. Vojnotehnickiglasnik, 49(4-5), pp.429-438.Slideshare.net. (2016). Lucene/Solr Revolution 2015: [online] Available at: http://www.slideshare.net/joaquindelgado1/lucenesolr-revolution-2015-where-search-meets- machine-learning [Accessed 16 Apr. 2016].

[6] Saedsayad.com. (2016). Naïve Bayesian. [online] Available at: http://www.saedsayad.com/Naïve_bayesian.htm [Accessed 21 Apr. 2016].

[7] Docs.opencv.org. (2016). Introduction to Support Vector Machines — OpenCV 2.4.13.0documentation. [online] Available at:http://docs.opencv.org/2.4/doc/tutorials/ml/introduction_to_svm/introduction_to_svm.html [Accessed Apr. 2016].

[8] Codefellows.org. (2016). 5 Reasons why Python is Powerful Enough for Google. [online] Available:https://www.codefellows.org/blog/5-reasons-why-python-is-powerful-enough-for-google [Accessed 23 Apr. 2016].

[9] IMPYTHONIST. (2015). Build massively scalable RESTFul API with Falcon and PyPy. [online] Available at:https://impythonist.wordpress.com/2015/09/12/build-massively-scalable-restful-api-with-falc on-and-pypy/ [Accessed 23 Apr. 2016].

[10] Shalev-Shwartz., (2014). Understanding Machine Learning: From Theory to Algorithms. Cambridge University Press.WhatIs.com. (2016). What is multithreading? - Definition from WhatIs.com. [online] Available at: http://whatis.techtarget.com/definition/multithreading [Accessed 23 Apr. 2016].

[11] Suzanne Embury(2015),List of Suggested Agile Practicea Available at: https://moodle.cs.man.ac.uk/file.php/357/Coursework/ [Accessed 24 Apr. 2016].

[12] Sentiment140.com. (2016). Sentiment140 - A Twitter Sentiment Analysis Tool. [online] Available at: http://www.sentiment140.com/ [Accessed 25 Apr. 2016].

[13] Lextutor.ca. (2016). LISTS DOWNLOAD. [online] Available at: http://www.lextutor.ca/freq/lists_download/ [Accessed 25 Apr. 2016].

[14] Research Blog. (2016). All Our N-gram are Belong to You. [online] Available at: http://googleresearch.blogspot.co.uk/2006/08/all-our-n-gram-are-belong-to-you.html [Accessed 25 Apr. 2016].

[15] Wiegand, M., Balahaur, A. and Montoyo, A. (2010). Proceedings of the Workshop on Negation and Speculation in Natural Language Processing, Uppsala, July 2010, pp. 60–68.

[16] Flask.pocoo.org. (2016). Flask (A Python Microframework). [online] Available at: http://flask.pocoo.org/ [Accessed 28 Apr. 2016].

[17] Kohavi, R. and John, G. (1995). A Study of Cross Validation and Bootstrap for Accuracy Estimation and Model Selection, Stanford CA. 94305 , pp. 2-5

[18] Picard, Richard; Cook, Dennis (1984). "Cross-Validation of Regression Models".Journal of the American Statistical Association 79 (387): pp. 575–583 doi:10.2307/2288403

[19] Metafused Shoots, S., data, H. and Landscape, M. (2015). Metafused Blog. [online] Blog.metafused.com. Available at:http://blog.metafused.com/search?updated-min=2015-01-01T00:00:00-08:00&updated-max= 2016-01-01T00:00:00-08:00&max-results=3 [Accessed 30 Apr. 2016].

[20] Frank, E. (1998). Naive Bayes for regression. Hamilton, N.Z.: Dept. of Computer Science, University of Waikato.

[21] Yang, Z. (2010). Machine learning approaches to bioinformatics. Singapore: World Scientific.Hammell, T. (2005). Test-driven development. Berkeley, CA: Apress,ISBN-10: 159-0-593-278

[22] Steinwart, I. and Christmann, A. (2008). Support vector machines. New York: Springer.ISBN-13: 978-0-387-77241-7

[23] Arbuckle, D. (n.d.). Learning Python testing. ISBN 978-1847198846

[24] Percival, H. (2014). Test-driven development with Python. Sebastopol, CA: O'Reilly Media. ISBN-13: 978-1449364823

[25] Gorelick, M. and Ozsvald, I. (2014). High performance Python. Sebastopol, CA: O'Reilly. ISBN-13: 978-1449361594