Twitter Sentimental Analysis using AI:

Shauryan Dave,
Devang Patel Institute of Advance Technology and Research
sndave18@gmail.com

Daksh Chauhan,
Devang Patel Institute of Advance Technology and Research
dakshchauhan06@gmail.com

***Abstract*— Since 2012, there have been several changes in the field of artificial intelligence. The usage of AI I s f or ma r k e t I n g is one of the key changes.**
We now have a very good NLP algorithm after many years of study and development (natural language processing). Today, NLP has a wide range of applications, including autocorrect, word prediction, and text processing. Big firms like Google and Amazon filter a lot of data and analyze it using NLP and other Machine Learning algorithms. Sentimental data analysis can be used to identify flaws in a product and can also assist in the launch of a product in accordance with market trends. The study of NLP and AI is presented in this publication for Sentimental Analysis on Twitter, aimed at non-technical users.

Keywords—Artificial intelligence, Recurrent neural networks, Speech recognition system , Speech to text .

I.INTRODUCTION

The problem is Running sentiment analysis on social media data is an incredibly powerful tool. Marketing teams can understand consumer sentiments about their products, accounting & research teams can forecast sales, while research firms can predict demographic sentiments. However, due to the inherent complexity of parsing large datasets and usage of complex ML and data science techniques, sentiment analysis presents a steep learning curve to non-technical users. This Web app is name as "Market Miner" by us.

The Solution

We aim to provide non-technical users with a simple interface to run sentiment analysis on relevant Twitter datasets, with little compromise in terms of functionality and effectiveness. Since knowledge of programming, data science and machine learning are not required, anyone can leverage this Web app to understand public opinions better.

How It Works

Using sentiment analysis algorithms in themselves are useless. Datasets must first be preprocessed to generate useful insights. Therefore,Market Miner works by guiding users through a 4-step pipeline to achieve a final Report. The steps are (i) data extraction from Twitter, (ii) dataset filtering, (iii) dataset categorization, and finally (iv) sentiment analysis and Report generation.

Market Miner Web App(MM)

I will first introduce the architecture of MM, then provide a step-by-step description of how a MM pipeline works, and what some best practices are. An analysis of the ML and statistical techniques used will be provided later in the report.

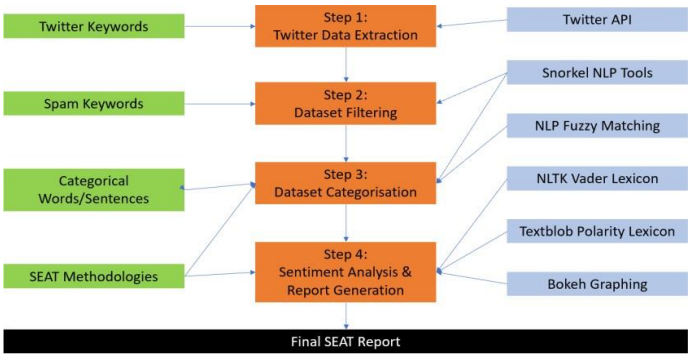

Figure 1 :Overview of Market Miner Pipeline

Web App Architecture

The backend of MM web app is programmed in Python, leveraging the Django web framework. I chose Django because its "batteries included" philosophy provides greater functionality and scalability compared to Flask. Despite its steeper learning curve, my efficiency when using Django dramatically increased with time spent, allowing me to focus more on the features of MM pipeline and improving its user experience.

After completion , w e wi l l deploye ou r Djangoproject on Heroku, while utilising its PosgreSQL service. With this deployment, MM would no longer be a proof-of-concept, but a free tool that everyone can make use of, without needing technical skills.

I. Twitter Data Extraction

The first step builds a firm foundation for every MM(market miner) Report. Users have a certain topic, idea, or character in mind when using MM. They wish to find out public sentiment regarding a topic of interest. However, due to the large volume of Twitter data, it is impossible for users to manually sift through and copy-paste tweets from Twitter's website. So how do we extract large volumes of data, while maintaining precision?

Step 1 automates the extraction of Twitter data, by leveraging its developer API. Users simply needs to select up to 5 relevant keywords and choose the number of tweets they wish to extract. Two measures are in-placeto ensure precision. Firstly, users can choose to ensure results *do not contain* certain keywords. Thus, strictly irrelevant queries will be excluded from the dataset. Secondly, users can attach prefixes to keywords, such as "#" and "@". This feature is especially beneficial when targeting a trend or company/person [4].

However, Step 1 does contain shortcomings. Twitter API has limitations on the number of queries a user can provide. In order to prevent data extraction errors and simply user experience, I have limited the number of queries to 5. This limitation should only be restricted to a small body of users. More significantly, (the free version of) Twitter API only provides historical data for *up to the past 7 days*. The short timeframe prevents us from implementing historical trend analysis features.

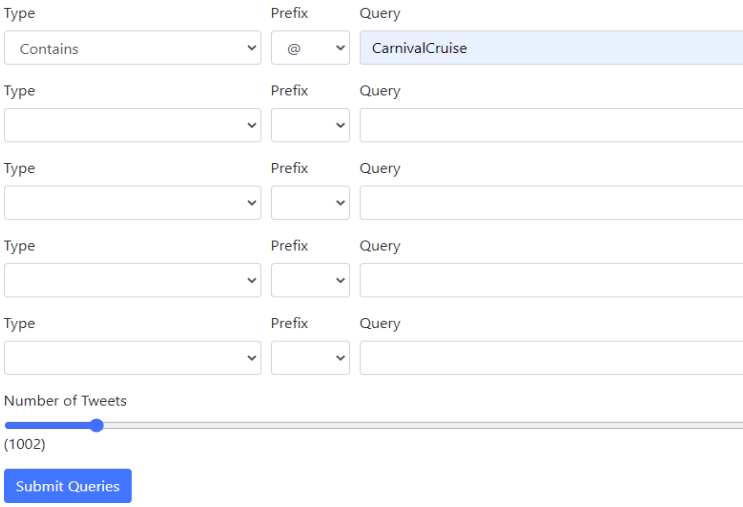

Figure 2: User Interface of Step 1

II. Dataset Filtering

Step 2 allows users to filter their dataset, thus removing irrelevant tweets from affecting sentiment analysis. Despite Step 1's features of removing certain keywords from dataset, it is not comprehensive enough. Irrelevant subtopics can have numerous keywords, despite Step 1 only offering 5 query options. By implementing a separate step to filter our dataset, users can more thoroughly remove irrelevant tweets from clouding results. Users can enter up to 10 keyword groups. Each keyword group represents a type of irrelevant data. Each contains at least one keyword to help MM identify tweetsthat fall into each group.

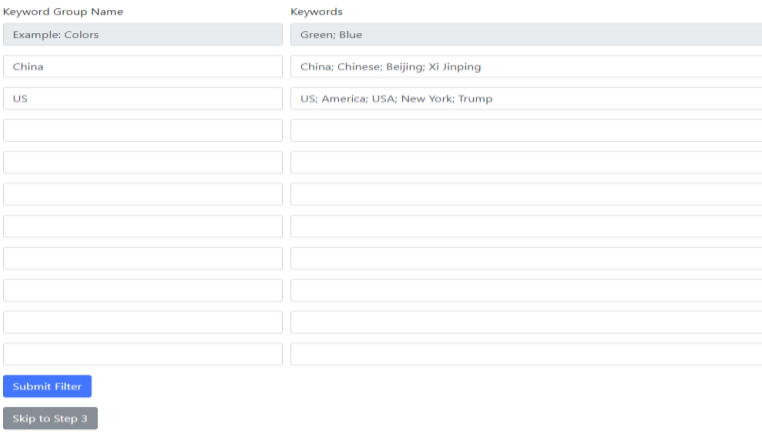

Figure 3: User Interface of step 2

In order to produce a list of irrelevant tweets using user's keywords, MM's backend utilises Snorkel's Labeling Functions to streamline the process. The algorithm's key idea is that when at least one keyword is identified in a tweet, it will be flagged as irrelevant. The keywords must be *standalone* words in the tweet, so keywords which are part of another word would not be flagged. For example,

if the keyword is “virus”, the word “coronavirus” will not be identified. This simple underlying logic allows users to understand how MM works, without compromising on functionality. Snorkel streamlines the identification algorithm with its Labeling Functions and simple use of Regex. Therefore, Step 2’s backend is effective, yet easy to maintain.

III. Dataset Categorisation

Step 3, the final preprocessing step, aims to categorise Twitter data according to what the user requires. Running sentiment analysis on a single body of data is limited in usefulness, since social media commentary on a topic often relates to different subtopics. For example, commentary on Coca Cola can relate to Coke Original, Coke Zero, etc. In order to unlock sentiment analysis’ full potential, it is essential to categorise the data into relevant topics, thus generating more helpful outcomes. Step 3 allows categorisation in a simple, yet effective manner.

There are 2 available methods of categorisation, absolute method and NLP method. Absolute method is like Step 2’s. Categories are matched to tweets based on whether keywords are found in tweet text . Keywords are entered in the same fashion as in Step 2. Once again, Snorkel’s Labeling Functions will take care of the categorisation. Please note that not all tweets would be categorised, and some tweets may be classified into more than one category. Absolute method should be used when,

i. each category can be adequately expressed using these keywords;

ii. you value a simple approach which you can understand easily; and

iii. you are unsure if your categories encompass nearly the entire dataset.

The NLP method uses a completely different methodology. As the name suggests, NLP techniques (tf-idf and k-nearest neighbors algorithm) are used to match tweets to relevant categories using user input. How these techniques work will be detailed in the Machine Learning section below. NLP method should be used when,

i. the categories are broad and cannot be adequately expressed with short keywords;

ii. you have at least 1000 tweets in your dataset; and

iii. you are sure that your categories emcompass nearly the entire dataset.

The NLP method works such that all tweets would be categorised into one, and only one, of your categories. Like most machine learning models, there must be enough data to produce accurate results. While the 1000 tweets minimum is arbitrary, it is a good general rule to follow. Instead of entering several keywords for each category, you will input a body of text (e.g. several senteces) which narrates the given category. Producing enough relevant text is vital for the ML model to work well. Since all tweets will be categorised, regardless of how far-fetched, please ensure your categories represents the dataset well.

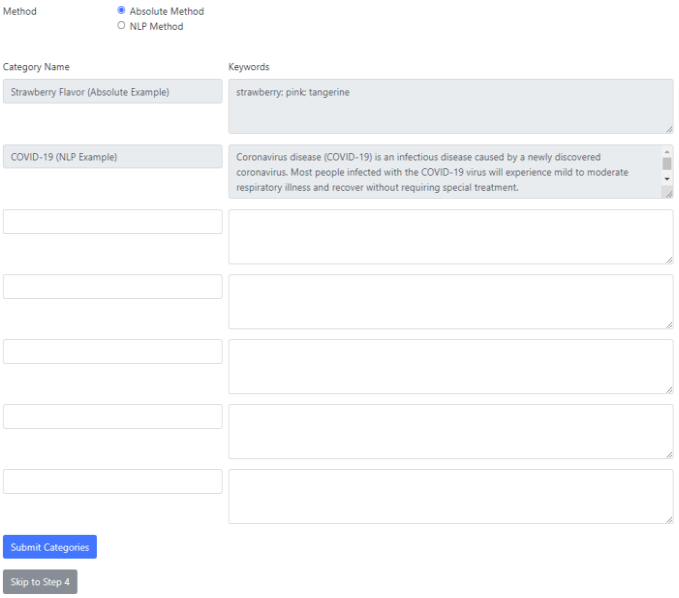

Figure 4: User Interface of step 3

IV. Sentimental Analysis

Finally, we have reached the step of running the sentiment analysis. Despite this being the most crucial step of the process, it is ironically the easiest for users to run. All that they need to do, is to choose one or more of the sentiment analyses options to run on their data. Usersare encouraged to choose all availabe options, since they are unlikely to understand the nuances of each model. MM will aggregate the results of all models, to produce a more accurate sentiment reading.

On the backend, the available two models (NLTK Vader and TextBlob) are both pre-trained ones, so no manual labeling from users to implement as training data is required. How NLTK Vader and TextBlob are trained and their effectiveness will be expanded on in the Machine Learning section. Since the MM pipeline emphasizes simplicity to cater to a non-technical audience, a requirement to label hundreds of tweets manually would go against our philosophy. Thus, pre- trained models are the most viable sentiment analysis option.

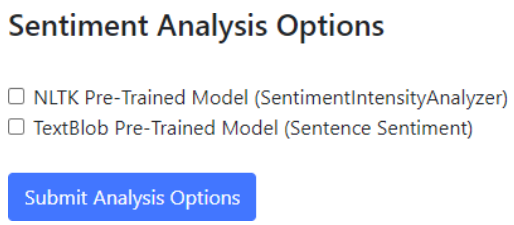

Figure 5: User interface of step 4

V. *Report Generation*

Sentiment analysis is only as powerful as our ability to visualize and analyze it. Even though sentiment analysis has already run on the dataset, the creation of MM report is nonetheless vital to the process. Users will rely on this report to generate insights on public sentiment about their topic of interest. The typical MM report consists of a series of pie chart visualizations, based on the different categories and models selected by users.

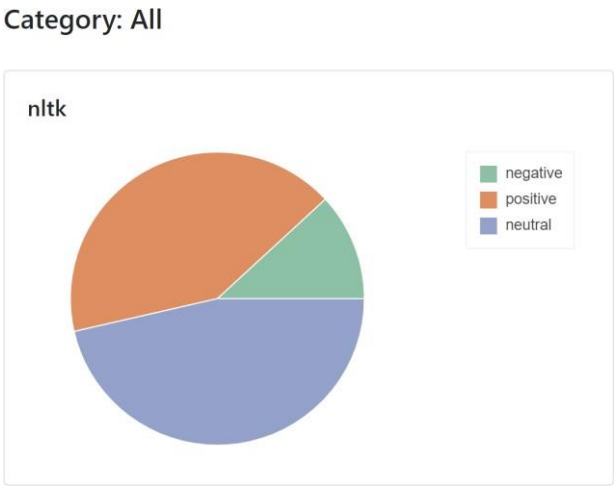

Figure 6: Report Generation

The contraints of being simple and easy-to-use is equally prominent when finally running sentiment analysis. There are 2 options for running sentiment analysis: (i) Use an out of the box, pre-trained sentiment analysis model; or (ii) Training some kind of machine learning or deep learning model using tweets manually labeled by users, such as Long Short-Term Memory (LSTM) and Linear Support Vector Classification (LinearSVC). Machne learning algorithm [3].

However, practical concerns eliminated the "train a model" option. In this [2] , where a data scientist attempted sentiment classification with LSTM, using 239,233 lines of sentences from the Stanford sentiment treebank data. Despite the sophisticated algorithm and enormous dataset, he managed a result of 48.6% accuracy. Training a deep learning model with just a few hundred training data points is bound to fail – not to mention the significant computing and manpower costs.

Pre-trained models, on the other hand, have negligible computing and manpower costs, while possessing reasonable accuracy. A commonly overlooked benefit is its reliability. Since these lexicons are well understood, users can easily account for its limitations, while being assured of a certain level of accuracy. Alternative methods, such as training a LSTM model, could have widely different accuracies, depending on the dataset. Their unpredictability would pose a significant challenge to users, for how can you trust the results if you don't understand the model?

In the final product, users can choose the NLTK Vader model, the Textblob Sentiment model, or both. An aggregate method allows users who do not understand the limitations of each model, to produce an average of both models. According to [1] , VADER is an amalgamation of existing well- established sentiment word-banks (such as LIWC, ANEW and GI) and new lexical features. These new features are created using a wisdom-of-the-crowd (WotC) approach, where lexical intensity ratings are obtained from independent human raters and experts. The dataset consists of tweets, reviews and articles. On the other hand, TextBlob's sentiment analysis model is trained using Naïve Bayes Classification, relying solely on movie reviews. Based on personal experience and online commentary, VADER is geared more towards *informal* texts (which include slang, emojis, etc), while Textblob performs better with regards to *formal* language. This is likely due to the use of social media texts in VADER's methodology.

VI. *Conclusion & Future Use*

Following the goal of creating a sentiment analysis tool for non-technical users, I created a tool that could potentially be useful for marketing (brand monitoring), customer relations (customer feedback), etc.

If provided more time and resources, we would (i) create a monitoring feature where reports would automatically absorb more data from Twitter everyday, (ii) create a time-series feature for historical trend analysis if additional funding is secured, and (iii) create a RESTful API to integrate MM's technologies with other platforms (e.g. HubSpot's Marketing Software).

REFERENCES

1. VADER: A Parsimonious Rule-based Model for Sentiment Analysis of Social Media Text
2. https://medium.com/analytics-vidhya/sentiment-analysis-for-text-with-deep-learning-2f0a0c6472b5
3. https://www-nlp.stanford.edu/courses/cs224n/2009/fp/3.pdf
4. http://cs229.stanford.edu/proj2011/GoelMittal-StockMarketPredictionUsingTwitterSentimentAnalysis.pdf