Ahmad Afghan

Brooke Porter

Jessica Serrano

CSC 664 Multimedia Systems (Undergraduate)

Final Report

Identify Drug Dealers on Social Media

**Introduction:**

Social media makes it very easy for illicit substances to be bought and sold over the internet. There is too much data in too many different locations, as well as anonymity, for police to realistically track drug use or drug sales over the internet. This means that drugs have become easy to distribute, easier to acquire even for kids, and dangerous because there is no guarantee of drug quality or even if a buyer has been given the right drug.

We will analyze this problem in a number of steps. First, we will describe the components of the problem, and how we address them. Second, we will give a general description of how we chose to solve the problem and why we took the approach that we did. Third, we will describe prior relevant work, and review our inspirations for this project. Fourth, we will give a detailed and technical description of our approach. Lastly, we will give the data showing how well our implementation worked.

**Proposed Problem Formulation:**

A large facet of this problem is that people use different words in different contexts to mean different things, and that slang changes all the time. It is difficult to search for information specific to any given drug, without also getting extra noise. A machine can’t tell the difference between the sentences “I smoked some weed today” and “I pulled the weeds from my yard”, for instance. Both sentences include the word “weed” but in very different contexts. We would have to instruct the machine to avoid sentences which include “yard”, for example, to remove the second sentence. Additionally, the word “dope” can be used to specifically refer to drugs, but is lately used as slang to replace “cool” or “awesome”. That is something that we have to watch out for.

**High-Level Description:**

To help enable law enforcement, we chose to evaluate Twitter data using machine learning to find tweets related to drugs. Using machine learning, it becomes easier to filter out unrelated data to find tweets only related to drugs, and then filter further down to tweets who likely belong to drug dealers.

We are splitting tweets into one of two categories, based on their sentiment: either positive or negative. Basic marketing strategies show that to make a sale, a person must be enthusiastic, passionate about their product, and must try to convince a potential buyer that their product is better than the competition. Looking at this strategy, we decided to focus solely on tweets with positive sentiment to represent potential sales, as a sales person is unlikely to hold a negative sentiment to their words.

To split the tweets, first we have to tokenize them and clean the data to be usable by a sentiment analysis network. Then we have to train the network with specific data for training. Finally, we can run our real data through the network.

**Prior Work:**

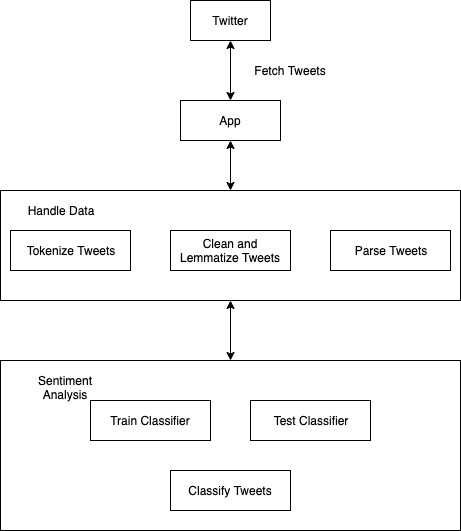
In the process of our research, we found two published papers related specifically to Twitter data collection and machine learning algorithms from which we took inspiration. The first paper, “Assessing Perceived Organizational Leadership Styles Through Twitter Text Mining” [1], in which tweets were collected from various Italian organizations from the 2015 Forbes Global 2000 and analyzed using a 10-factor leadership model to determine an organization’s approach to leadership. La Bella et al. first used basic preprocessing techniques to filter through the gathered information, then did a sentiment analysis on the tweets to determine positive and negative associations. Since our focus is to discern sentiments within drug related tweets, and finding the difference between positive and negative tweets, this paper expressed a method we could use for training our algorithm to determine text patterns of a drug user or seller.

The second paper we found, “Public Reactions to e-Cigarette Regulations on Twitter: a Text Mining Analysis” [2], analyzed opinionated tweets regarding the new FDA regulations on e- cigarettes implemented in 2016. The tweets were preprocessed then analyzed through text mining to parse through and group the information into related topics and further interpret the data. This was more closely related to our general problem, as it focused solely on determining the sentiment found in tweets about e-cigarette regulations, which is within the domain of drug use and exchange. However, the methods used in this paper differed from how we wanted to determine sentiments in our formulation.

The third paper we used, “SentiBench - a benchmark comparison of state-of-the-practice sentiment analysis methods” [3], compares the various methods of sentiment analysis to show the 24 most popular and their advantages and disadvantages. This paper is useful because it lists the options in an unbiased way, allowing us to choose the method that works for us. It focuses solely on the different tools used for sentiment analysis, and does not work to solve a related problem to ours.

Taking these prior works into consideration, we decided to follow the same process of data collection and training an algorithm to determine the sentiments of the tweets. By taking the approach of training the data to find positive and negative sentiments in the tweets, and focusing on positive sentiments and business and marketing language, it would be easier to determine drug sellers from within our collected data.

**Detailed Description:**



**Section: Obtaining Data (Brooke Porter)**

We started by gathering our data. We chose to use Twitter for the same reasons that we read via our resources: “because data are freely accessible via the Twitter API, with much less constraints than other social media platforms such as Facebook“ [1]. We obtained a Twitter developer account to get API keys and tokens. In Python, we use a library called Tweepy to assist in fetching random tweets. Tweepy needs to be given four different API keys from Twitter in order to authenticate fetching tweets, so those keys were set as environment variables for privacy on the repository. The keys also needed to be added under the Github settings for the repository, so that it would have access to them as well.

After authenticating Tweepy, we calculate the time for 15 minutes in the future, and tell it to fetch tweets for that long. First, we write the header to a csv file if that file does not already exist. Then, we instantiate the Stream Listener so that Tweepy knows what to do. In the Stream Listener, we set the end time so that the listener knows when to stop, and use the function on\_status(self, status) to actually handle the tweets themselves. We get the values for user\_id, stat\_id, creation, tweet\_body, and name from the tweet and write those fields into the csv file.

If there is an error, we check to see whether or not the error code is 420, as that is the code signifying that we have reached our rate limit from Twitter on streaming tweets.

Finally, we tell Tweepy to go ahead and fetch tweets, using the on\_status and on\_error functions in the listener. After 15 minutes, it ends.

We implemented a Github workflow, so that Github will automatically run this program every half hour and commit the changes. The workflow file also needs to have the Twitter API environment keys added to it, as well. Every half hour, Github runs the file for its required 15 minutes, then commits and pushes the changed csv file. We ran into difficulties in this step, because Github has a file size limit of 100MB. The csv files reach 100MB at a little over 700,000 tweets on average, and we needed to use much more than that, so the program needed to be updated to use a new csv file every time we hit that limit. We stopped running the tweet fetching once we reached 8 data capped csv files, leaving us with 5,264,920 total tweets.

**Section: React Frontend (Ahmad Afghan)**

We were initially going to display the final results from our analysis to a React frontend. After gathering our data, it was parsed into a MySQL database, and the React app fetched the data from the database. We showed this during our midterm presentation, however we found that it was rather redundant to parse csv files into a database to display them since the csv files were already parsed data, and we decided to focus our efforts more towards handling and analyzing the data instead of displaying it over the web.

**Section: Handling Data (Brooke Porter)**

We started working with our data once we had a few million tweets to parse through.

To start with, we require the use of the NLTK library, so we use nltk.download() first to obtain the required functional libraries. This was initially giving certification errors, so we added ssl.\_create\_unverified\_context and ssl.\_create\_default\_https\_context into the function, resolving the certification problem.

Next, the data needs to be tokenized into arrays of words. This is also done similarly by La Belle in her research [1]. We open the csv file and parse through each line, treating the row as a dictionary, using the key “tweet\_body” to get the actual content of the tweet. NLTK has a class specifically for tokenizing tweets, and includes optional parameters for removing @mention handles and for reducing repeated letters. The word “boo” with 5 o’s would instead be shortened to 3 o’s. However, it is still seen as a separate word from “boo” with 2 o’s, which does cause some noise in our training and testing. Once we have our tokenized tweet bodies, we return them as an array, leaving us with an array of arrays of words.

Afterwards, the tokenized tweet bodies need to be cleaned and lemmatized. For each array of words per tweet, we examine each word and give it a tag using the NLTK pos\_tag function. This tags each word with a set of letters representing the type of word: NNP is the tag for a proper noun, NN for a common noun, VBN for a past-tensed verb, and so on. Any noun is always going to start with NN and any verb is always going to start with VB. We can look specifically for tags which start with those letters and reassign their tag to just n, just v, or else just a. Then, when we run the lemmatize function, the word will be replaced with its base word. “Running” becomes “Run”, “Using” becomes “Use”, and so on. While tagging each word, we also take the time to remove hyperlinks, as they are unnecessary data to our model. Lastly, we verify that the token is not an empty string, that the token is not just punctuation, and that the token does not exist in the list of stop words. Once we verify the legitimacy of the word, we add it to a new array for representing that specific tweet, and add that to the array which we return, leaving us with an array of arrays of cleaned and lemmatized tokens.

Lastly, before we can work with the data, the array needs to be converted to a dictionary.

The cleaning and lemmatizing step takes the most time out of the entire program, so we added in basic caching to that step. Within the cache folder of the project, if a csv file representing the data does not already exist, then we write a header and the cleaned and lemmatized results to a csv file with the same name. Before performing the tokenizing, cleaning, or lemmatizing steps, we verify if the data already exists in the cache and load from that instead. This drastically reduces the time it takes to run the program.

**Section: Sentiment Analysis (Brooke Porter)**

We chose to use sentiment analysis to categorize tweets as having either positive or negative connotation. We used a free and open-source dataset from Kaggle (<https://www.kaggle.com/kazanova/sentiment140>) which includes 1.6 million tweets, pre-annotated and split evenly between positive and negative sentiment. Sentiment140 is one of the more popular sentiment analysis methods listed by Ribeiro [3] for tweets in particular.

To use this data set, we needed to split it apart between positives and negatives in our code, so that we could tokenize, clean, and lemmatize the tweet bodies as covered in the previous section without losing their annotation. The dictionary keys related to each tweet are different in the annotated dataset from the data we gathered, so we used a different set of fields. When splitting the file, we check if the value for the field “target” is less than 2. This implies that the sentiment is negative, whereas greater than 2 implies positive. A value of exactly 2 is meant to be a neutral sentiment, but the dataset does not actually have any of those. Depending on the value of “target”, we add the tweet to either the positive list or negative list using “text” as the tweet body.

Because the training data is structured differently than the actual data, we use a separate tokenizing function specific for it, but the returned arrays can then be passed through the existing cleaning and lemmatizing function, as well as preparation for the model. The cleaned training data is also cached.

We decided to use a Naïve Bayes Classifier for our network, and opted to train using 70% of the data and test on the remaining 30% from our annotated dataset. This came out to training with 1.12 million annotated tweets, and testing with the remaining 480k. In order to perform this, we needed to convert our split training data back into one, and reapply the annotation. For all of the tweets in the positive half, we convert to a dict with “Positive”, and for the negative half we convert to a dict with “Negative”. We then combine the two halves back together, and shuffle them to randomize the order for training and testing. We finally tell the classifier to train and then test the data, and output the accuracy as well as the top 10 most informative features. Our average accuracy over 8 tests came out to be 75.80% with very little variation. The lowest accuracy is 75.67% and the highest is 75.89% out of those 8 tests. This tells us that after training the model with over 1 million tweets, and testing on about half a million tweets, the network was about 76% accurate on judging the sentiment of our pre-annotated dataset, and consistently reached the same general level of accuracy. That is, however, only for judging positive or negative sentiment.

Naïve Bayes is not very good at understanding sarcasm or context. If someone were to tweet “Oh man, I love Mondays!” with a negative emoji, the classifier will recognize the text as being very positive, not recognizing that the negative emoji would negate the positivity of the statement. I am actually very surprised to see how high the accuracy came out to be, given the shortcomings of Naïve Bayes. I am under the impression that the annotated training dataset we used probably included fewer “trick” phrases, and that our accuracy would be reduced using a more challenging dataset.

After finishing training and testing, we were finally able to run the network on our real data. We run the real data through the tokenizing, cleaning, and lemmatizing functions, however this time we also filter out the data we do not need. If the tokenized tweet does not contain one of our chosen drug-related words [2], then we do not include it in the final data to test on. We tell the classifier to classify the cleaned real data, and only consider the “Positive” classified tweets.

**Experimental Evaluations:**

Using only one csv file of our tweet data, we found out the following: with 720,263 total original tweets, 1038 were drug related according to our filters. That means 0.144% of tweets are drug related, by our filters. Of those drug related tweets, 535 have positive sentiment according to our trained Naïve Bayes Classifier, meaning that 51.5% of drug related tweets have positive sentiment.

Using all 8 of our csv files of tweet data, we found out the following: with 5,264,920 total original tweets, 7729 were drug related, making 0.147% of tweets. Of those drug related tweets, 4009 were considered positive sentiment, making 51.9%.

I only tested the “one csv file” test once, getting the above results. I tested the “8 csv files” 7 times, getting very similar results with each test.

This data is very interesting. With 700k tweets and upwards of 5m tweets, the percentage of tweets that are drug related is very close, as well as the percentage of those tweets that are also positive sentiment. The number of tweets does not seem to change the ratios at all.

Unfortunately, there is a good amount of error in this analysis. Looking at some of the results, it is very clear that we have false-positives, and I am sure that there are also false-negatives. Many of the resulting tweets are not actually related, but happen to include our chosen drug words or slang. We did not find drug sales with the ease that we expected. However, I did find some results that could be drug sales in looking over some of our results, so there is a measure of success to this approach.

We are seeing so much noise in our final data because our list of chosen drug words is not specific enough, and too short. We would need to add to our stop list as well, given slang terms that do not normally accompany drug related words. We could likely improve our accuracy by using a different classifier, such as Long short-term memory (LSTM). An LSTM classifier would better understand sarcasm and context, by weighing words used later in a sentence higher than words used near the beginning, helping to eliminate false-positives and false-negatives.

**Conclusion:**

In an effort to make identifying potential drug sellers easier within the massive amounts of data that comes from Twitter, we chose to parse and filter tweets related to drugs, and run them through a sentiment analysis network to gauge for potential sales. Our results show that about 0.145% of all tweets are related to drug use, and about 51.5% of those have positive sentiment. We can assume that someone looking to make a sale would have a positive sentiment, so we can theorize that a percentage of that 51.5% are related to sales. This is a very small amount of tweets, relative to the large amount of them on Twitter.

Our original goals were to specifically look for drug sales, but we ended up with more noise than is really acceptable. It became more of general drug related tweets. We get too many tweets that include some of the drug related slang, but not actually reference drugs. We would need to work further with our list of drug related words, and our stop list, to improve the filtering process.

Another thing we could improve is our accuracy. We get around 76% on our training and test dataset, but 76% of 1m tweets means about 240k tweets are incorrectly categorized. We could improve this accuracy by changing from a Naïve Bayes Classifier to Long short-term memory.

**References:**

[1] La Bella, Agostino, et al. “Assessing Perceived Organizational Leadership Styles through Twitter Text Mining.” Journal of the Association for Information Science and Technology, vol. 69, no. 1, 2018, pp. 21–31. <http://jpllnet.sfsu.edu/login?url=http://search.ebscohost.com/login.aspx?direct=true&AuthType=ip,cookie,url,uid&db=bth&AN=126974700&site=ehost-live>

[2] Lazard, Allison J, et al. “Public Reactions to e-Cigarette Regulations on Twitter: a Text Mining Analysis.” Tobacco Control, vol. 26, no. e2, 2017, pp. e112–e116. <https://search-proquest-com.jpllnet.sfsu.edu/docview/2116442032?accountid=13802&pq-origsite=primo>

[3] Ribeiro, Filipe N, et al. “SentiBench - a Benchmark Comparison of State-of-the-Practice Sentiment Analysis Methods.” EPJ Data Science, vol. 5, no. 1, 2016. <https://link.springer.com/content/pdf/10.1140/epjds/s13688-016-0085-1.pdf>