Real Sent Real-time Sentiment Analysis

An implementation of NLP, sentiment analysis and data scraping in real time

[1]Ishan Farooq, [2]Abhist Singh, [3]Apurva Vishvakarma, [4]Hemant Tiwari, [5]Varun Singh
[1]Student, [2]Student, [3]Student, [4]Student, [5]Student
[1]Computer Science & Engineering,
[1]Babu Banarsi Das National Institute of Technology & Management, Lucknow, India.

Abstract: Sentiment Analysis is the process of 'computationally' determining whether a piece of writing is positive, negative or neutral. It's also known as opinion mining, where we derive the opinion or attitude of a speaker/writer towards a concept. Sentiment Analysis is one of the most popular applications of Natural Language Processing (NLP), which we can do which in turn leverages Machine Learning (ML) models depending on the implementation of the application and the exact needs.

Here in this project, we have tried to computationally determine the positive emotions and negative emotions based towards any concept/person/thing using Sentiment Analysis in real time. The prime objective of this web application is to provide the users who can potentially be anyone from individuals to organizations, freelancers, etc. with the power to easily access the sentiments of people around the world on literally anything be it a product, person, concept, etc. as long as it is being discussed on the widely popular and extensively used social media and micro blogging platform Twitter.

Index Terms **– Sentiment Analysis, Real-time, Natural Language Processing, Twitter, Naive Bayes.**

I. INTRODUCTION

Our project is built upon Python as the base due to the fact that there are extensive number of open source libraries available on Python that we can leverage in our project. In addition to that, we use libraries like Dash to render our real-time graphical user interface for the user to interact with. This enables us to update the user interface in real-time providing user with a rich user experience. We use intuitive graphs and color coding to display our data and insights to the user making the whole experience easy to access and interact with.

The prime objective of this project is to provide the users who can potentially be anyone from individuals to organizations, freelancers and even general public with the power to easily access the sentiments of people around the world on literally anything be it a product, person, concept, etc. as long as it is being discussed on the widely popular and extensively used social media and micro blogging platform Twitter. This can be used to enable companies to know what people are talking about their products when they talk about it. This is a very valuable insight for companies releasing products and wanting to know more about the public reaction towards it.

1.1 What is RealSent?

RealSent is a tool designed specifically to be used by users to get detailed insights about any search term entered by the user. It provides the real-time analysis about the sentiments of the public on that particular term. In addition to that, it also provides a longer-term analysis and overall analysis to the user.

1.2 How does RealSent work?

With user's input term, we store that term in our database and search the processed tweets texts to match that term. This is done using Full Textual Search (FTS) and the results are given to the user in real-time. It takes the twitter stream and stores all the tweets after cleaning, processing and analyzing them enabling them to be retrieved as the user types in real-time.

1.3 How will RealSent tool save time?

As the user enters the search term, he/she gets the insights on that particular term instantaneously. This makes the manual and resource extensive (both time and manpower) work of sentimental analysis automatic by collecting data from twitter and processing it while retrieving it and then storing in the database. This makes the whole system work in real-time.

1.4 What makes RealSent different?

Two words describe what makes RealSent different: Accessibility and speed. Currently, if a user wants such insights, he/she has to pay for using such currently available tools. RealSent provides the insights that individuals or companies may want without and charges and in real-time so that such data is accessible to all without having to pay or sign-up for any kind of service.

1.5 How much data can RealSent process?

RealSent can process approximately 3.3 million tweets in a day and store them in its database. This is increased everyday as the system is left to run, with ~3.3 million tweets being added per day. This can be increased by using a different tier of Twitter API and further increasing the computational power of the system.

II. TECHNICAL DETAILS

This section focuses on the technology stack used in the backend and frontend of the system.

- PYTHON: The whole application is built from ground up in Python programming language. This is primarily because of the vast option of libraries available and its efficiency for tasks such as Natural Language Processing.
- DASH: Dash UI is a framework in Python to develop and support web based user interfaces that can update in real-time efficiently with least possible computing overhead. This is chosen as the tool itself is updated in real-time.

- TextBlob: This is a library which harnesses the power of Natural Language Processing to be used in the system. This uses a Naïve Bayes classifier under the hood to assign a sentiment value to the words to help in sentiment analysis of the tweet as a whole.
- HTML & CSS: These are used to give the basic structure and styling to the web-app. The layout is written in HTML inside the Python files that is then rendered by the Dash framework.

III. DATABASE DESIGN

SQLite3 is an easy to use and light weight Database Engine and its module is a part of the python standard library which are embedded into Python. This module is a SQL interface which is compliant with DB-API 2 specification standard. SQLite3 supports the standard relational database features like the SQL syntax, transactions, prepared statements, truncating, triggers, and it is also self-contained, & server-less. It is very fast and lightweight, and the entire database is stored in a single disk file making it a good choice for this system. In addition, it requires only little memory at runtime which is beneficial for system that operates in real-time. SQLite3 supports the data types TEXT, INTEGER, and REAL. All other types should be converted to one of these types before saving them in the database to avoid any related issues. SQLite3 itself does not validate if the types written to the columns are actually of the defined type or not and we should provide the check ourselves to ensure the same.

3.2 SQLite3 Architecture

3.2.1 SQLite3 Module

The module SQLite3 contains all general classes and methods for working with databases. It provides methods to open, query, update, commit, delete, close, and other operations on the database. More specifically, SQLite3 module provides the insert(), update() delete(), and select() methods which are used extensively in this system.

3.2.2 FTS in SQLite

Since the system is real-time and involves searching and matching of terms from millions of existing entries, it is essential to employ a searching method that is equally fast and efficient while consuming least possible resources. FTS3 and FTS4 are SQLite virtual table modules that allows us to perform full-text search on a set of documents stored in the database table, even if the table contains many large documents.

One of the most common way to describe what really a full-text search is "what Google, and other search engines do with documents placed on the World Wide Web". Here users input a term, and the full-text query system finds the set of documents that best matches that term considering the operators and groupings the user has specified (if any).

IV. APPLICATION COMPONENTS

The main components of the application are the backend that handles the sentiment analysis and the application UI. The backend handles tasks that include fetching the tweets from the internet, cleaning them, analyzing them, storing them to the database, and also retrieving them when the user searches for a term. The front-end renders the real-time updating UI and shows the results in an intuitive manner. It also handles the user input that is the search term and sends it to the back-end to fetch the results. It consists of all the graphs, and the lists showing the related tweets.

4.1 Backend

4.1.1 NLP & Sentiment Analysis

For this, TextBlob library is used as it serves the purpose quite well. It is used to first filter out the stop-words from messing with the polarity of tweet and to find better-related terms which are more relevant. The tweets are then analyzed for their sentimental value and are assigned a polarity score accordingly between -1 to 1. For each tweet, stop-words, punctuations are scrapped as they can hinder the polarity score and the relevant terms which are generated for the given term. Each word is given a weight/penalty. Negative words are given negative values and positive words are assigned positive values. In addition to this, Naive Bayes classifier is used to classify them as positive or negative using this library.

4.1.2 Related trends analyzer

To generate the most relevant and most used terms with the input term, all the possible nouns are extracted from the current tweet and stored, this is done for all the tweets. Next, these are ranked and the ones occurring the most number of times are used. Also, the tokens are separated and the nouns are stored in a mapped data structure for each of the keyword entered every time a new search is performed in the database. Not storing these words lowers the disk usage and have a minimal impact on response time as storing these for each search term is a futile approach in the sense that a keyword may only be entered once and storing data for it would just take more disk space.

4.2 Application UI

4.2.1 Search component

This is the only module that requires user input. The user types in a search term and the application updates in real-time to show the corresponding data to the user. This is under the hood implemented using React for real-time data flow to the backend and FTS in the database.

4.2.2 Real-Time sentiments graph

This module shows the real-time sentimental values associated with a tweet and the volume of the tweets per second on a line & bar mixed graph. This updates in real-time every second.

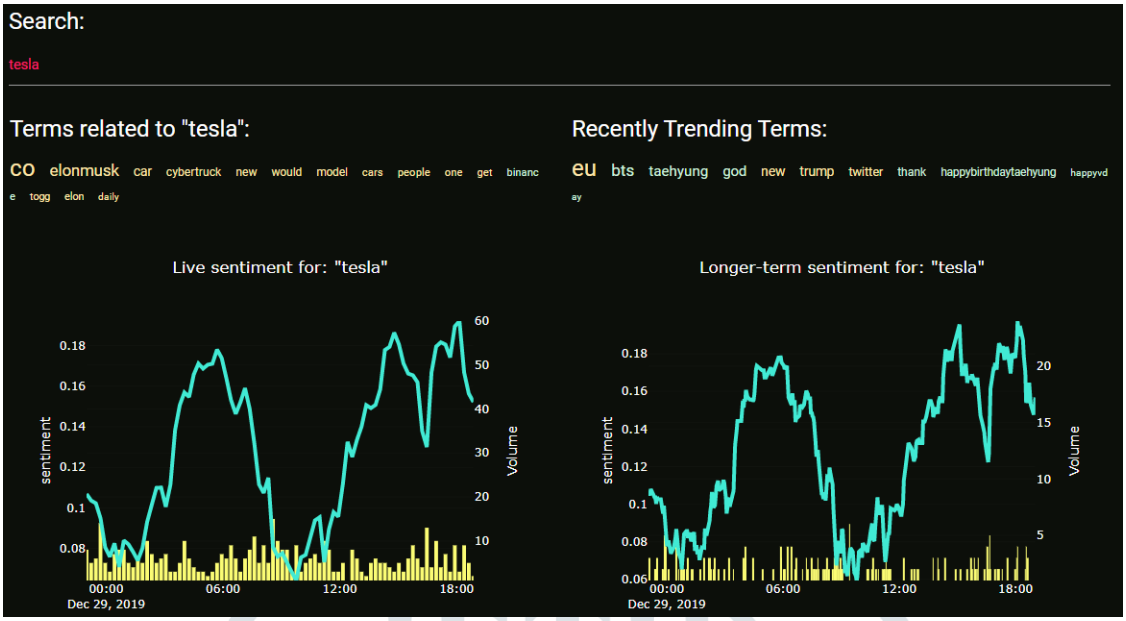

Figure 1 Real-Time graphs components

4.2.2 Longer term sentiments graph

This module shows the longer-term sentimental values associated with a tweet and the volume of the tweets on a line & bar mixed graph. This updates every 3 minutes and shows a total of 5 hours of the sentimental and volume data of the given term.

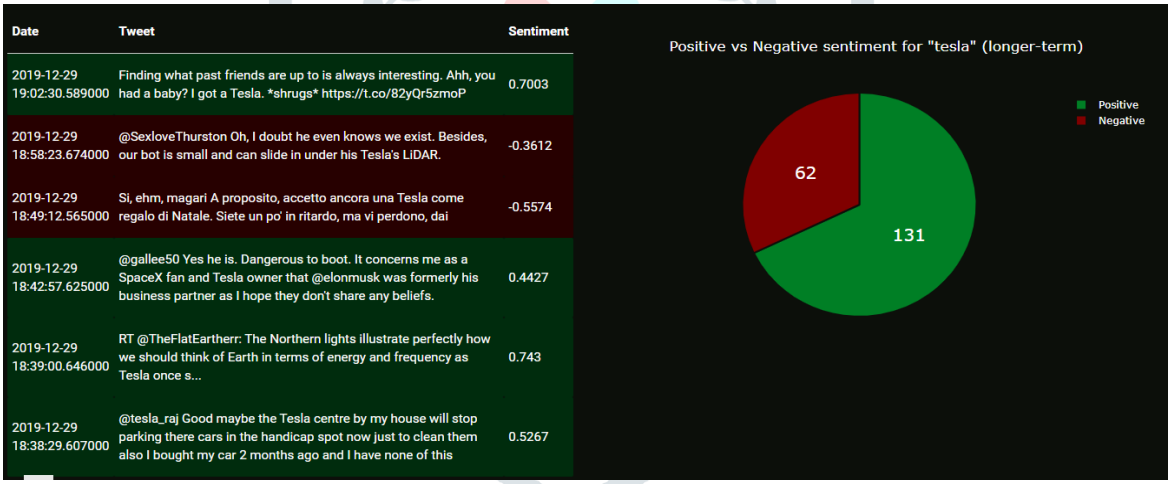

Figure 1 Real-Time graphs components

4.2.3 Live tweets feed

This module shows the real time tweets that are being analyzed with the sentimental values associated with it. These are also color-coded for easier readability. These update in real-time every second or as a new tweet that is relative to the term comes in.

4.2.4 "Related To" terms component

This module shows the related terms people are talking about when they talk about the term being queried. This gives valuable insight, especially when used for a product. This is updated as the terms change or the term is updated in the search bar. These are also color-coded and of variable size. The larger size signifies a more frequently used term.

4.2.5 "Trending" terms component

This module shows the real-time trending terms on the twitter platform. This is updated as the terms change on the Twitter platform itself. These are also color-coded and of variable size. The larger size signifies a more frequently used term.

4.2.5 Positive vs Negative chart component

This module shows the longer-term positive VS negative sentiments of the public on the given term visualized on a color-coded and annotated pie chart. This is updated every 3 minutes with the longer-term graph.

V. WORKFLOW

We have made the app in a way that it makes interacting with it incredibly easy. All it requires is an input term and that's all there is to it. The whole process is shown in the flow diagram below.

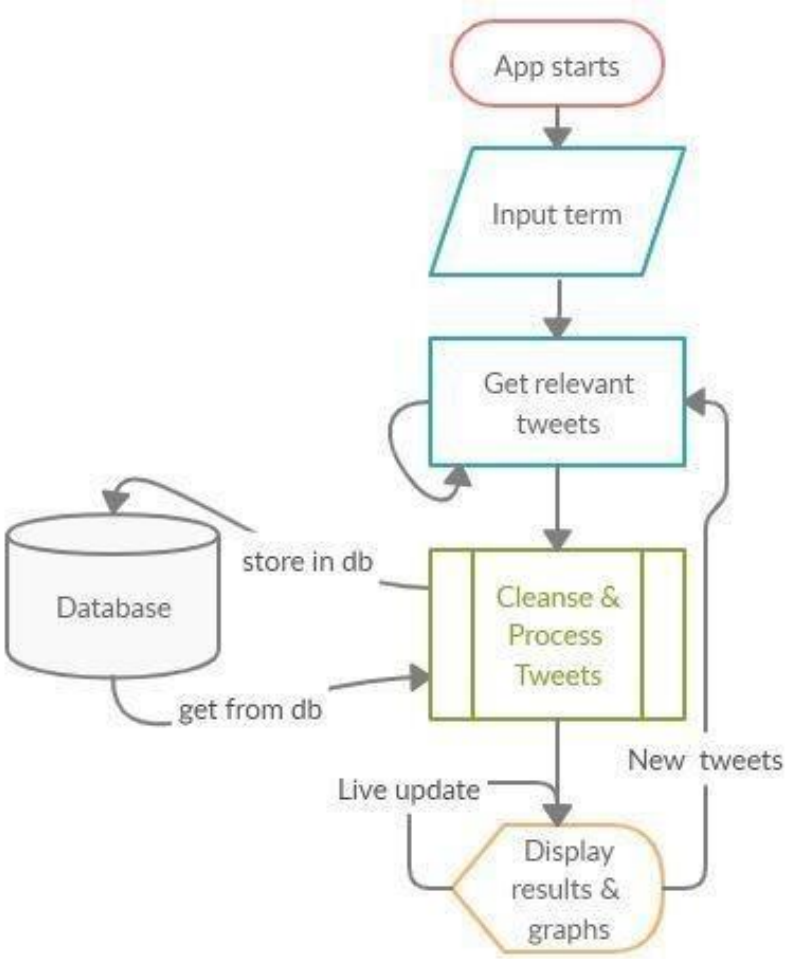

Figure 3 Workflow

As seen in the flowchart, the very first step after the input is provided is to find the relevant tweets from the system. Also, before these can be processed, they need to be stored in the database and after processing, they're stored to retrieve them for longer term sentiment analysis. This also includes cleaning the tweets, removing stop words, tokenization, assigning polarity values, etc. These processes are explained on the next page in the order they should be performed

VI. Conclusion

We presented results for sentiment analysis on Twitter. We use previously proposed state-of-the-art unigram model as our baseline and report an overall gain of over 4% for two classification tasks: a binary, positive versus negative and a 3-way positive versus negative versus neutral. In this, we presented a very comprehensive set of experiments for both these tasks on manually annotated data that is a random sample of the stream of tweets. We investigated two kinds of models: tree kernel and feature based models and demonstrate that both these models outperform the unigram baseline. For our feature-based approach, we do feature analysis and it reveals that the most important features are those that combine the prior polarity of the words and their parts-of-speech tags and also of that of the word that follows it. We tentatively conclude that sentiment analysis for Twitter data is not that different from sentiment analysis for other genres. In future work, we will explore even richer linguistic analysis, for example, parsing, semantic analysis and topic modelling.

VII. Acknowledgment

We would first like to thank our Head of Department Prof. Diwakar Yagyasen for letting us choose this project. We would also like to thank and show gratitude to our project coordinator professor Shadab Siddiqui for guiding us in this project and helping out in every way possible. We also take this opportunity to thank our faculties for their time and help and would also thank our parents for supporting us throughout.

References

[1] Dan Klein and Christopher D. Manning. 2003. Accurate unlexicalized parsing. Proceedings of the 41st Meeting of the Association for Computational Linguistics, pages 423–430.

[2] Alessandro Moschitti. 2006. Efficient convolution kernels for dependency and constituent syntactic trees. In Proceedings of the 17th European Conference on Machine Learning.

[3] Alexander Pak and Patrick Paroubek. 2010. Twitter as a corpus for sentiment analysis and opinion mining. Proceedings of LREC.

[4] B. Pang and L. Lee. 2004. A sentimental education: Sentiment analysis using subjectivity analysis using subjectivity summarization based on minimum cuts. ACL.

[5] P. Turney. 2002. Thumbs up or thumbs down? Semantic orientation applied to unsupervised classification of reviews. ACL.

[6] C M Whissel. 1989. The dictionary of Affect in Language. Emotion: theory research and experience, Acad press London.
[7] T. Wilson, J. Wiebe and P. Hoffman. 2005. Recognizing contextual polarity in phrase level sentiment analysis. ACL.
[8] TextBlob library, https://textblob.readthedocs.io/en/dev/
[9] DASH framework, https://plotly.com/dash/
[10] Twitter API, https://developer.twitter.com/en/docs