Data Collection of Political Tweets on US Elections with Twitter and 4Chan

Bhuvan Chadha
Computer Science
SUNY Binghamton
bchadha1@binghamton.edu

Daniel Paul
Computer Science
SUNY Binghamton
dpaul12@binghamton.edu

Naveen C. Poda
Computer Science
SUNY Binghamton
npoda1@binghamton.edu

Naresh K. Anantharam
Computer Science
SUNY Binghamton
nananth1@binghamton.edu

INTRODUCTION

Our project, as mentioned in the proposal is focused on analyzing political tweets in twitter. We used the *requests* library for python and streamlined the live tweets with the help of Twitter APIs. We have also used 4chan as a secondary data source to compare data from both data sources the first one being twitter.

KEYWORDS

Twitter, Tweets, 4Chan, Data Collection, MongoDB

CHALLENGES

Our project, as mentioned in the proposal is focused on collecting political tweets made on the ongoing election results in the US. Our initial methodology of data collection was to collect historical posts from Robinhood. But this was not possible as there was access and authentication issues to access Robinhood due to various reasons. Hence as a secondary data source we had switched to 4chan and the subcategory of pol in 4chan to collect data on various political tweets or /*pols/*. As the data was being collected, the trend observed was that there were fewer posts at the beginning of elections and the number of posts increased immensely as the campaign went on and as the voting had begun. For this reason, we had collected data from both twitter and 4chan at various timeframes like at the beginning of the campaign, during the voting sessions and after the results were declared. Finally, we ended up collecting all the tweets based on hashtags of contestants, usual slogans etc. This would give a wider dataset for performing sentiment analysis as next steps. This dataset could be refined and cleaned further to derive the results that are required.

METHODOLOGY

To collect tweets pertaining to the requirement, a certain number of hashtags were isolated and based on them, tweets were collected. Using the real-time tweet streamlining API, we were able to gather around 10,000 tweets that matched the hashtags which we used as filters for the data collection. The gathering of data was done on election days - before, during and after the results were declared. Our choice of hashtags for the tweets collection was based on campaign slogans, contestants, and keywords that we wanted to closely follow. The hashtags used for the elections were of the format '#trump', '#MAGA', '#trump2020', '#Biden', '#uselection' etc. This in turn, can provide further information to collect even more tweets relating to those incidents. As the election vote counting started this week, collecting these tweets would enlarge our dataset adding new data which can be analyzed along with our existing data. For the data collected from 4chan we did not use any specific keywords or hashtags as there is a predefined category called pol especially for the politically related posts.

IMPLEMENTATION

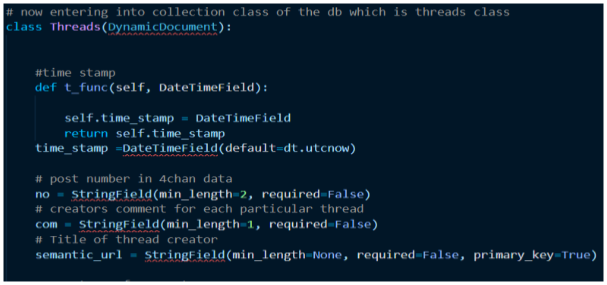
At first, we ran our code on one of our local machines and collected data there. This was done in python using the Jupyter Notebook.

Coming over to data from 4chan as mentioned earlier we have collected data from pol subcategory. For this we had to take a snapshot of the first 10 pages at one point of time since the pages load dynamically. We had imported suitable packages like bs4, datetime,urllib,requests and MongoEngine.

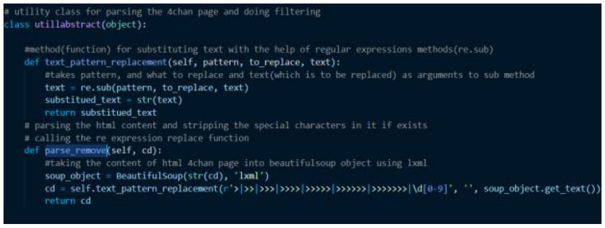
The data we have collected comprises of 4 main components called poll name, timestamp at which it was posted, postid and comments on it. This was done by using a method utillabstract to filter the 4chan page and parse it accordingly. We also had to use another method text_pattern_replacement to substitute text with the help of regular expressions methods. During this process we had encountered few special characters which had to be stripped out and for this we had used the parse_remove method.

The Twitter data collection is implemented iteratively using the *requests.get ()* function. There are 6 attributes which we were able to retrieve which include the main text, author id, creation date, language of the tweet, possible sensitive and the public metrics.

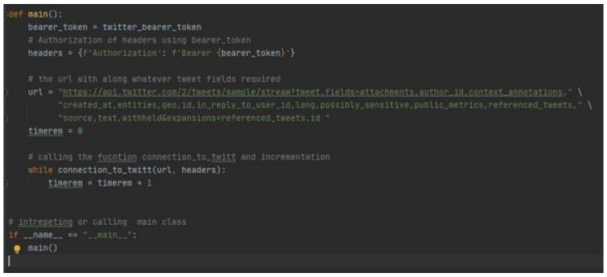
Below are some of the code snippets of the 4chan and Twitter implementation.



```python
# now entering into collection class of the db which is threads class
class Threads(DynamicDocument):


    #time stamp
    def t_func(self, DateTimeField):

        self.time_stamp = DateTimeField
        return self.time_stamp
    time_stamp =DateTimeField(default=dt.utcnow)

    # post number in 4chan data
    no = StringField(min_length=2, required=False)
    # creators comment for each particular thread
    com = StringField(min_length=1, required=False)
    # Title of thread creator
    semantic_url = StringField(min_length=None, required=False, primary_key=True)
```



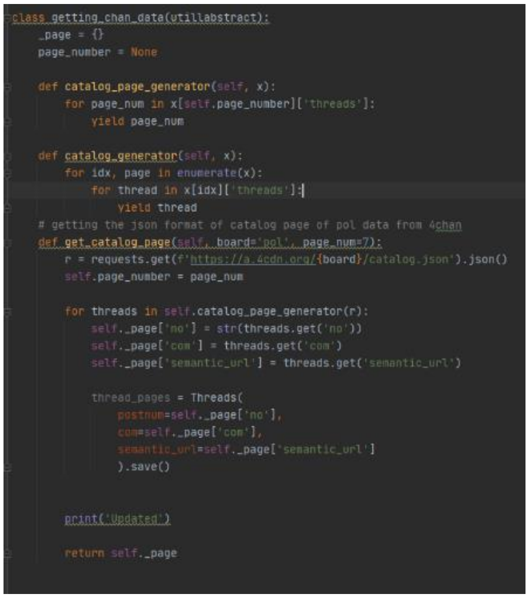
```python
# utility class for parsing the 4chan page and doing filtering
class utillabstract(object):

    #method(function) for substituting text with the help of regular expressions methods(re.sub)
    def text_pattern_replacement(self, pattern, to_replace, text):
        #takes pattern, and what to replace and text(which is to be replaced) as arguments to sub method
        text = re.sub(pattern, to_replace, text)
        substitued_text = str(text)
        return substitued_text
    # parsing the html content and stripping the special characters in it if exists
    # calling the re expression replace function
    def parse_remove(self, cd):
        #taking the content of html 4chan page into beautifulsoup object using lxml
        soup_object = BeautifulSoup(str(cd), 'lxml')
        cd = self.text_pattern_replacement(r'>|>>|>>>|>>>>|>>>>>|>>>>>>|>>>>>>>|\d[0-9]', '', soup_object.get_text())
        return cd
```



```python
def main():
    bearer_token = twitter_bearer_token
    # Authorization of headers using bearer_token
    headers = {f'Authorization': f'Bearer {bearer_token}'}

    # the url with along whatever tweet fields required
    url = "https://api.twitter.com/2/tweets/sample/stream?tweet.fields=attachments,author_id,context_annotations," \
          "created_at,entities,geo,id,in_reply_to_user_id,lang,possibly_sensitive,public_metrics,referenced_tweets," \
          "source,text,withheld&expansions=referenced_tweets.id "
    timerem = 0

    # calling the fucntion connection_to_twitt and incrementation
    while connection_to_twitt(url, headers):
        timerem = timerem + 1


# intrepeting or calling  main class
if __name__ == "__main__":
    main()
```

DATA STORAGE

For data storage we had used MongoDB as our database. The advantage of using this database is that we were able to arrive at a situation where the format of the data being collected was arranged as per the column names or filters we had used irrespective of the order in which it was collected.



```python
class getting_chan_data(utillabstract):
    _page = {}
    page_number = None

    def catalog_page_generator(self, x):
        for page_num in x[self.page_number]['threads']:
            yield page_num

    def catalog_generator(self, x):
        for idx, page in enumerate(x):
            for thread in x[idx]['threads']:
                yield thread
    # getting the json format of catalog page of pol data from 4chan
    def get_catalog_page(self, board='pol', page_num=7):
        r = requests.get(f'https://a.4cdn.org/{board}/catalog.json').json()
        self.page_number = page_num

        for threads in self.catalog_page_generator(r):
            self._page['no'] = str(threads.get('no'))
            self._page['com'] = threads.get('com')
            self._page['semantic_url'] = threads.get('semantic_url')

            thread_pages = Threads(
                postnum=self._page['no'],
                com=self._page['com'],
                semantic_url=self._page['semantic_url']
                ).save()


        print('Updated')

        return self._page
```

DATA VISUALIZATION AND STATISTICS

Using the data, we had collected, we moved on to visualize it into a chart to have a better look at how much data was collected over time. Below are those graphs we had obtained.

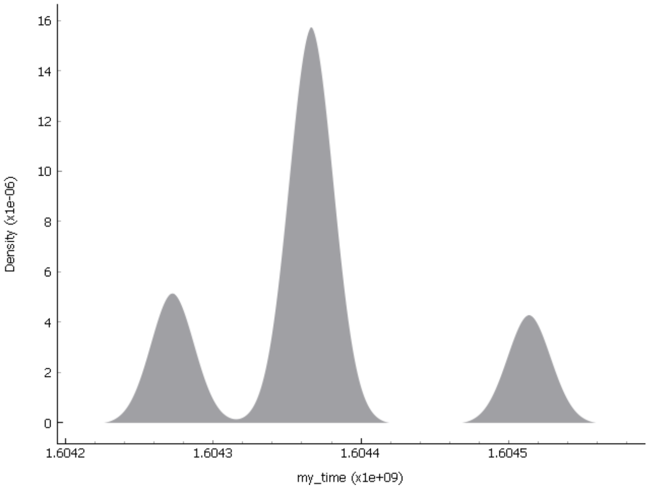


Figure 1. 4chan data Before counting

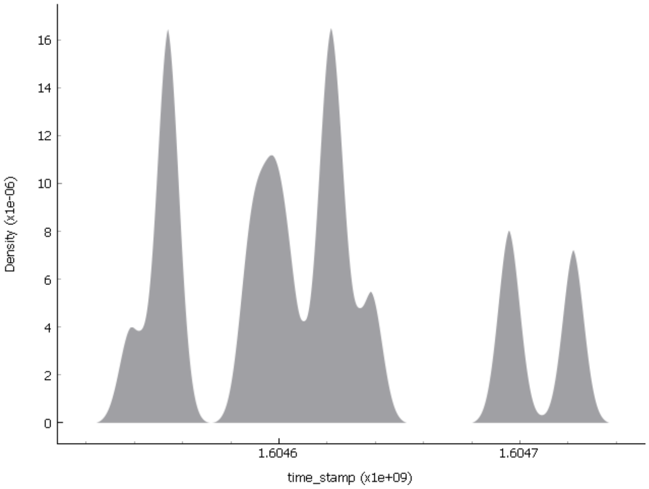


Figure 2. 4chan data after counting

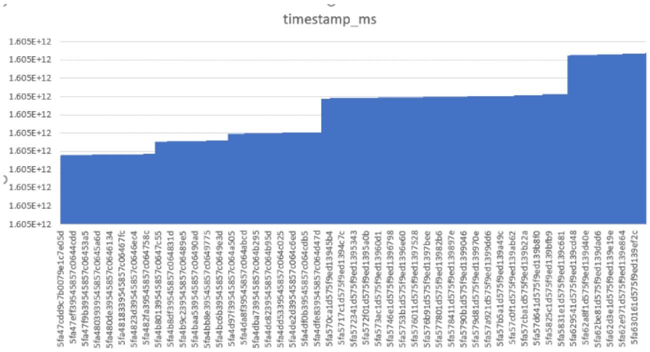


Figure 3. Twitter data before counting

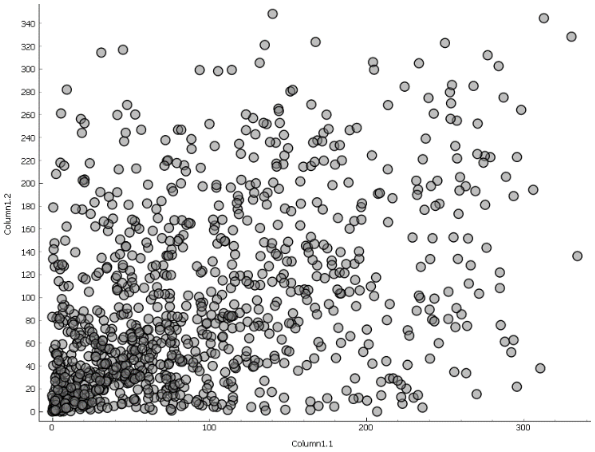


Figure 4. Twitter Data before counting

REFERENCES

[1] Twitter Developer Docs https://developer.twitter.com/en/docs/apps/overview

[2] 4Chan website https://4chan.org

[3] Yehia Khoja. 2019. Twitter data collection tutorial using Python https://towardsdatascience.com/twitter-data-collection-tutorial-using-python-3267d7cfa93e

ACKNOWLEDGMENT

It gives us immense pleasure to thank Prof. Jeremey H Blackburn for guiding us through this project and helping us improve in the subject with all his knowledge, expertise, and experience.