Data Processing Algorithms: Tweet Processing

Fatema Bhayani
6-26-2020

Table of Contents

1 Introduction

The algorithms developed in this project can be categorized into the following categories:

1. Twitter Download Algorithms: These algorithms download social network data from Twitter using the Twitter API.
2. Tweet Processing Algorithms: These algorithms process the tweets by removing stop words, hashtags, tweets in foreign languages etc. Moreover, these algorithms also analyze the word frequency of the user's tweets and determine what communities they belong to.
3. Clustering Algorithms: These algorithms cluster users based on their similarity and their connections in the social network.

This report will focus on Tweet Processing Algorithms. After the tweets are downloaded, they need to be processed and cleaned before they can be analysed.

2 Word Frequency

There are three aspects to word frequency:

2.1 Daily Tweet Download

Daily Tweet Download is a background service that downloads a new set of random tweets on a daily basis. It is customizable so that users can specify, for example, how they would like to receive monitoring information (such as how many Tweets were downloaded), whether it be via email or through other supported mediums.

2.2 Daily Tweet Processing

Daily Tweet Processing is a background service that "processes" the daily downloaded Tweets. The Daily Tweet Processing is organized into the following steps

- remove non-English Tweets
- remove stop words
- stemming

2.3 Compute Word Frequency

Word Frequency takes the processed tweets as the input and for each word that occurred in the tweet, a Word Frequency Vector keeps an entry of how many times this word has been used. Call this the word count. Then it compute the total word count, i.e. the sum of the word count over all words that the user has used.

It outputs a dictionary with all words, with frequency f(w) of a given word is given by

$$f(w) = \frac{word\_count(w)}{total\_word\_count}$$

The functions that perform word frequency are given below:

Function Name	Input	Description
generate_global_word_frequency_vector	Date, input_collection_name, output_collection_name	The function draws the tweets from the Mongo database collection "input_collection_name" and maps different words to the number of times they are used. It is calculated based on random tweets that we download from Tweepy.
generate_user_word_frequency_vector	Date, User, output_collection_name	The function draws the tweets of a specific user from the Mongo database collection. It maps different words used by a specific user to the number of times they are used. This is calculated based off all the user's tweets.
generate_relative_user_word_frequency_vector	start_date end_date user output_collection_name	The function compares the word frequency in the set of random tweets and the set of user tweets. It maps different words used by a specific user to the number of times they are used in the global collection. This is calculated based off all the user's tweets and all the global tweets.

So before using these two functions, we need to make sure that the prerequisite data is downloaded first and stored in the database. This is done using the TwitterDownloader.

3 Tutorial and Examples

3.1Word Frequency: Daily Tweet Download

In order to automate the daily download of raw tweets, I have created a script called raw_tweet_scheduler_daemon.

To use this script, simply call it like so,

```
python3 raw_tweet_scheduler_daemon.py --num-tweets-to-download <num_tweets> --
time-to-download <time> --options <option-value>
```

Notice that we provide two arguments,

1) number of tweets to download
2) time each day we want to download our tweets

“options” currently serves as a placeholder for additional configuration options that this program will provide that are not yet implemented (such as how a user would like to receive monitoring information).

Furthermore, if num-tweets-to-download is not specified, the program will assume that the user wants to download at the full rate limit. If time-to-download is not specified, the program will download at a default set time.

There are three aspects of this script.

1) Schedule daily.
2) Download raw tweets.
3) Run as a daemon in the background.

To address the issue of scheduling, a Python library called schedule is used. This allows us to run the raw Tweet download function at a specified time daily.

To address the issue of running in the background, I am using another Python library to run the script as a daemon. This library handles many issues associated with running a daemon, such as not printing to stdout.

Addressing the issue of downloading raw tweets, I am using an updated version of my random Tweet download method,

```
get_random_tweets(self, num_tweets: int, output_collection_name: str)
```

The way this works is that I have the following function,

```
def download_daily_tweets():
```

Our scheduler calls this function daily. In download_daily_tweets, I generate a collection name, which is based on the date. I then pass in the given collection date to get_random_tweets.

In turn, get_random_tweets creates a new collection with the specified name and where documents(elements) in this collection are Tweet objects (contain information like text, tweeter, location).

3.2Word Frequency: Daily Tweet Processing

To process the tweets that are downloaded, we can call the following function.

```
def process_raw_tweets(self, tweet_ID, text, output_collection_name):
```

This function processes the text of the tweet by removing any links to webpages, numbers, emojis, hashtags, usernames, foreign words, and stop words. Then it saves the result into a Mongo database in the output_collection_name Collection.

3.3Word Frequency: Compute Word Frequency Vector

To compute the word frequency of a user's tweets we can call the following function.

```
def generate_user_word_frequency_vector(self, date: datetime, user,
output_collection_name, date_input_config=None, output_config=None)
```

The user word frequency vector maps different words used by a specific user to the number of times they are used. This is calculated based off all the user's tweets.

To compute the word frequency globally, we can call the following function:

```
def generate_global_word_frequency_vector(self, date: datetime,
output_collection_name, date_input_config=None, output_config=None)
```

The global word frequency vector maps different words to the number of times they are used. It is calculated based on random tweets that we download from Tweepy.

```
def generate_relative_user_word_frequency_vector(self, start_date: datetime,
```

```
end_date: datetime, user, user_word_freq_config={}, global_word_freq_config=None,
output_collection_name="", output_config=None):
```

The generate_relative_user_word_frequency_vector compares the word frequency in the set of random tweets and the set of user tweets. It maps different words used by a specific user to the number of times they are used in the global collection. This is calculated based off all the user's tweets and all the global tweets.