ENG 678 Final Project

Executive Summary

Dr. Egbert

Sherman (Rui Hu)

**Building a Twitter Scraper and**

**a Prototype Dictionary-based Sentiment Analyzer**

**Introduction**

Sentiment analysis has gained popularity in recent linguistic studies (Liu, 2020). According to Liu (2020), sentiment analysis is the “study of people’s opinions, sentiments, emotion, and attitudes.” There are two types of sentiment analyses, namely machine-learning-based sentiment analysis and dictionary-based analysis (Liu, 2020). In machine learning sentiment analysis, texts are converted to the variables for computational analysis and thus making the texts lose their linguistic underpinnings.

The purpose of the project was to create a prototype dictionary-based sentiment analyzer and investigate the texts from the perspective of corpus linguistics. To be more precise, I will compare the prototype dictionary-based sentiment analyzer against the pre-trained machine learning sentiment analyzer. This investigation is important as it gives researchers a chance to compare the two types of sentiment analyzers and it shed some light on future sentiment dictionary building and corpus linguistic research.

In this executive summary, by following the final project guidelines, I will cover the following topics: research questions (goals), methods (steps I took to accomplish the goal), challenges, and self-reflection. Aside from the requirements from the guidelines, I would like to briefly describe the data source, i.e., the corpus, interpret the collected data, and address the limitations of the prototype.

**Description of the Corpus**

Texts from Twitter were used to test the dictionary-based sentiment analyzer. Biber, Egbert, and Davies (2015) investigated the compositions in the searchable web and pointed out that a “typical web search provides little information about the register of the documents that are searched” (Biber et al., 2015). Fortunately, Twitter contained a wealth of texts with easy-to-distinguish written registers, such as news Tweets and personal mini-blog posts (also referred to as personal Tweets).

After some investigation on personal Tweets posted by influential people, such as Tim Cook, Elon Musk, and Bill Gates, I found negative Tweets were scarce, regardless of which sentiment analyzer I used to analyze the text. Finally, I chose ABC news on Twitter (@abc) as the data source for creating the corpus.

**Research Questions**

The investigation of the sentiment of the Tweets has led me to the following guiding research questions:

1. What is the distribution of the ratings of the collected Tweets produced by dictionary-based and pre-trained machine-learning-based sentiment analyzer?
2. To what extent do the above-mentioned ratings correlate with each other?

**Methods**

***Twitter scraping***

To collect the desired corpus, I created a Python program named *twitter\_scraper.py* by mainly using Twitter Application Programming Interface (API) and Python module Tweepy to handle the data. The program includes two main functions, as Table 1 shows.

**Table 1***Two main functions of the Twitter Scraper*

|  |  |  |
| --- | --- | --- |
| **Function Name** | **Argument(s)** | **Feature** |
| *get\_tweets()* | *user screen name,*  *tweet count* | connect to Twitter, get the Tweets, pre-process the Tweets, and write the results to a CSV file |
| *clean\_tweets()* | *user screen name* | perform cleaning of the Tweets and save results as a CSV file to prepare for sentiment analysis |

**Getting the Tweets.** In the process of getting the Tweets, a Python module *Tweepy* (Roesslein, 2020) is to be initiated with the settings stored as a file named *setting.py*. The settings include four elements (namely consumer key, consumer secret, key, and secret) needed for Twitter Authentication. After successful Twitter Authentication, the function will obtain as many as 3200 Tweets from the account the user specifies. It is worth noting that, as Twitter API returns a maximum of 200 Tweets by using the traditional method, a built-in cursor within *Tweepy* is introduced to perform pagination[[1]](#footnote-1). By using pagination, the program can get a maximum number of 3200 Tweets until it reaches the limit by Twitter.

After getting the Tweets from the Twitter API, the Tweets are to be cleaned for the first time. A nested dictionary is created to store the following information: 1) time when Tweet was mined, 2) Tweet creation time, 3) Tweet unique ID, 4) name of the Twitter account, 5) screen name of the Twitter account, 6) language, 7) favorite count, 8) retweet count, and most importantly, 9) the text of the Tweet. The nested dictionary is converted to a Pandas Data Frame (The Pandas Development Team, 2020), with an added extra column of the text length count. The Data Frame is written out as a Comma Separated Values (CSV) file whose name ends with *\_tweets.csv*.

**Cleaning the Tweets.** As for the cleaning of the Tweets, the following methods are applied to the Pandas Data Frame. First, all punctuations and special symbols including @ and # will be removed, with regular expression matching. Second, stop words will be removed as well. Stop words are semantically and grammatically insignificant words (Liu, 2020). In this program, a list of stop words[[2]](#footnote-2) is loaded into the program as a pre-defined set. Then, if the texts include a word that matches the content in the pre-defined set, that word will be deleted. Third, the tweets will be set to lowercase to prevent inconsistency. Finally, the results will be saved into another CSV file whose name ends with *tweets\_processed.csv*.

***Sentiment analysis***

Bravo-Marquez, Mendoza, and Poblete (2013) mention that there are two main types of sentiment classification, which are subjectivity and polarity. In this project, I will focus on polarity, which can be roughly classified as *positive* (polarity score = 1), *negative* (polarity score = -1), and *neutral* (polarity score = 0) (Liu, 2020).

To analyze the sentiment of the Tweets, I created a program named *sentiment\_analyzer.py*. As Table 2 suggests, the program contains a pre-trained machine learning analyzer, a dictionary-based analyzer, and a sentiment model correlation tool.

**Table 2***Three main functions of the Sentiment Analyzer*

|  |  |  |
| --- | --- | --- |
| **Function Name** | **Argument(s)** | **Feature** |
| *analyze\_sentiment\_pretrained()* | *user screen name* | analyze the sentiment using the pre-trained machine learning model from a Python model called *TextBlob* |
| *analyze\_sentiment\_dict\_based()* | *user screen name* | analyze the sentiment using the dictionary-based analyzer |
| *sentiment\_model\_correlation()* | *user screen name* | write out two ratings datasets to a new CSV file; create a new csv file to store the correlation data |

**Building sentiment analyzers.** As for sentiment analysis using the pre-trained machine learning model, *TextBlob* (Loria, 2018) is applied to the Tweets in the Pandas Data Frame, and the sentiment ratings of each Tweet are saved to a column named *rating\_pt* (the acronym *pt* is a short form of pre-trained)*.* As for the dictionary-based model, first, a list of opinion lexicons (positive words and negative words, totaling 6800 words)[[3]](#footnote-3) will be loaded into the program as two respectively pre-defined sets. Second, each Tweet will be matched against the pre-defined sets, and then the positive words and negative words are found.

In terms of the most important step in building the dictionary-based sentiment analyzer, the sentiment rating of the text is calculated according to the following formula:

*Sentiment rating = Positive score + Negative score*,

where one positive word equals 1 point, and one negative word equals -1 point.

Theoretically, the sentiment rating can be any number, and it calls for the need to normalization of the rating, thus making the range of each rating to be in the interval of [-1, 1] (both inclusive). In this program, the *Maximum Absolute Value* method is used, and the formula can be written as shown below.

Since this method is built in one of the Python modules named *scikit-learn* (Pedregosa et al., 2011), the normalized ratings can be quickly calculated. The normalized ratings of each Tweet are saved into a column named *rating\_db* (the acronym for *dictionary-based*).

Finally, two CSV files containing the ratings are saved. The CSV files have filenames ending with *\_tweets\_pretrained\_sentiment.csv* and *\_tweets\_dict\_based\_sentiment.csv*. It is worth noting that in the meantime, two frequency dictionaries of all positive words and negative words are generated.

**Investigating the correlation between the ratings produced by two models.** A simple correlation between the two ratings is conducted to better describe the relationship between the two models. Python has a module named *pingouin* (Vallat, 2018), and it contains several useful statistic models, I incorporate the module into the program so that the program can calculate Pearson’s R quickly. The correlation results are saved into a CSV file ending with *\_tweets\_sentiment\_model\_ratings.csv.*

**Data and analysis**

***Summary of Tweets***

As for Tweets gathered from the Twitter scraping program, a total of 3200 Tweets on April 23, 2021. The program collected the Tweets dated from March 26, 2021, to April 23, 2021. Table 3 shows a summary of the Tweets.

**Table 3***Summary of the Tweets collected by the Twitter Scraper (sorted by month)*

|  |  |  |  |
| --- | --- | --- | --- |
|  | **March** | **April** | **Total** |
| **Count of Tweets** | 611 | 2,589 | 3,200 |
| **Favorite Count** | 179,509 | 846,867 | 1,026,376 |
| **Average Favorite Count** | 294 | 327 | 321 |
| **Retweet Count** | 46,553 | 247,417 | 293,970 |
| **Average Retweet Count** | 76 | 96 | 92 |
| **Text Length** | 82,349 | 342,900 | 425,249 |
| **Average Text Length** | 135 | 132 | 133 |

As Table 3 suggests, 611 Tweets posted in March and 2589 Tweets posted in April were collected. The total average length of the Tweets is 133 characters. For each Tweet, the average favorite count is 321 times, while the retweet count being 293,970 times.

***Analysis of the ratings***

In terms of the ratings produced by the two sentiment analyzers using the dictionary-based model and the pre-trained machine-learning-based model, I used SPSS (IBM Corp, 2020) to analyze the data and produced a summary of descriptive statistics for the ratings, as shown in Table 4, Figure 1, and Figure 2. This sub-section also helps in answering Research Question 1.

**Table 4***Descriptive statistics of the ratings produced by sentiment analyzers (N = 3200)*

|  |  |  |
| --- | --- | --- |
|  | **Sentiment Analyzer Model** | |
| **Statistics** | **Dictionary-based Model**  **(*rating\_db*)** | **Pre-trained Machine Learning Model**  **(*rating\_pt*)** |
| **N** | 3200 | 3200 |
| **Mean** | -.059 | .030 |
| **Median** | .000 | .000 |
| **Mode** | .000 | .000 |
| **Min** | -1.000 | -1.000 |
| **Max** | .800 | 1.000 |
| **Midpoint** | -.100 | .000 |
| **Range** | 1.800 | 2.000 |
| **Std. Deviation** | .213 | .220 |
| **Variance** | .045 | .048 |
| **Skewness** | -.242 | .139 |
| **Kurtosis** | .933 | 4.551 |

**Figure 1**

*Frequency distribution of sentiment rating by dictionary-based model*

![Chart, histogram

Description automatically generated]()

**Figure 2**

*Frequency distribution of sentiment rating by pre-trained machine learning model*

![Chart, line chart

Description automatically generated]()

To answer Research Question 1, from Table 4, Figure 1, it can be found that for dictionary-based ratings, *Mean* = -.059, *Mode* = .000, *Median* = .000, and *Midpoint* = -1.000, and thus *Midpoint < Mean < Median* = *Mode.* Table 4 and Figure 2 shows that for pre-trained machine learning model, *Mean* = .030, *Mode* = .000, *Median* = .000, and *Midpoint* = .000, and thus *Mode = Median = Midpoint* < *Mean.* As for standard deviation and variance, noticeably, the ratings produced by two models suggest similar variance, with *SD* = .213 and *Variance* = .045 for ratings produced by dictionary-based model, and *SD* = .220 and *Variance* = .048 for the pre-trained machine learning model.

**Table 5***Frequency table of neutral sentiment rating (rating = 0) (N = 3200)*

|  |  |  |
| --- | --- | --- |
|  | **Dictionary-based Model**  **(*rating\_db*)** | **Pre-trained Machine Learning Model**  **(*rating\_pt*)** |
| **Count of neutral sentiment** | 1,432 | 1,542 |
| **Portion of neutral sentiment** | 44.8% | 48.2% |
| **Total tweets** | 3200 | 3200 |

It is worth noting that the distribution of ratings of the dictionary-based model is close to a normal distribution, as both skewness and kurtosis (*Skewness* = -.242 and *Kurtosis* = -.139) are close to “0.” However, as for the distribution of ratings of the dictionary-based model, it is obvious that *Kurtosis* = 4.551, while *Skewness* = .139. It is safe to conclude that the ratings have a heavy portion of data (rating = 0, neutral) that outnumbers the rest. To further investigate the heavy portion, the frequency table (Table 5) is analyzed. From Table 5, both sentiment analyzers rated a huge portion of Tweets as neutral with similar output. Suppose the pre-trained machine learning sentiment analyzer is accurate, then it is safe to rudimentarily summarize that the dictionary-based sentiment analyzer has reasonable accuracy in distinguishing neutral sentiment. However, to further investigate the relationship between the two models more in-depth investigation is required.

***Correlation***

To further investigate the relationship between the ratings produced by the two models and seek answers for Research Question 2, I explored the correlation between the ratings produced by dictionary-based and pre-trained machine-learning-based sentiment analyzers. The results are shown in Table 6.

**Table 6***Correlation matrix of dictionary-based sentiment rating and pre-trained machine-learning-based sentiment rating (N = 3200)*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | ***r*** | ***r*2** | **adjusted *r*2** | **95% CI** | ***p*-value \*** |
| **Pearson’s Correlations** | .357 | .128 | .127 | [.33, .39] | 7.462 × 10-97 |

*Note*: CI stands for confidence interval; \* *p*-value shows statistical significance, *p* < .001

Table 6 shows the correlation between the ratings produced by two models, where *r* = .357, *p* < .001, with a 95% CI of [.33, .39], suggesting that there is a statistically significant positive correlation between the ratings produced by dictionary-based and machine-learning-based sentiment analyzers. It is worth pointing out that the 95% CI is tight, and the *p-*value is tiny (*p* = 7.462 × 10-97) (Everitt & Skrondal, 2010 and O'Brien & Yi, 2016). Hence, I am very confident to draw this conclusion.

**Challenges and limitations**

Through the process of exploration and investigation, I encountered several challenges including figuring out efficient and effective data structures to store data and conducting data analysis in Python. Fortunately, I tackled most of the challenges along the way. However, one of the most challenging tasks was scraping the Twitter data and building a corpus. Originally, I was only able to scrape a maximum of 200 Tweets using *Tweepy*. After multiple attempts, I introduced a *cursor* built within *Tweepy* and was able to collect more tweets. As Twitter API limited the maximum Tweets a user/developer could obtain at one time, I endeavored to obtain approximately 3200 Tweets in maximum. I have explored other methods as well, but it involved bypassing the Twitter API restrictions.

As for the limitations of this project, as I mentioned the dictionary-based sentiment analyzer was a prototype, it remains a question of how to improve it - In the prototype, I only took option lexicons (positive and negative words) into consideration when it came to the calculation of rating. In the current version of the program, the rating formula is simply interpreted as the count of positive words minus the count of negative words, or “positive score + negative score”, where a positive word equals 1 point, and a negative word equals -1 point.

However, other factors such as negation words, punctuations, emoticons, and even emojis can impact the sentiment ratings, as well. By incorporating the factors mentioned above, I might be able to improve the sentiment rating formula.

As for the choice of text, in this project, I explored texts from different registers and finally decided on analyzing the Tweets from ABC News (@abc). However, there are other news accounts on Twitter as well, such as Fox News, CNN, CBC News, CNBC News, and so on. I would like to investigate other news accounts on Twitter in the future. Also, it requires more research to answer the question of whether the ratings would vary for texts of other registers, such as personal Tweets by famous people.

**Conclusion and reflection**

In this project, I created a Twitter scraper and a prototype dictionary-based sentiment analyzer, and I applied the model pre-trained machine-learning-based sentiment analyzer. The ratings produced by the dictionary-based sentiment analyzer positively correlate with that by the pre-trained machine-learning-based analyzer. However, I would not declare that the dictionary-based sentiment analyzer has been flawless so far, as suggested by the correlation coefficient *r* = .357.

As discussed above, although both ratings by the two models showed a similar portion of neutral sentiment, it is imperative to improve the rating formula of the dictionary-based model to reflect higher accuracy, especially for negative sentiment and positive sentiment. Noticeably, I only focused on analyzing the Tweets by ABC News using both sentiment analyzers. In future research, different Twitter accounts and Tweets of different registers need to be taken into consideration.

All in all, this class helped me gain (or regain) my interest in programming and guided me to explore more in the field of corpus linguistics and text processing. I would say I have applied everything I learned in class, including dictionaries, lists, and so on in the classroom. Meanwhile, to tackle the issues when I was working on the project, I learned to make use of Pandas Data Frame, Numpy (Harris et al., 2020), as well as other useful Python modules. I am confident that in the future, I will be able to improve the prototype sentiment analyzer by comprehensively incorporating the knowledge I gained in the class and outside of the classroom.

1. https://docs.tweepy.org/en/v3.10.0/cursor\_tutorial.html [↑](#footnote-ref-1)
2. https://github.com/stopwords-iso/stopwords-en [↑](#footnote-ref-2)
3. http://www.cs.uic.edu/~liub/FBS/opinion-lexicon-English.rar [↑](#footnote-ref-3)