# Twitter Friendship Analysis Project Report

Rachel Routly

COMP 1405Z

November 8th, 2020

## Introduction:

In the following report I will summarize the findings of and discuss the implementation of my final project of Twitter Friendship Analysis in python. Though the project did not work as well as I would have hoped, this implementation is the best possible way I could have done this project considering the timeframe and the somewhat limited course concepts that we discussed as this class is only an introduction/first year course.

## Method One:

Runtime → O(n + k + n + k + s + p log p ) = O(2n + 2k + s + p log p ) = O(n + k + s + p log p)

- n = length of the first users tweets
    - First loop of O(n) is the function call for get_user_tweets on line 231.
    - Second loop of O(n) lines 246-264
        - Within this loop is a second for loop that iterates over the words in a tweet, however this was ignored for the runtime analysis as the worst case runtime is constant at O(280) due to twitter character maximum
- k = length of second users tweets
    - First loop of O(k) is the function call for get_user_tweets on line 233.
    - Second loop of O(k) lines 266-282.
        - Within this loop is a second for loop that iterates over the words in a tweet, however this was ignored for the runtime analysis as the worst case runtime is constant at O(280) due to twitter character maximum
- s = either length of user1_interests or length of user2_interests, whichever is shorter
    - This loop is within an if statement to find the smaller list of interests to increase efficiency as only the interests found in both lists are relevant.
- p = the length of the combined_likes
    - This is the complexity of the merge sort program that I modified to take a list and two dictionaries, explain later in the report

The benefits of this approach is mainly that it is the simpler implementation and that it does not require the corpus to be read into memory which, though constant, takes approx. O(1,000,000) based on the large number of words. Seeing this implementation also highlights the benefits of the Corpus approach.

As explained below, the drawbacks of this method over the Brown Corpus one is that the words are never separated into categories which both increases the words passed to the get_sentiment function and, more importantly allows adjectives to remain in the list of interests for a user.

## Method Two - Brown Corpus:

Runtime → O(n + k + n + k + s + p log p ) = O(2n + 2k + s + p log p ) = O(n + k + s + p log p)

- n = length of the first users tweets
    - First loop of O(n) is the function call for get_user_tweets on line 288.

    - The second loop of O(n) for this function has two internal loops, however this does not change the complexity to O(n^2) because both internal loops have a worst case runtime of O(280) due to tweet character maximums
        - Lines 307-340
- k = length of second users tweets
    - First loop of O(k) is the function call for get_user_tweets on line 290.
    - The second loop of O(k) for this function has two internal loops, however this does not change the complexity to O(n^2) because both internal loops have a worst case runtime of O(280) due to tweet character maximums
        - Lines 343-371
- s = either length of user1_interests or length of user2_interests, whichever is shorter
    - Same as method one as it falls in the common_processing function
- p = the length of the combined_likes
    - Same as method one as it falls in the common_processing function

Though this method has one extra internal for loop for each user, this does not change the runtime complexity significantly as both loops have a constant worst case runtime of O(280) for the maximum 280 characters in a tweet. This extra loop was needed to allow for the adjective words to be taken out and the sentiment gauged separately before incorporating the interests of that particular tweet into the user's overall interests.

Though this method does not improve on the runtime complexity of the first method it is more accurate; this is because it gauges the sentiment based only on the adjectives and does not include adjectives in the interests of a user. Comparing Figure 1, which compares my twitter with my cousins using the first method and Figure 2 which uses the second method; it can be seen that not only does the Brown Corpus approach remove adjectives such as "pretty", "perfect", "proud", "happy", and "love" from our common interests, but at least one of the predicted interests is accurate to our relationship as both of us love the actor Lin Manuel Miranda and the interest "lin" can be found in our predicted top 10 interests.

Figure One.

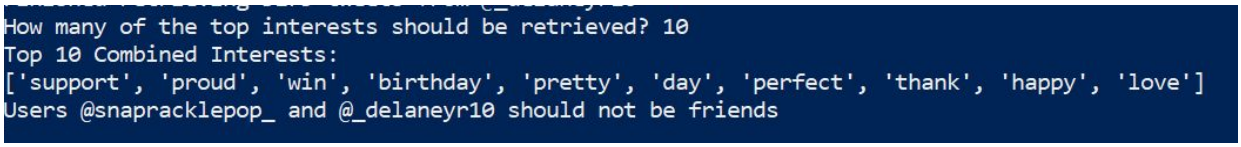
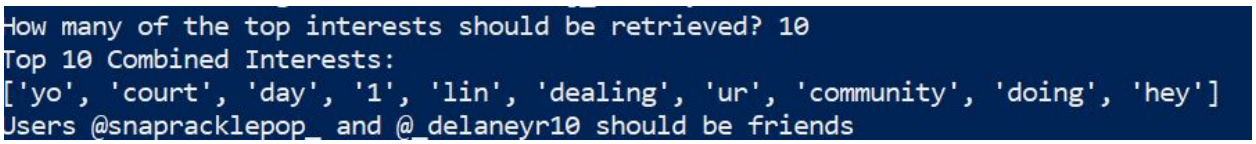



Figure Two.



The main drawback of using the brown corpus to identify the different parts of speech is the age and context of the corpus; it is based on written prose from the 1960s which generally uses vastly different language from tweets in the present day. As discussed later this could be partially mitigated by using a more modern text source, such as a thesaurus.

## Other Functions

To get the highest combined interests I had to sort the list of combined interests, to do this I made a modified version of the merge sort algorithm that took a list and two dictionaries as parameters rather than just one list. Not only does this implement sorting and recursion as discussed in the course, it also modifies and builds on one of the key algorithms we discussed proving and cementing my understanding of merge sort.

The other main function that I had to consider the algorithm for was my relevancy function. When I first implemented my program I was treating all interests as equal, but then considered that tweets that are from a longer time period ago will be less relevant to their current interests than ones that happened recently. I used the function

f(months) = - (months - the number of months since the creation of twitter) ^ 2

This gives a quadratic function where the x-intercept of the line is on the creation month of twitter, this value was chosen as no tweets could happen before twitter was created and so the more months have passed since the current date, the less relevant the interests in that tweet are to a user. I chose to use a quadratic function instead of a linear function as the curve of the quadratic better represents interests than a purely linear function.

## Tradeoffs/Optimization

The biggest tradeoff that I made was sacrificing memory for a better runtime. I did this by holding the relevancy values of the interests in a dictionary so that retrieval would happen in O(1) while also holding the interest names in a list to allow iteration. Additionally, I saved minimized space usage where I could by modifying the merge sort function to take two dictionary values and add them, rather than making another dictionary with the contents of both.

Another piece of optimization that I implemented was checking which user's list of interests was smaller and iterating over the smaller when combining the values because if only worrying about the values that occur in both lists only any value that is not in one of the lists cannot be in both lists.

## Computational Thinking:

There are four main parts of computational thinking, decomposition, pattern recognition, abstraction, and algorithms.

The algorithms that I implemented are explained earlier in this report.

One of the ways that I used abstraction in my project was by simplifying the function I used for relevancy based roughly on the number of months since the tweet was made; instead of wasting processing and memory to account for leap years or months with different day counts, I used a rough estimate as this is not the important detail of the function.

I also used decomposition by breaking my program down into smaller more manageable pieces. This was especially helpful to me as when developing specific functions with narrower uses I was able to test it in another file without wasting time on the retrieval of every tweet in the user's timeline.

Finally, my pattern recognition skills allowed me to save time by not "reinventing the wheel" with some of my functions. I noticed that I would be able to use some concepts that we discussed in class, such as the merge sort function, and slightly modify them for use in my project. This saved me a lot of time while implementing everything and likely made my outcome more efficient than if I had not noticed the similarity between these problems.

## Issues:

The main issue that remains in my code is not something that I can reasonably fix as it is an occasional issue with calling the Twitter API through the Tweepy library. As working with APIs was not a significant part of this course, I elected to simplify the process by using a library to allow focus on the algorithm development that is relevant to course evaluation. However the Tweepy library can have performance issues depending on the proxy settings of the end user. If this issue occurs while running the code either check your proxy settings as they can interfere, or use a VPN. This is especially important if on Carleton wifi/ethernet as I consistently ran into issues with retrieving tweets until I used a VPN.

## Improvements:

The most obvious improvement to this project would be to implement some form of machine learning in two main areas. First, it could be used to better perform sentiment analysis based on the context of the tweet, rather than just the positive/negative word content of the tweet. Second it could be used to better analyze tweets for interests improving on my current solution that just removes "stop words" and leaves all other words as interests. However, implementation of machine learning algorithms is beyond the scope of a first year course.

Another simpler improvement that I could have made would have been using a thesaurus instead of a corpus for the part of speech tagging. Despite it requiring more space as it would be a much larger file, it would have likely been much more accurate. My algorithm would mostly stay the same for this improvement, the storage of information/parsing of the text file would be a little different depending on the file structure of the thesaurus/dictionary being used. The runtime complexity wouldn't change as aside from the initial processing taking longer for a larger file, since both approaches would store in dictionaries the runtime remains O(1). This would not entirely solve the issue of twitter language being different from that of the 1960's prose in the

Brown Corpus, but it would likely at least partially mitigate it by including some more modern language.

## Extensions:

One extension that I considered adding was to allow the user of my program to specify the dates between which they want tweets from. However when looking into the Twitter api documentation for a parameter to implement this feature, I found that searching by dates is only available for premium users. Another way that this could be implemented would be checking the tweets after they are retrieved and discarding the ones that fall after the specified date, but as this is quite inefficient I opted to not include this feature.

## External Resources Used:

This project would not have been possible without four key text files, cited below.


na. Stop Word List 1. nd. http://www.lextek.com/manuals/onix/stopwords1.html

Mckenney, Dave. negative-words.txt. 2020. Retrieved from Lecture Code on cuLearn

Mckenney, Dave. positive-words.txt. 2020. Retrieved from Lecture Code on cuLearn

Nelson, Francis & Kucera, Henry. Computational Analysis of Present-Day American English. 1967. Providence, RI: Brown University Press.


## Conclusion

Though the project does not work extremely well for the intended purpose (predicting friendship of twitter users) the algorithms that I implemented in the project work as intended and the complexity of both space and runtime have been taken into consideration. Due to the informal nature of tweets, the words used are often not able to be checked against dictionaries or existing word lists. When creating my project proposal I did not take into account that without using machine learning it would be very difficult to differentiate what is an “interest” and what is just filler portions of vocabulary that fall outside of traditional grammar or spelling. Additionally, what I was able to create may have alternative uses such as comparing the general vocabulary of two users. In conclusion, though the project does not work as intended, it does cover the concepts that we worked on in the course and improvements to its function would not be possible without stepping far outside the realm of the course content.