Project Report

Ninad Tungare U49815021
Harshad Reddy Nalla U60360285

1 INTRODUCTION

As the part of Course Project for CS506-Tools of Data science, we worked on the Project Involved-Data Analysis. The project was completed under the guidance of Professor Andrei Lapets, BU Spark and Caleb McDermott. The project is to analyze the social media data based political issue and political representatives in different constituents. Also to provide more insights on the Boston 311 data.

1.1 TWITTER DATA

Involved wanted us to conduct analysis on Twitter data since Twitter is an important resource for issues and gauge the current standing, of the political figure we are trying to lookup, in the eyes of the people who vote for him/her.

When looking up the tweets for a handle we discovered that there are four major types of tweets:

1. Tweets sent by the handle: These are the tweets that a person/account has sent out and have importance relevance to the project. Also known as the original tweet.
2. Tweets/comments on the tweets sent by the handle: These are the tweets are some other person/account on a tweet which was sent by the account we looking up. Also known as the tweets in a thread starting with the above mentioned original tweet.
3. Tweets which mention the account: These tweets are sent by some other person/account but has the account we looking up or the name of the person we looking up mentioned in the name. Also known as the original tweet by some other person/account.
4. Tweet which is comment and mentions the account: These tweets are comments sent by some other person/account but has the account we looking up or the name of the person we looking up mentioned in the name. Also known as the tweets in a thread starting with some random original tweet. This tweet is the most important kind of tweet since it is some account trying to bring the entire thread of tweets to the attention of the account we querying and hence we need to mine the entire thread.

1.1.1 Data Mining

We mined the above mentioned tweets as follows:

1. Tweets sent by the hand and Tweets/comments on the tweets sent by the handle: When mining the tweets sent by an account we noticed that we were getting the original tweet and all its sub-threaded tweets too and thus did not need to write a different algorithm to retrieve these two different types of tweets.
2. Tweets which mention the account: Twitter has APIs in place to retrieve tweets based off of a query, to retrieve these kinds of tweets we queried the account and the name of the account.

3. Tweet which is comment and mentions the account: When we retrieve a tweet which mentions the account it can have a tag called 'in_reply_to_id', which is the id of the upper level tweet and we can iteratively follow this sequence, till we reach the highest level, to retrieve the entire thread of tweet.

1.2 BOSTON 311 DATA

Boston 311 Data is real-time data that is put up via the various UIs that they have implemented. As such Involved wanted us to conduct analysis on this data since people of Boston can record the issues they see around them, such as improper parking in their neighborhood, or incomplete garbage collection, etc. These issues also play a major role in how political figures of Boston is perceived by his constituents. As such we collected Boston 311 to conduct analysis and created statistics of the data and visualized the data to make it have relevance and readability.

In terms of the 311 data we could do a lot more analysis since all the issues are tagged with the location and as such we can even visualize the data on a map.

2 PROJECT OBJECTIVES

1. With reference to a representative, what topics are being discussed most frequently on Twitter?

Involved wanted us to use the twitter tweets to analyze and retrieve the important topics for a representative. This was to relate the current issues and topics which were being discussed in the community, with reference to the representative.

The way we accomplished this is by extracting the tweets mentioned above and analyze the data using TF-IDF and generate word-weightages. These word weightages can be visualized or used for further analysis.

2. Using Boston 311 data, which departments are getting the most number of issues in a given time period?

Involved wanted us to use the Boston 311 Data and do exploratory data analysis with the data, one the analysis we conducted was to visualize the amount of the request made by the community over the departments, such that they could see the change in requests quantity over a period of time based on a department and also how were the requests being handled by the department.

3. Using Boston 311 data, in a selected department what are the most prominent issues?

Another analysis we conducted was to visualize the trend of requests. As each department address number of different issues, Involved wanted to visualize the trends in the requests which are addressed by a specific department. In other words, they want to visualize, for each department, the cases they handled and the trend in terms of number of issues.

We developed an application to visualize the Boston 311 Data in a map by creating date and department filters. We also created multiple visualizations in terms of trying to show data trends of the number of issues versus the departments and/or cases and number of issues versus a date range to show when were the spikes in the number of issues and maybe there are real world causes for this.

3 ALGORITHMS AND ANALYSIS

3.1 PHASE 1

3.1.1 Data Collection

We had to collect tweets from twitter for the user handles of some public representatives for Involved. Collecting Tweets from twitter requires knowledge of Twitter API and requesting and retrieving data from API calls. Twitter API have different endpoints for different kind of data. As our requirement was to gather all the tweets we can for the public representative, tweets such as which are sent by them, sent in reference to them or mentions them. We developed a script for collection all the tweet via twitter API with help of Tweepy, which is a python library for extracting twitter data. [1]

The data collection scripts provide Tweepy with Authentication credential for getting tweets from twitter. Twitter requires developer to create an application in Twitter API and provide the authentication key and credentials for querying the tweets from twitter, so that they can enforce different levels of security and only allow certain accesses, and also twitter provides tweets based on the type of developer account, Standard Account can retrieve only tweets going back 7-10 days while there are other paid accounts which can be used to get more historical tweet data.

We developed our script with following functionality:

- Gathered the data using Tweepy wrapper search, user timeline API calls, as we required all the tweets in a thread we had to transform the script to reiterate over the gather tweets from user's timeline and also the tweets which mentioned them and fetched comment id from the tag 'in_reply_to_id' and retrieved all the comment which doesn't directly mentions the user but are referenced to them.
- Used geocode to narrow our search to the Massachusetts area, as specified by Involved, for gathering the tweet for the user handle. We passed geocode to the Tweepy API calls which return tweet which are bounded by a specified radius surrounding the specified geocode, longitude and latitude.
- Twitter API returns tweets which URL link instead of full text, we had to configure the API Call to provide us full-extended text for the tweets.
- As the tweets collection will increase with time we preferred to use MongoDB NoSQL databases to avoid performance issues caused by the big amount of data, plus NoSQL and Twitter use JSON for communication and as such they worked better together.

Biggest limitation we face when using the Twitter API is that with standard twitter account credentials we could only fetch 7-10 days old tweets.

3.1.2 Tokenization and Lemmatization

We used NLTK [2] for text processing on the tweets we gathered. NLTK provide various different techniques to process the textual data. It contains text processing libraries for tokenization, parsing, classification, stemming, tagging and semantic reasoning. We used word tokenizer to split and break down the documents of text data into tokens [3]. We pass documents of text data to the NLTK word tokenizer and we get the non-duplicate different words present in the document as tokens.

For example, a tokenizer divides a string into substrings by splitting on the specified string:

```
sentence = """At eight o'clock on Thursday morning Arthur didn't feel
very good."""

Tokens = ['At', 'eight', "o'clock", 'on', 'Thursday', 'morning',
'Arthur', 'did', "n't", 'feel', 'very', 'good', '.']
```

Once we tokenized the text, we implemented lemmatization and stemming to reduce the common word which mean the same but are spelled differently.

- Stemming is the process of reducing inflected (or sometimes derived) words to their word stem, base or root form generally a written word form. Example:

```
A stemmer for English operating on the stem "cat" should identify such
strings as cats, catlike, and catty.
Words fishing, fished, and fisher -> stem "fish".
```

- Lemmatization depends on correctly identifying the intended part of speech and meaning of a word in a sentence, as well as within the larger context surrounding that sentence, such as neighboring sentences or even an entire document. Example:

```
In English, the verb 'to walk' may appear as 'walk', 'walked', 'walks',
'walking'.
The base form, 'walk', that one might look up in a dictionary, is called
the lemma for the word.
```

We had to firstly preprocess the data by removing all the hashtags, @ and emoticons from the tweets before processing it with NLTK, we also tried to break down the most hashtags in to different words. During the processing of words into token using NLTK word tokenization, we tried several different experiments of Stemming and Lemmatization noticed that Lemmatization gave us better word reductions and as such we used Lemmatization for word reductions. We also removed all the stop words and punctuation from the document text as words like “is”, “a”, “the”, etc. are more frequent but are not relevant while trying to derive the meaning full words from the textual data.

3.1.3 TFIDF Word Weight Vector

We apply TFIDF [4] on corpus of words processed by NLTK word tokens. We use Gensim to build our TF-IDF model for the corpus. We used Gensim model for TF-IDF weights model because Gensim provides the functionality to add documents to its corpus without burning memory and thus we won’t have to keep whole text in memory before providing it to genism model. It makes it run time efficient.

We provide the word tokens to Gensim’s data structures to build the model. Gensim substitutes terms for integer IDs using a dictionary structure, so we build the dictionary which contains unique instances of terms found within the collection. It is a complete lexicon of the vocabulary used in the documents. Building the dictionary is as simple as importing the data structure and passing our stemmed tokens to it. Then we converted the text corpus into a vector format. Then we passed the corpus to TFIDF for

analysis. TF-IDF determines what words are most important for distinguishing the content of one document from another. Once the analysis is completed we got the list of most important words with their TFIDF weights from the tweets data.

TF-IDF is defined as Term-Frequency Inverse-Document-Frequency:

$tf(t,d) = 0.5 + 0.5 \cdot \frac{f_{t,d}}{\max\{f_{t`,d}: t` \in d\}}$, Term Frequency Formula

$idf(t,D) = log\frac{N}{1+|\{d \in D: t \in d\}|}$, Inverse Document Frequency Formula. N = total number of documents in the corpus.

$tfidf(t.d.D) = tf(t,d) \cdot idf(t,D)$, TF-IDF Formula

Example:

```
support,22.568015684390737
meeting,22.551946421825885
morning,22.32533303514948
community,22.268832467335052
council,21.727723238298807
student,21.542411216290102
help,21.537602755410827
```

Using these TF-IDF weights, we created a word cloud to give the word weightage more visual meaning. We used a Python Word Cloud package, this package takes in the word and an 'intensity' of the word, in our case the intensity is the TF-IDF weightage of the word, and the package renders an images where is assigns different font sizes based of the intensity.

Limitation of TF-IDF is that it identifies the terms on which the document is most authoritative although there is no theoretical basis for this. Proving it has been something of a bane for the computer science community.

3.2 PHASE 2

3.2.1 Visualizing Boston 311 Data

We wanted to visualize the Boston 311 data on geographic map, we used Folium package [5] and Pandas to achieve the visualization of Boston 311 data. The Boston 311 data contains Boston public request over various departments, each department handle set of 'Subjects' and each 'Subject' contains various 'Case Titles'. We made sure to use Pandas GroupBy to segregate the Cases to better Visualize the Boston 311 Data. We used the Pandas interface to read the CSV for the Boston 311 data and used Pandas to do the initial operations to retrieve the necessary data for the visualizations.

- To visually display the Issues on the Map, we did groupby on the data on 'Subjects' key and use the geolocation coordinates from results to display the Folium map.
- Folium map is scoped to Massachusetts to better visualize the map.

- We create text message containing Case Titles, Open Date, Closed Date and Address of Complaint and pin this text message on the map marker.
- As there was multiple request made we decided to cluster the data, to reduce screen clutter and make it more readable. We used Folium's MarkerCluster method to display the multiple request from similar locations.

We implemented filters to reduce the scope of data that the user wants to see, since we also observed that we could not visualize all the data since there is simply too much data. User can change the following filters:

- Date Filter user can change date range to filter out the specific period of data.
- Subject Filter user can filter through Subject titles.
- Case Filter user can filter through Case titles.

Limitation of using folium on large dataset is that while visualizing large data, the rendering time of the map would be significantly large.

3.2.2 Visualize the Request Trend

We plotted graph for the number of request made based upon subject. We used Pandas DataFrame GroupBy to group the data based on the subject and then used the result to plot the graph between the number of request vs subject title.

We implemented filters to reduce the scope of data that the user wants to see, since we also observed that we could not visualize all the data since there is simply too much data. User can change the following filters:

- Subject Filter user can filter through Subject titles.
- Case Filter user can filter through Case titles.

3.2.3 Visualize the Trend in Service Issue over a time Range:

We plotted graph for the number of request made based upon subject over a time period. We used Pandas DataFrame GroupBy to group the data based on the subject and then used the result to plot the graph between the number of request per subject title over a time range, since every issue has an Open Date.

We implemented filters to reduce the scope of data that the user wants to see, since we also observed that we could not visualize all the data since there is simply too much data. User can change the following filters:

- Date Filter user can change date range to filter out the specific period of data.
- Subject Filter user can filter through Subject titles.
- Case Filter user can filter through Case titles.

4 PROJECT SUMMARY

4.1 TWITTER DATA

As mentioned above we create word weightages and word clouds for the twitter data, we believe that it is a very important visualization, since at just one glance we are able to tell what most people are talking about when it comes to that account.

Figure 1 Examples of Word Clouds of the Twitter Data

By looking at the above word clouds we can tell that the person who belongs to the first word cloud does a lot of work in Revere and tends to work a lot with schools, and we can draw similar conclusions from the other images too.

4.2 BOSTON 311 DATA

We created three major visualizations for the 311 Data:

_id	CASE_ENQUIRY_ID	open_dt	closed_dt	CASE_STATUS	CLOSURE_REASON	CASE_TITLE	SUBJECT	REASON	TYPE	QUEUE
1	101000295615	2011-07-01 02:28:04	2011-08-01 15:21:46	Closed	Case Closed Case Resolved No Sidewalk and poor site visability would not implement any crosswalks for these reasons.	New Sign Crosswalk or Pavement Marking	Transportation - Traffic Division	Signs & Signals	New Sign Crosswalk or Pavement Marking	BTDT Sign Marki
2	101000295616	2011-07-01 03:03:48	2011-07-22 16:13:45	Closed	Case Closed Case Resolved completed	Street Light Outages	Public Works Department	Street Lights	Street Light Outages	PWD Light (Inter
3	101000295617	2011-07-01 03:12:31	2011-07-01 06:12:38	Closed	Case Closed Case Resolved	Highway Maintenance	Public Works Department	Highway Maintenance	Highway Maintenance	PWD Dorc
4	101000295618	2011-07-01 03:43:07	2011-07-05 09:16:10	Closed	Case Closed Case Noted	Notification	Mayor's 24 Hour Hotline	Notification	Notification	INFO

Figure 2 Snippet of Boston 311 data

4.2.1 Boston 311 Data on Map

In this visualization we plotted the data on an interactive map. When we select individual points, we see information of that 311 issue, for example the one selected in the image above.

Like mentioned above we have even enabled the clustering setting which allows for better readability and we can zoom into the areas from where information is more relevant to us.

Figure 3 Map of 311 Data

4.2.2 Boston 311 issues Count Vs Department

In this visualization we have plotted the different counts of issue we see across the different issue subjects. This visualization is very important in understanding which departments have the most issues coming to them. For example, we see that the "Public Works Department" has received over 800K issues.

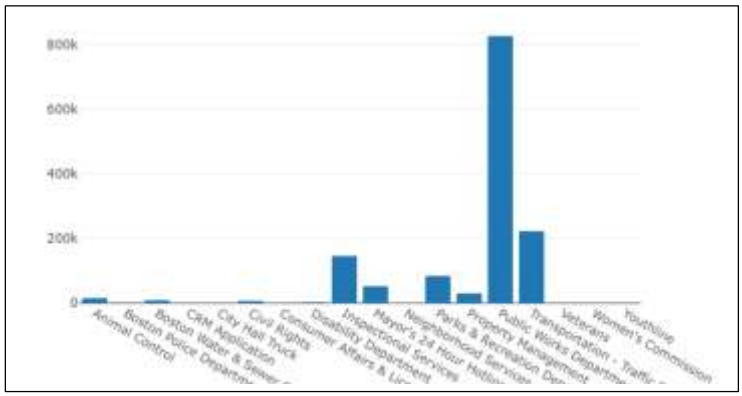

Figure 4 Counts of Issues Vs Department

4.2.3 Boston 311 issues Count Vs Date

In this visualization we have plotted the different counts of issue we see across time series. This visualization lets us understand around what time of the year do which departments receive their peaks of issues coming to them. For example, we see that between September and October was when most departments received their peaks of issues over the year.

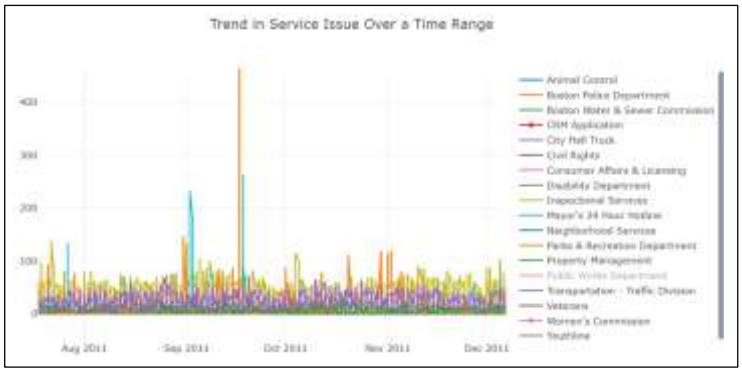

Figure 5 Counts of Issues Vs Date

5 FUTURE WORKS

We came up with very good approach to combine the data we have for Boston 311 with the data we have from tweets. We think we can use Zero-Shot Learning for Text Classification [6] this method could let us classify the tweets into subjects and case titles similar to the 311 data and as such provide a common ground between the two and this could lead to even more analysis that can be done via the combination of the two datasets. For example, we can combine the data for a political figure and figure out which Boston 311 department's issues are usually tweeted out to this person's account. This could help Involved combine two different datasets and provide a more thorough analysis for their clients.

If we had access to the paid sectors of twitter data, we could look up more historical data and we can combine this analysis with whether it is an election year and other such factors, and give an analysis of when a political figure needs to be more active on Twitter, since around that time more tweets would be sent to their account.

We can use historical data and use sentiment analysis on this data to get a general idea of that person's approval ratings over the course of the year or the term. We can then generate separate word weightages for each of the sentiments and figure where a representative is good at his/her job and where he/she is lacking and as such can help improve that representative.

6 REFERENCES

[1] "Tweepy - Python Twitter API wrapper," [Online]. Available: http://docs.tweepy.org/en/3.7.0/getting_started.html.

[2] "Natural Language Toolkit," [Online]. Available: https://www.nltk.org/.

[3] "Word Tokenize," [Online]. Available: https://www.nltk.org/api/nltk.tokenize.html.

[4] "TFIDF," [Online]. Available: http://www.tfidf.com/.

[5] "Folium," [Online]. Available: https://python-visualization.github.io/folium/.

[6] M. M. S. Pushpankar Kumar Pushp, "Train Once, Test Anywhere: Zero-Shot Learning for Text Classification," [Online]. Available: https://arxiv.org/abs/1712.05972.