Geo-visualizing Flooded Areas Utilizing Twitter API and Google Maps API

RENCE LENNARD A. GALARAGA

Ateneo de Davao University, renlagalaraga@addu.edu.ph

RAY JAMES AMER L. GONZALES

Ateneo de Davao University, rjagonzales@addu.edu.ph

KYLE STEPHEN R. TANCIO

Ateneo de Davao University, ksrtancio@addu.edu.ph

School of Arts and Sciences

Ateneo de Davao University

October 2021

1 INTRODUCTION

1.1 Background of the Study

The Philippines, a tropical peninsula, is hit by many storms in a year. It is also near the equator, where storms are frequent. Associated with these storms will be flooding, as heavy rainfall from different areas contributes to the rise of water levels of rivers and seas. Because of this, many people need rescue efforts because there are places within the city that are flooded frequently. During flooding situations, some people might be trapped within their homes as the water rises. The central 9-1-1 system may be overwhelmed with calls from areas in need of assistance, and as a result, they might not be able to respond to each call. There are emergency respondents that volunteer themselves in times of dire need. The researchers would like to develop a tool capable of bridging the gap between the people in need during flooding and the volunteer respondents. The Davao River, Matina River, and Talomo River are among the rivers situated within Davao City. If there are storms and heavy rains, some city areas become submerged due to flooding, particularly those near the previously mentioned bodies of water. Modern technology has brought us the power of instant information at our fingertips. With the birth of the internet and the creation of online social networks, the dissemination of information is broader than ever. With people posting status updates on social media, the researchers would like to harness its potential through developing a data-gathering method that utilizes status updates on Twitter to visualize possible areas that are flooded and get quicker information regarding affected areas. The researchers will develop this hypothetical tool, as it might have further impacts for faster information gathering.

2 PROBLEM STATEMENT

This study is conducted because the researchers would like to create an alternative way of gathering and presenting information where flooding within Davao City is involved. A data collection tool will be developed, and a data representation tool will be integrated into a website where residents of Davao City will be able to access.

The study aims to answer the following questions:

1. How can tweets be gathered using Twitter API?
2. What are the bases on which a tweet is considered relevant in the context of flooding disasters?
3. What areas within Davao City are highly documented by Twitter?
4. How reliable is the developed website when it comes to getting information and visualizing the data regarding flooding within Davao City?
5. How to utilize Google Maps API in the context of Geo-Visualization?

3 OBJECTIVES OF THE STUDY

The study has the following general objective:

To create an alternate way of data-collection relating to the flooding within Davao City which is accurate and reliable.

The study has the following specific objectives:

1. To identify how tweets are gathered utilizing Twitter API.
2. To determine what are the basis on which a tweet will be considered relevant in the context of flooding disasters.
3. To determine what areas are documented more from tweets.
4. To identify the reliability of the developed website when it comes to getting information and visualizing the data regarding flooding within Davao City.
5. To Identify how Google Maps API was implemented in the context of geo-visualization.

4 SIGNIFICANCE OF THE STUDY

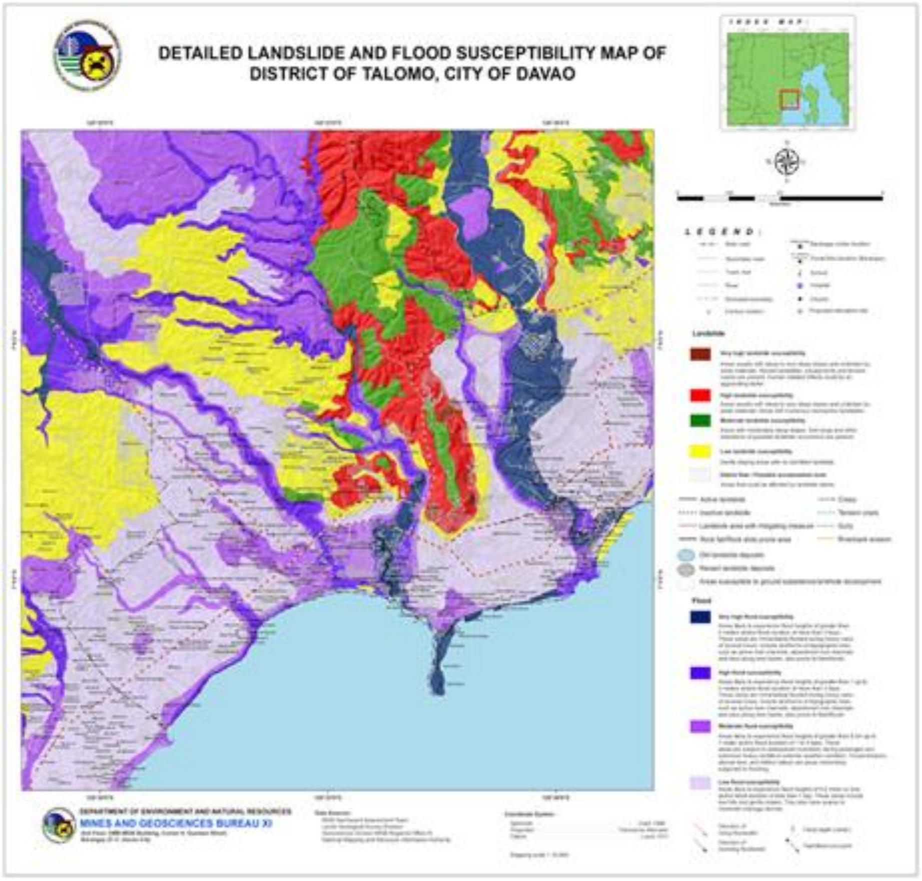

DETAILED LANDSLIDE AND FLOOD SUSCEPTIBILITY MAP OF DISTRICT OF TALOMO, CITY OF DAVAO
LEGEND:
DEPARTMENT OF ENVIRONMENT AND NATURAL RESOURCES
MINES AND GEOSCIENCES BUREAU XI

Figure 1.4-A: A detailed landslide and flood susceptibility map of District of Talomo, City of Davao

The researchers propose to create an information system that harnesses the power of one of the social media: Twitter, to gather information talking about flooding within the area of Davao City. An example can be seen in the map presented in Figure 1.4-A, areas with dark shades of blue are the areas that have a very high flood susceptibility within Talomo District. The development of an alternative data collection system will be beneficial for the volunteer responders who want to have immediate information regarding areas that need help, and for the emergency responders of Central 9-1-1 to have an alternative source of information that will be supplemental to their current system. This research will also benefit future researchers who will delve into the topic of using Twitter API and geo-visualization of data.

5 SCOPE AND LIMITATIONS

The researchers are developing a text mining tool and a geo-visualization tool which will be integrated into an output website that people can access. This research will only cover areas situated within Davao City only and will not cater to any results that are beyond the borders of the city. The researchers will only use Twitter as their source of raw information. The project will only gather tweets that are made public and will not include any tweets coming from private accounts on Twitter. The research will only gather tweets that are made in the Filipino and/or English language. The research project will only retrieve tweets during the span of 1 to 2 months from the creation of the output website. The research will only search for past tweets that are posted during a flood advisory that is in place which will be posted and retrieved manually by the researchers from the Facebook pages of Davao City Disaster Risk Reduction and Management Office, and the City Government of Davao.

6 REVIEW OF RELATED LITERATURE

6.1 Flooding in the Philippines

Over the past decades, recurrent flooding associated with typhoons, monsoons, and heavy rains, has proven to be a significant issue in The Philippines. Recent large-scale floods associated with typhoons Haiyan (Yolanda) in 2013, Bopha (Pablo) in 2012, Nesat (Pedring) and Washi (Sendong) in 2011, as well as Ketsana (Ondoy) and Parma (Pepeng) in 2010, have all led to massive flooding and thousands of casualties (Cadag et al., 2017). Flooding in the Philippines is associated with many other factors, such as rampant deforestation, clogged waterways, and blocked sewage systems. Consequently, leading news agencies and newspapers have recently tagged the Philippines as the 'Baha (flood) Republic. It also appears that climate change has become a popular explanation for flooding. The Filipino government created geo-hazard maps, which outline areas prone to natural disasters, and are publicly available to reduce vulnerability at a community level. The geo-hazard maps, developed by the Mines and Geosciences Bureau of the Department of the Environment and Natural Resources, uses color to classify areas as low, moderately, or highly susceptible to floods, flash floods, and landslides. There are marked areas from being prone to riverbank erosion (DENR, 2009). This study benefits the whole research by looking at the past research and actions towards flooding in the country and thus will make it easier as to what steps should be taken next or what research should be done.

6.2 Web Scraping and Text Mining

A technique that extracts data from the World Wide Web (WWW) is called web scraping, also known as web extraction or harvesting. The process of scraping data from the Internet can be divided into two sequential steps: acquiring web resources and then extracting desired information from the acquired data. Specifically, a web scraping program starts by composing an HTTP request to obtain resources from a targeted website. After extracting, the data is stored in a file system or database, which could be used for later retrieval or analysis. Hypertext Transfer Protocol (HTTP) or any web browser is utilized to scrape data found on the web, which is then accomplished either by a user, manually, or by a bot or web crawler automatically.

According to Mooney et al. (2015), web scraping is commonly recognized as an effective and versatile method for gathering large amounts of data since the WWW generates an overwhelming amount of heterogeneous data regularly. Web scraping strategies have evolved from more minor ad hoc techniques to respond to a variety of situations, from human-assisted processes to fully integrated applications capable of converting whole websites into well-organized data sets. As stated by Butler (2017), state-of-the-art web scraping tools are capable of parsing markup languages or JSON files and integrating with computer visual analytics.

Text Mining, sometimes referred to as Text Data Mining, or the knowledge discovery from textual databases is generally referring to the process of extracting the important data, by a bot that is developed programmatically which has the right algorithm such as a deep learning algorithm to transform unstructured text into a structured one which can be used for data analysis (MonkeyLearn, n.d.; Tan, 1999). As mentioned by IBM (n.d.), the text data in which the text miner will be encountering can be organized into 3 categories: Structured Data where data is in a tabular format where it is categorized through a series of rows and columns, Unstructured Data where it does not contain any predefined format, and Semi-structured Data which is a combination between the aforementioned categories.

There is also another concept that is similar to text mining in terms of approaches: text analytics. While both deal with text data as its source, the difference between the both of them is that text analytics is an older discipline which deals with the computational linguistics field, and text mining is a newer discipline that is also related to the fields of statistics, data mining, and machine learning (Varun, n.d.). By defining web scraping and text mining, the researchers will be given an idea as to how these two relate to each other.

6.3 Text Mining in Online Social Networks

Through these social networks, people can quickly and confidently share their points of view regarding various global issues by uploading their posts, text comments, and blogs. As people write words or sentences with errors, to let them write or search with proper grammar and structured sentences, the text mining approach is used. In social networking websites, individuals, for the most part, use unstructured or semi-structured language for correspondence; therefore, extracting logical patterns with exact data from such unstructured types of data is a complicated job. The utilization of text mining techniques on social networking websites can uncover significant results about communication practices between people (Salloum et al., 2017). It is to be noted that some online social media disallows crawling on their website. Facebook for example, in their site scraping terms of service page, has deliberately mentioned that automated data collection in their website is not allowed unless permitted by Facebook itself (Facebook, 2010). In this literature review, the researchers will have an understanding of the rules and terms of services that surround text mining and will be given a precaution when text mining online social networks.

6.4 Twitter API

A study done by Ablazo (2019) has stated that Twitter's rules regarding the sharing of data are lenient and the platform even provided a Streaming API, which performs functions such as searching for queries that are needed by a user. As of this time of the research, Twitter has made two versions of the API which are v1 and v2. According to Twitter Developer's website, Twitter API v2 is the new version that precedes Twitter API v1. With this knowledge, the researchers will have an idea as to what API will be used in the project to retrieve tweets.

6.5 Tweepy

Tweepy is open-sourced, hosted on GitHub, and enables Python to communicate with the Twitter platform and use its API. Tweepy provides access to the well-documented Twitter API. With Tweepy, it is possible to get an object and use any method that the official Twitter API offers. Tweepy is an excellent open-source library that provides access to the Twitter API for Python. Although the documentation for Tweepy is a bit lacking and does not have many demonstrations, the fact that it depends on the Twitter API, which has excellent documentation, makes it probably the best Twitter library for Python.

6.6 Visualizing Data from Mined Text

Visualizing data from mined texts is a relatively complicated process. Structuring uncategorized data into a readable format is a difficult task for many, as such, there are occupations solely dedicated to transforming this data into a comprehensible form for companies to understand. As explained in a blog by MonkeyLearn (2019), it is possible to transform that data into quantitative charts and reports. An example that they use is customer feedback on a specific product, in which they graph the text data by tagging each response – whether that's before or after the response itself is sent is irrelevant – and quantify it by how many people promote it, feel passive about it, or overall dislike what was tagged. Tags can range from Product Features, Pricing, and even Customer Support reviews. The overall process of data visualization can be summarized into two steps; Analyzing data, and then using Business Intelligence tools to try to understand it. However, there are several substeps that exist within them, so it is much more complicated than the two steps initially appear to be. With this in mind, the researchers of this paper have confirmation that visualizing data from text is possible. With this information, the researchers have an idea as to what are the possible techniques that can be used in visualizing the data that was gathered.

6.7 Geo-visualization

Geo-Visualization, as defined by Storie J. (n.d.), is a critical tool to understand potential urban issues. Or, more specifically, uses a cartology of a specific area to display relevant data. Through the example given in the researcher's paper, the researchers can effectively see the locations where green-colored high-earning families live and where red-colored low-earning families live. It displays an urban location split by a river, of which the northern side has a poorer income than the southern side. This has numerous ramifications, such as giving recruiters potential locations to offer job opportunities. For the intents and purposes of this paper, the researchers plan to do a similar process to display locations in the urban city of Davao that are in dire need of aid during flooding or similar natural disasters. This literature on the topics where the research will revolve; this literature gives the researcher an idea as to what is Geo-visualization by defining it, and an idea as to how to do it.

6.8 Google Maps API

Google offered different API services which developers can utilize. In this research, the researchers will be utilizing APIs from Google Maps because the research is dealing with geo-visualization. One of Google's most valuable products is its location service, Google Maps. Many people use it to get from point A to point B; businesses like Uber and Lyft entirely depend on Google's location service. Google Maps tracks and keeps data on literally every step you take if the device's location services, or GPS is enabled. This location history data can then be utilized to display the number of locations points and pins you visit over some time. This generates a heatmap that is referred to as a Google Maps Heatmap. A heatmap is a display used to visualize the intensity of data at geographical marks. When the Heatmap Layer is on, a colored overlay will show on top of the map. By standard, areas of higher intensity will be marked red, and areas of lower intensity will show as green. Heatmap will benefit the research in terms of visualizing the data by marking particular points on the map according to the data collected. The Google Maps API also offers a geocoding service which is a utility to identify location coordinates which will be used in the geo-visualization process.

6.9 Text Mining and Geo-visualization

Generating geographical data from mined text isn't new as the research of Torget et. al. (2011) has already done some primary research regarding the possibility of extracting information from newspapers. In their research, however, they have used the technique called Optical Character Recognition (OCR) to mine text from the 230,000 newspaper samples that they got. They have also used a model in assessing language patterns: overall word counts, named entity counts, and topic models of particular date ranges (Torget et. al., 2011). This literature review provides the researchers with the idea that it is possible to integrate text mining and geo-visualization as it was done by the research of Torget et. al.

6.10 Social Media and Flood Mapping Approach

Twitter is a 'microblogging' system that lets you send and receive tweets which are short posts. Tweets can reach 140 characters long and can contain links. Twitter users follow other users. If a user follows someone, users can see their tweets in their Twitter 'timeline.' Users can pick to follow people and organizations with similar academic and personal interests. Users can create their tweets, or they can retweet information that others have tweeted. Retweeting means that information can be passed quickly and efficiently with a large number of people. A study done by Huang et. al. (2018), has looked into the possibility of harnessing the power of online social media, which in this case, Twitter, to create a faster flood mapping to administer support and create damage assessments to the affected areas. Their research team has filtered results by using Probability Index Distribution and by adding Wetness Information from post-event satellite imagery, it can give near real-time probabilities of flooded areas which is beneficial for immediate rescue attempts to people in need. With this information that was gathered in this section, the researchers already have the idea that using the online social networks as a tool to map flooded areas is possible however Huang et. al. has used a different technique in mapping areas that are flooded.

6.11 Developing a geo-visualization tool

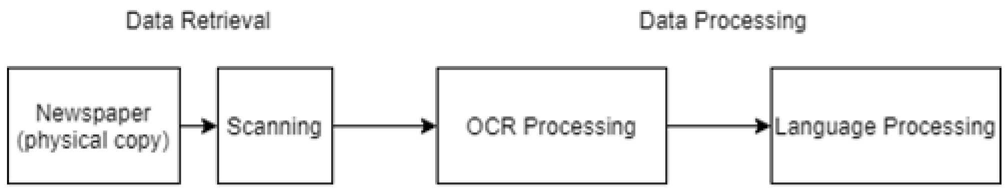


Figure 2.9-A: The process of getting data from newspapers from the research of Torget et. al.

The research of Torget et. al. (2011), has tried to create a map of the places that are mentioned in the 230,000 pages of the newspaper by utilizing Optical Character Recognition technology to convert the text from the scanned documents into a digital format. As seen in Fig 2.9-A, their process is linear in getting the needed data from the source. They have also used a quantitative model in assessing the language pattern of the retrieved text.

In the research of Huang et. al. (2018), they have managed to create a near-real-time flood probability map which helps in improving the awareness of the situation during a flooding event by looking at Twitter data and extracting its posting time, and geographic location possibly by retrieving the geotag from the post. In that research, it is much more accurate to locate where the post has been made and the accuracy of mapping because of the help of geolocation.

6.12 Django

Django is a high-level Python web framework that enables the rapid development of secure and maintainable websites. Django, built by experienced developers, takes care of much of the hassle of web development, so developers can focus on writing their app without reinventing the wheel. It is free and open-source, has a thriving and active community, excellent documentation, and many free and paid-for support options. Django helps developers write software that is Complete, Versatile, Secure, etc. Django began its development between 2003 and 2005 by a web team responsible for creating and maintaining newspaper websites. After creating several sites, the team began to factor out and reuse lots of standard code and design patterns. This standard code evolved into a generic web development framework, open-sourced as the "Django" project in July 2005.

6.13 PythonAnywhere

PythonAnywhere proved to be a freely accessible and valuable way to run Python code online without using a physical terminal. As stated in a Text-Mining paper by Sutton, S and Swickard, K (2020), where the researchers used PythonAnywhere as one of their many choices to perform Text-Mining. Their results showed that, while some measure of preparation was necessary before use, a text mining script based in PythonAnywhere allows the data preparation process to be done at an efficient rate.

6.14 Keyword Processing

Keyword processing is the ability to take in keywords from strings of data, and ensure they are viable to be used for research. As stated by a book published by Bird, S., Klein, E., and Loper, E. (2009), the researchers will be able to use one of Python's libraries to translate certain languages into English for ease of use, with such libraries as Googletrans, and eventually determine the main focus from those translations, with an NTLK function (Natural Language Toolkit). This helps avoid false positives and further improves the accuracy of the obtained data. Once data is obtained, the researchers will be able to utilize this library to aid in processing words from the data.

6.15 Keyword Classification and Mapping

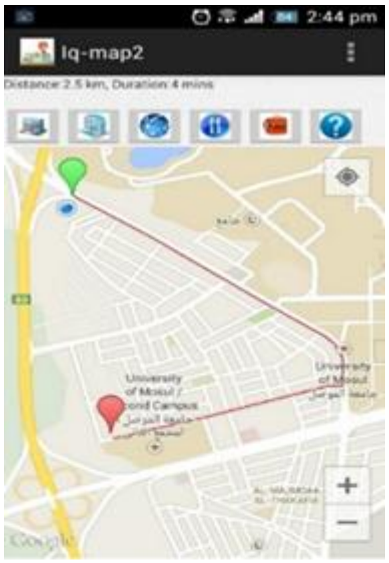

2:44 pm
Iq-map2
Distance 2.5 km, Duration 4 mins
University of Mosul / Second Campus
University of Mosul

Figure 2.13-A: A sample screenshot of the mobile app developed by Ibrahim and Mohsen (2014).

In a study by Ibrahim, O. A., and Mohsen, K. J. (2014), the researchers demonstrated that creating an accurate map of real-world locations is possible using Google API. As shown in Figure 2.13-A, this is an example of how a location will be shown on Google Maps integrated into an Android application. While the original intent of the paper was to display specified locations on a JavaScript website based on data. However, it should be possible for the researchers to switch over to Python to make use of Google's mapping API, and plot based on their data accordingly. To summarize, it is possible to fulfill the objectives of this paper by obtaining sentences from the scraped text, marking the mentioned location(s) on the mapping API if any, and plotting accordingly.

6.16 Bootstrap 5.1

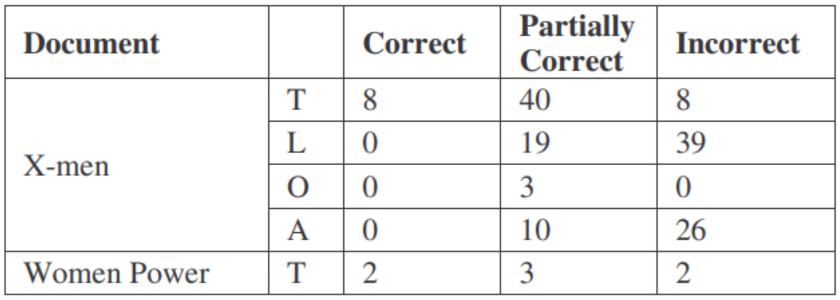
Bootstrap is a giant collection of useful, reusable parts of code written in HTML, JavaScript, and CSS. It's also a frontend development framework that lets developers and designers build fully responsive websites quickly. Essentially, Bootstrap saves you the stress of writing lots of CSS code, giving you more time to design web pages. Bootstrap is a powerful tool that enables a developer to get up and running quickly and stress-free. It makes it easy to integrate lots of great features that enrich a user's interaction with the web without coding them from scratch. This framework will help design the webpage and make it user-friendly in a short amount of time given that it saves time from coding the design from scratch.

6.17 Named Entity Recognition

In the study conducted by Mansouri et al (2011), NER or Named Entity Recognition was described to be a subproblem of information extraction in which tackles the classifying or identifying names, places, organizations, companies, and many more. The study claims that there are two integral parts to NER. The first part is the identification of the proper names and the second part is the classification of it, whether it is a name or etc. It is through this process that we hope to extract the names of locations – or more specifically streets, avenues, and cities – from public Tweets from Twitter. Mansouri et al also describe that NER to be inherent for the average human being, but for a machine, it is much harder to do so. This is because a machine lacks context in regards to the identified word. An example that the study gave was that of "The White House" being both a location and an organization, which a machine will have hard time differentiating due to a lack of context. The study also goes into detail about the various different types of NER, that is outside the scope of this paper.

6.18 Filipino Named Entity Recognition System

The research of Lim et. al (2007) created a training model for Named Entity Recognition in Filipino Texts and has acknowledged the difficulty in Named Entity Recognition as the named entity recognition relies on the capitalization but there are Filipino named entities that have a lower case such as in the case of "*Komisyon sa Wikang Filipino.*" Their research determines the performance of their Named Entity Recognition training model by checking if the tagged named entity of Tao (*Person*), Lugar (*Place*), Organization (*Organisasyon*), Miscellaneous (*Atbp.*) as Correct, Partially Correct, and Incorrect.



Document		Correct	Partially Correct	Incorrect
X-men	T	8	40	8
	L	0	19	39
	O	0	3	0
	A	0	10	26
Women Power	T	2	3	2

Figure 2.15-A: A snippet of the test results from the research of Lim et. al (2007)

From their experimental design, the "Correct" tag signifies their system was able to identify the bounds and class of the named entity, the "Partially Correct" tag signifies that (1) the bounds are correct but the class is wrong, or (2) the bounds was correct but the class is wrong.

6.19 Spacy Natural Language Processing

Spacy is an open-source library for free for Natural Language Processing in Python with many built-in capabilities. It's becoming more and more popular for processing and analyzing data in Natural Language Processing (NLP). Unstructured textual data is produced at a large scale, and it's essential to process and derive insights from unstructured data. To do that, you need to represent the data in a format that computers can understand. NLP can help you do that. Spacy offers a variety of models, but it lacks a Filipino training model for now; however, it does have a multi-language training model which will benefit the research.

7 CONCEPTUAL & OPERATIONAL FRAMEWORK

7.1 Conceptual Framework

The research that the researchers would like to propose is to combine the works of Torget et. al. and Huang et. al. to create a geo-visualization tool, which will contain processed data from Twitter.

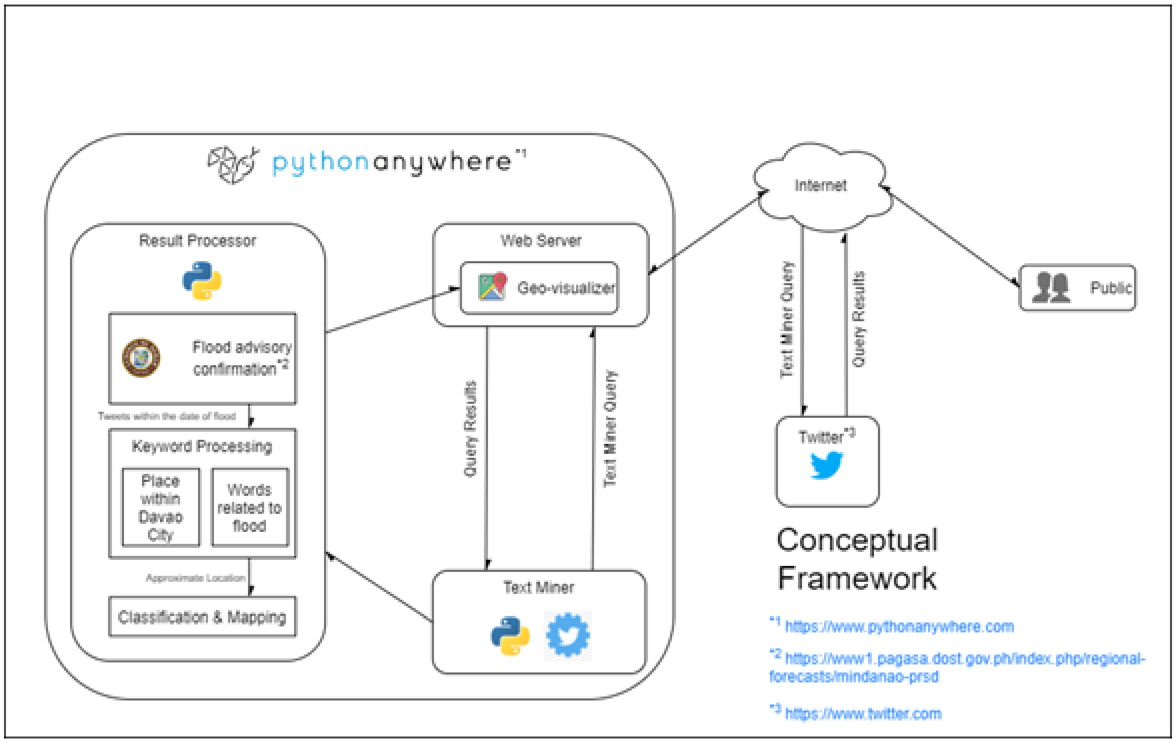


Figure 3.1-A: The proposed conceptual framework of the project

The researchers propose the system as seen in Figure 3.1-A. The researchers will be creating 3 main systems which are the webserver and the text miner and the result processor. The web server contains the geo-visualizer that uses Google Maps API to present the data that are processed by the text miner.

The next main system would be the text miner as this system will be responsible for taking data from Twitter by utilizing Twitter API. The system will retrieve tweets that will be used to be processed and confirmed by comparing dates wherein a flood advisory is posted by the City Government of Davao. Finally, the last main system would be the result processor; if the tweet/status that was retrieved corresponds with the advisory date which will be manually retrieved from the Facebook pages of the City Government of Davao, and the Davao City Disaster Risk Reduction and Management Office, then the tweet will be analyzed through in this system using keywords to check words that match a place from Davao City or is related to flooding, then it will be classified and mapped to the geo-visualizer. The website will be available for the public to access so that residents from Davao City can look and explore the website.

7.2 Operational Framework

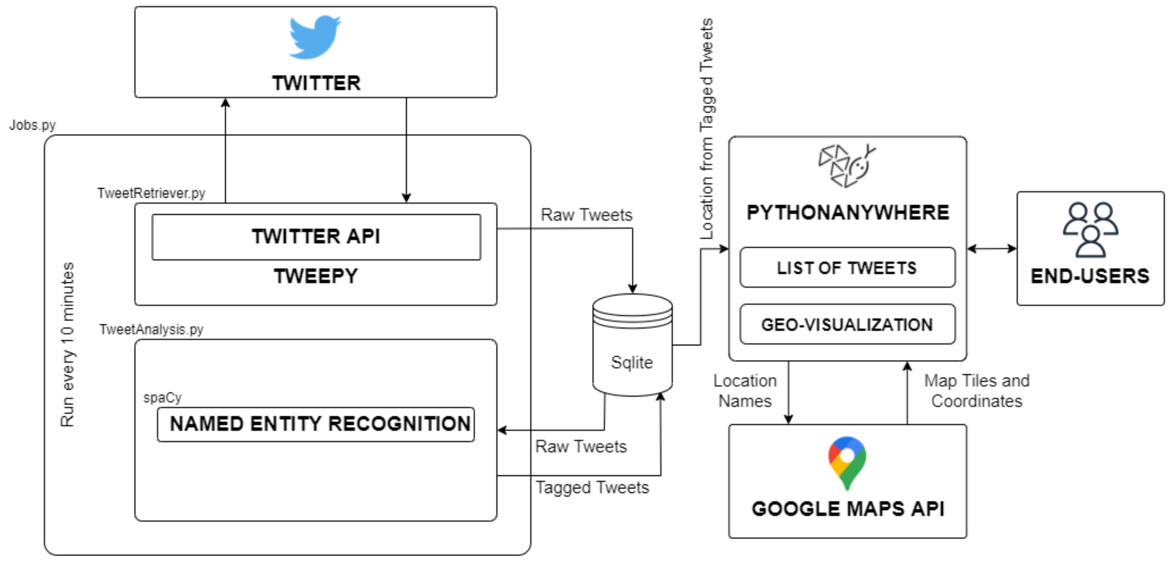
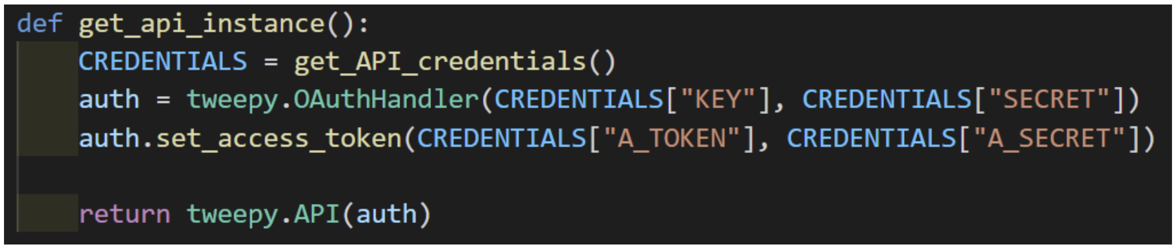


Figure 3.2-A: The Operational Framework of the research

8 METHODOLOGY

8.1 Development of the Twitter Retriever

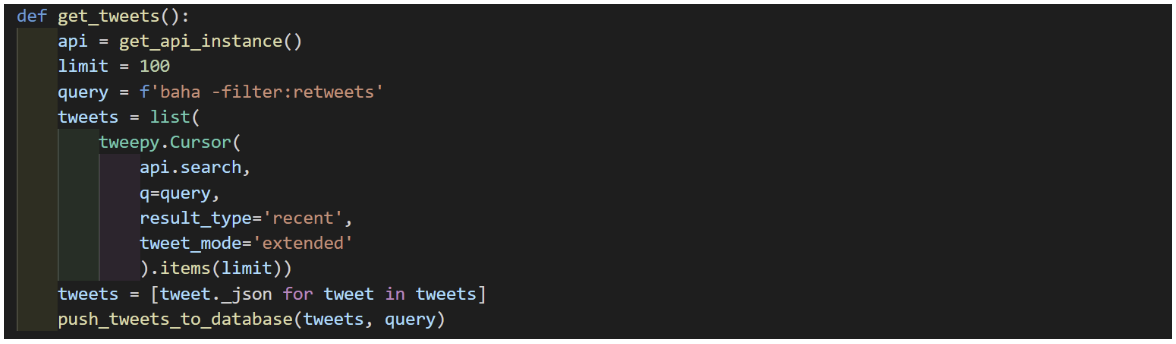
The researchers have first applied for an API key from the Twitter Developers website by creating a developer account prior to the development of the tweet retriever. After the acquisition of the API key for Twitter, the researchers are using the Tweepy Python library in order to use the Twitter API effectively on the python script to be developed.



```
def get_api_instance():
    CREDENTIALS = get_API_credentials()
    auth = tweepy.OAuthHandler(CREDENTIALS["KEY"], CREDENTIALS["SECRET"])
    auth.set_access_token(CREDENTIALS["A_TOKEN"], CREDENTIALS["A_SECRET"])

    return tweepy.API(auth)
```

Figure 4.1-A: Code snippet to get the API instance

The code for creating an instance of Tweepy is seen in figure 4.1-A. The `CREDENTIALS` variable contains the API Key, API Secret, Access Token, and the Access Secret, all of which are required in order for Tweepy to integrate to Twitter API. The researchers would then create an instance of Tweepy where the API credentials are embedded.

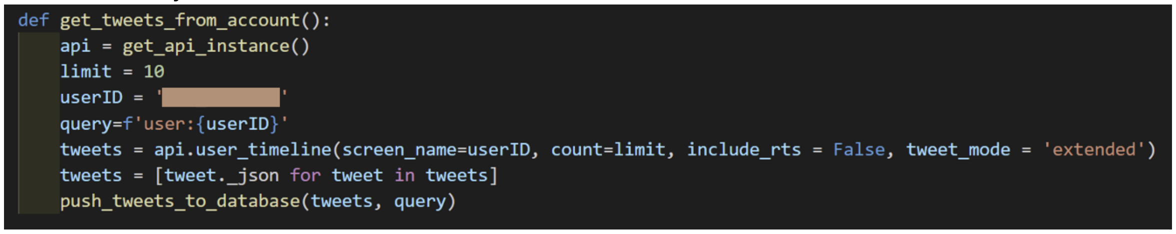


```
def get_tweets():
    api = get_api_instance()
    limit = 100
    query = f'baha -filter:retweets'
    tweets = list(
        tweepy.Cursor(
            api.search,
            q=query,
            result_type='recent',
            tweet_mode='extended'
            ).items(limit))
    tweets = [tweet._json for tweet in tweets]
    push_tweets_to_database(tweets, query)
```

Figure 4.1-B: Code snippet to get tweets from Twitter

In order to get the tweets from Twitter, the researchers have coded the function seen in figure 4.1-B. The API instance is stored in the `api` variable. The `limit` variable is the number of tweets to be retrieved when the tweet retrieving process is run. The `tweets` variable contains the typecasted result from the Tweepy instance. The researchers have set the following parameters to the instance of Tweepy: the first parameter is the `api.search` where it tells Tweepy to utilize the search API, the `q` parameter contains the query string from the `query` variable, the `result_type='recent'` parameter tells the instance to retrieve tweets that are recently made, and finally, the `tweet_mode='extended'` parameter will tell the instance to retrieve the full tweet instead of the truncated tweet which is only 144 characters long. The `items()` method sets the count of tweets that are needed to be retrieved which has the value of the `limit` variable. The first `tweets` variable contains the list of tweets and the configuration object from Twitter that is why on the second `tweets` variable, only the tweet objects which are stored in the `_json` key of the first `tweets` variable are extracted.

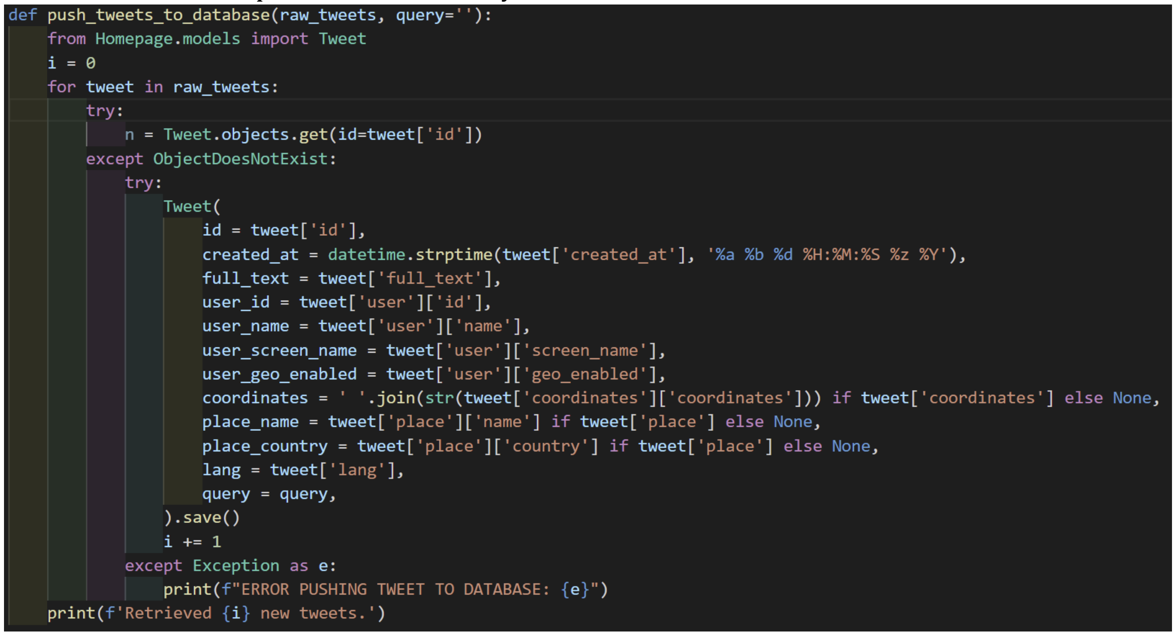
In order to get more tweets related to flooding, the researchers would activate the retrieving system during heavy rain or during an active flood advisory. This is to ensure that more data will come from Davao City.



```
def get_tweets_from_account():
    api = get_api_instance()
    limit = 10
    userID = '[redacted]'
    query=f'user:{userID}'
    tweets = api.user_timeline(screen_name=userID, count=limit, include_rts = False, tweet_mode = 'extended')
    tweets = [tweet._json for tweet in tweets]
    push_tweets_to_database(tweets, query)
```

Figure 4.1-C: Code snippet to get tweets from the testing Twitter Account

The researchers have also created a function to only retrieve tweets from the developer account made by the researchers. This is to create test cases easily and get results immediately without unnecessary noises such as unrelated public tweets. The function is almost the same as `` `get_tweets()` `` but the only difference is that it only gets tweets from a specified account, which in this case, is the developer account created by the researchers.



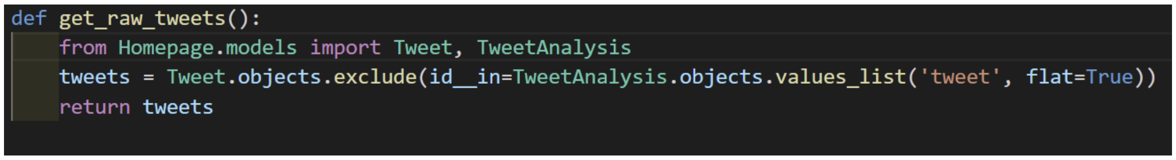
```
def push_tweets_to_database(raw_tweets, query=''):
    from Homepage.models import Tweet
    i = 0
    for tweet in raw_tweets:
        try:
            n = Tweet.objects.get(id=tweet['id'])
        except ObjectDoesNotExist:
            try:
                Tweet(
                    id = tweet['id'],
                    created_at = datetime.strptime(tweet['created_at'], '%a %b %d %H:%M:%S %z %Y'),
                    full_text = tweet['full_text'],
                    user_id = tweet['user']['id'],
                    user_name = tweet['user']['name'],
                    user_screen_name = tweet['user']['screen_name'],
                    user_geo_enabled = tweet['user']['geo_enabled'],
                    coordinates = ' '.join(str(tweet['coordinates']['coordinates'])) if tweet['coordinates'] else None,
                    place_name = tweet['place']['name'] if tweet['place'] else None,
                    place_country = tweet['place']['country'] if tweet['place'] else None,
                    lang = tweet['lang'],
                    query = query,
                ).save()
                i += 1
            except Exception as e:
                print(f"ERROR PUSHING TWEET TO DATABASE: {e}")
    print(f'Retrieved {i} new tweets.')
```

Figure 4.1-D: Code snippet to save tweets to the SQLite Database

The retrieved tweets would then get saved to the SQLite database by the `` `push_tweets_to_database()` `` function. The function would first check if the retrieved tweets are already in the database. This is necessary in order to avoid duplicate tweets within the database. If it is confirmed that the retrieved tweets are new, the function would first create an instance of a Tweet

Model object where the parameters are the values that are needed to be saved. The function would save the following information from the retrieved tweets: tweet id, creation date, the full text of the tweet, user id, user name, user screen name, user's geolocation settings (enabled or disabled), coordinates, name of the place, name of the country, the language, and the query used to get the tweet.

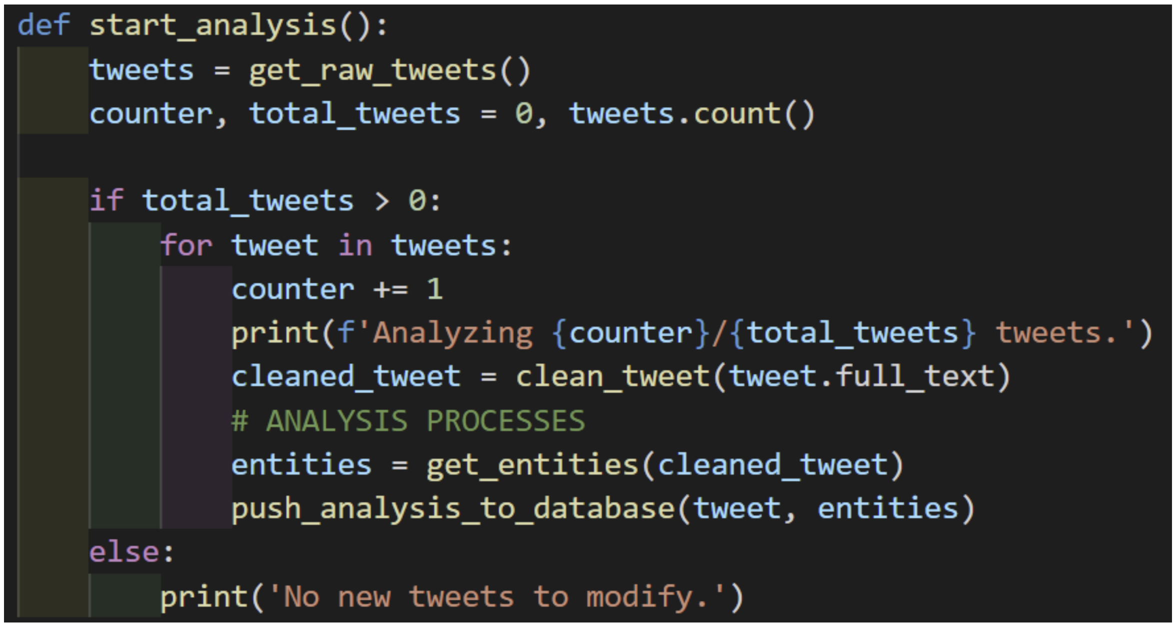
8.2 Development of the Tweet Processor/Tagger



```
def get_raw_tweets():
    from Homepage.models import Tweet, TweetAnalysis
    tweets = Tweet.objects.exclude(id__in=TweetAnalysis.objects.values_list('tweet', flat=True))
    return tweets
```

Figure 4.2-A: Code snippet to get tweets from the SQLite Database

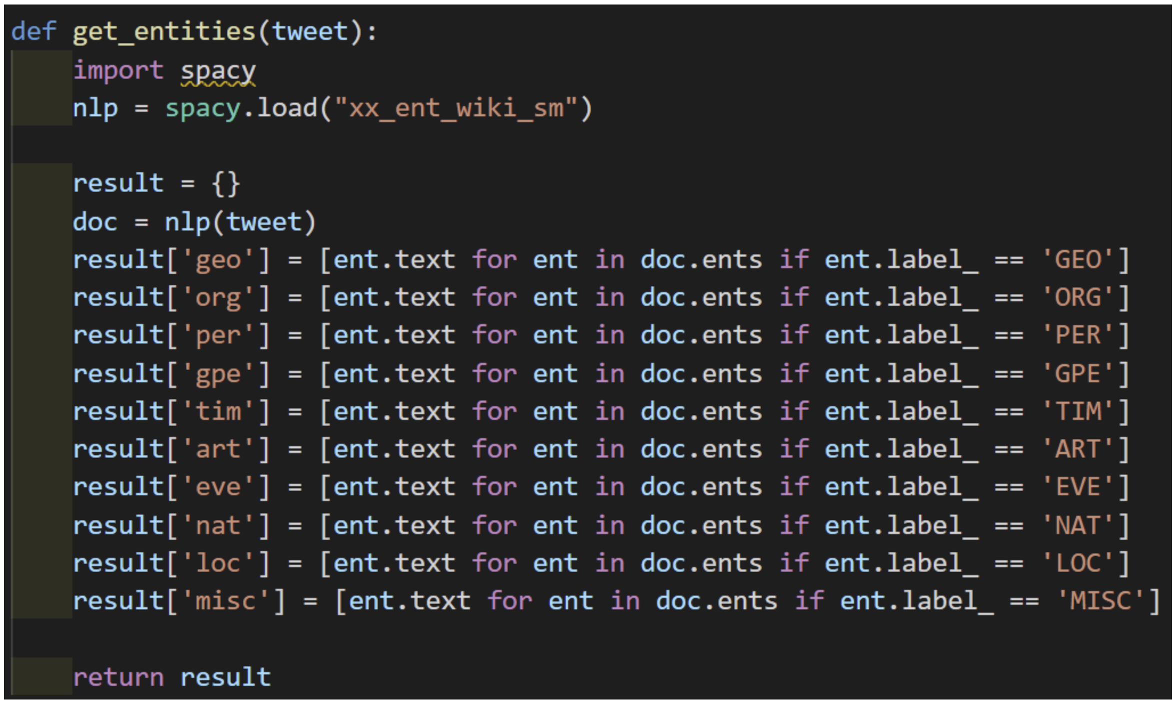
The analysis of the tweets would first start by retrieving the saved raw tweets from the database and excluding the tweets that are already analyzed/tagged.



```
def start_analysis():
    tweets = get_raw_tweets()
    counter, total_tweets = 0, tweets.count()

    if total_tweets > 0:
        for tweet in tweets:
            counter += 1
            print(f'Analyzing {counter}/{total_tweets} tweets.')
            cleaned_tweet = clean_tweet(tweet.full_text)
            # ANALYSIS PROCESSES
            entities = get_entities(cleaned_tweet)
            push_analysis_to_database(tweet, entities)
    else:
        print('No new tweets to modify.')
```

Figure 4.2-B: Code snippet for tweet analysis/tagging

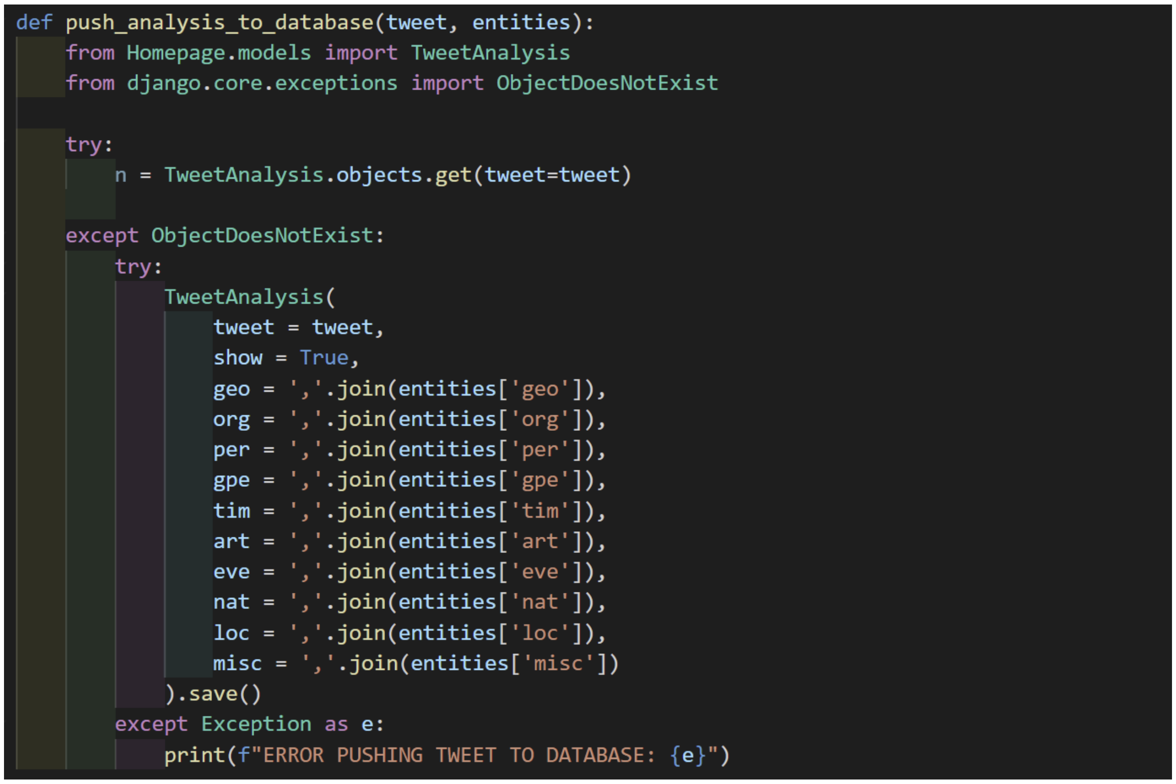
The `start_analysis()` function gets the tweets from the `get_raw_tweets()` function then for every tweet, then `clean_tweet()` function removes the mentioned users and hashtags from the tweet in order to extract locations more effectively. The `entities` variable stores the results from getting the entities from the `get_entities()` function.



```python
def get_entities(tweet):
    import spacy
    nlp = spacy.load("xx_ent_wiki_sm")

    result = {}
    doc = nlp(tweet)
    result['geo'] = [ent.text for ent in doc.ents if ent.label_ == 'GEO']
    result['org'] = [ent.text for ent in doc.ents if ent.label_ == 'ORG']
    result['per'] = [ent.text for ent in doc.ents if ent.label_ == 'PER']
    result['gpe'] = [ent.text for ent in doc.ents if ent.label_ == 'GPE']
    result['tim'] = [ent.text for ent in doc.ents if ent.label_ == 'TIM']
    result['art'] = [ent.text for ent in doc.ents if ent.label_ == 'ART']
    result['eve'] = [ent.text for ent in doc.ents if ent.label_ == 'EVE']
    result['nat'] = [ent.text for ent in doc.ents if ent.label_ == 'NAT']
    result['loc'] = [ent.text for ent in doc.ents if ent.label_ == 'LOC']
    result['misc'] = [ent.text for ent in doc.ents if ent.label_ == 'MISC']

    return result
```

Figure 4.2-C: Code snippet for getting named entities from the tweet

As seen in figure 4.2-C, the researchers have used the spacy Python Library, a natural language processing library for python, together with their 'xx_end_wiki_sm' model which is a multi-language model for natural language processing. An instance of spacy is stored in the `nlp` variable then it will be used to scan for named entities from the tweets passed from the function parameter. The entities that were specifically scanned by the users are the following: Geographical Entity, Organization, Person, Geopolitical Entity, Time Indicator, Artifact, Event, Natural Phenomenon, Location, and Miscellaneous. All of the results will be stored within the `result` variable.



```python
def push_analysis_to_database(tweet, entities):
    from Homepage.models import TweetAnalysis
    from django.core.exceptions import ObjectDoesNotExist

    try:
        n = TweetAnalysis.objects.get(tweet=tweet)

    except ObjectDoesNotExist:
        try:
            TweetAnalysis(
                tweet = tweet,
                show = True,
                geo = ','.join(entities['geo']),
                org = ','.join(entities['org']),
                per = ','.join(entities['per']),
                gpe = ','.join(entities['gpe']),
                tim = ','.join(entities['tim']),
                art = ','.join(entities['art']),
                eve = ','.join(entities['eve']),
                nat = ','.join(entities['nat']),
                loc = ','.join(entities['loc']),
                misc = ','.join(entities['misc'])
            ).save()
        except Exception as e:
            print(f"ERROR PUSHING TWEET TO DATABASE: {e}")
```

Figure 4.2-D: Code snippet for getting named entities from the tweet

After the entities are identified, the analyzed and tagged tweets will then be pushed to the database by the `push_analysis_to_database()` function.

8.3 Development of the Website Interface

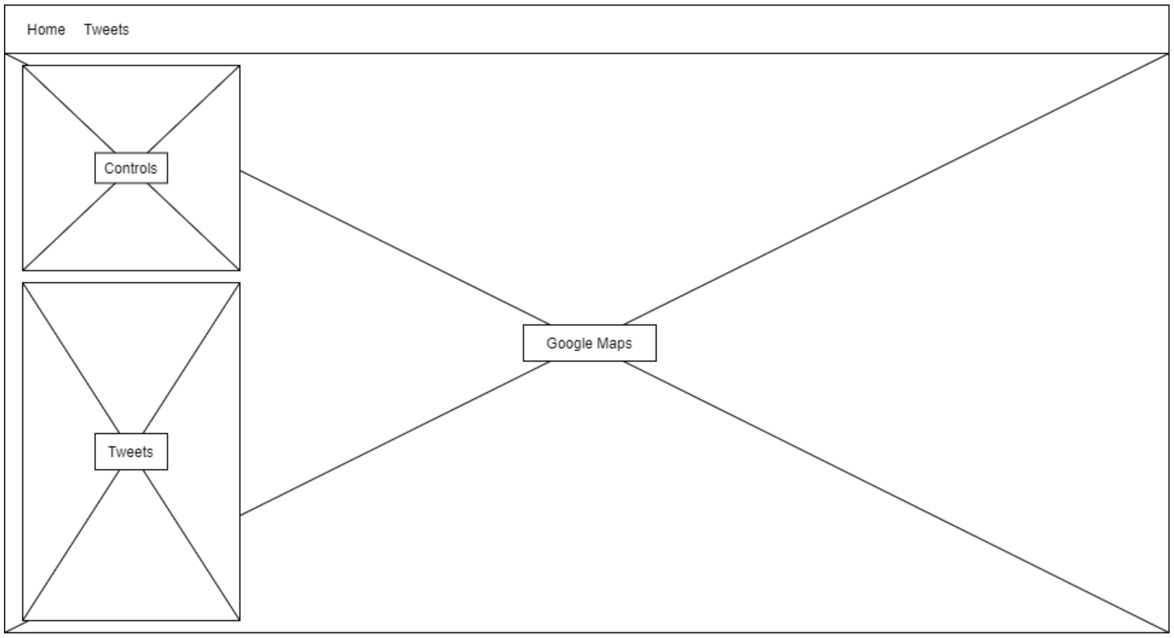
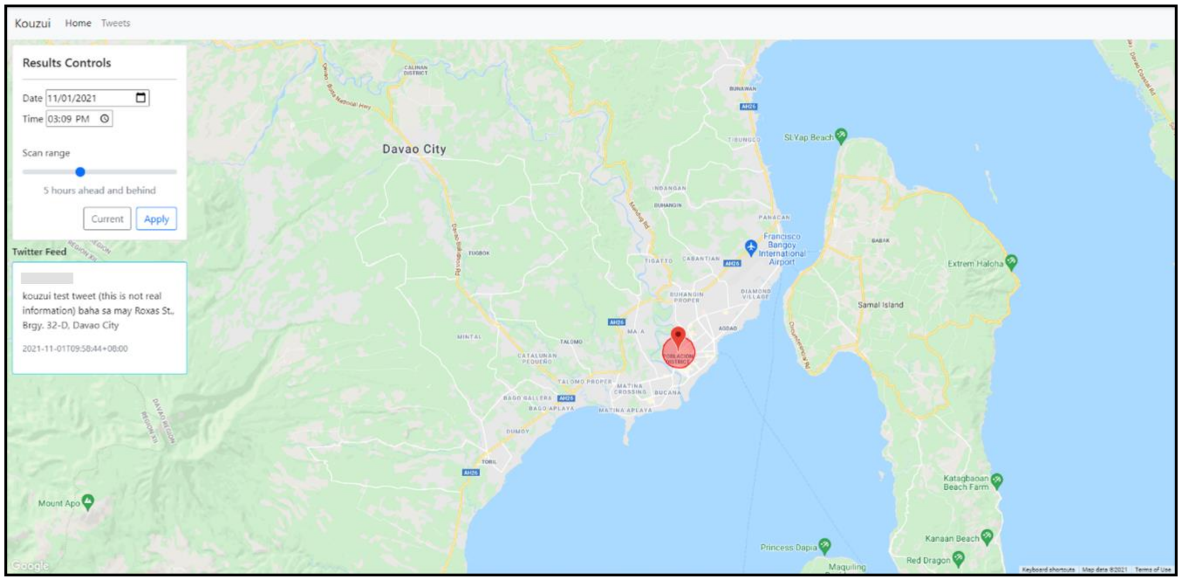


Figure 4.3-A: The planned layout for the homepage interface

The researchers have used a simple design for the homepage interface in order to create less clutter. The page will be covered by the Google Maps interface and the controls and tweets will be displayed on the left side of the page. The researchers have used Django as the backend framework, and Bootstrap as the front-end framework in developing the web interface.



Results Controls

Date 11/01/2021

Time 03:09 PM

Scan range

5 hours ahead and behind

Current Apply

Twitter Feed

kouzui test tweet (this is not real information) baha sa may Roxas St., Brgy. 32-D, Davao City

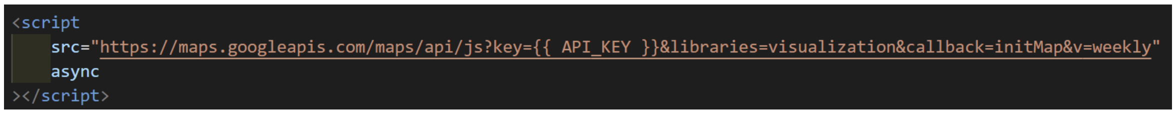
2021-11-01T09:58:44+08:00

Figure 4.3-B: The planned layout for the homepage interface

The coded webpage is seen in figure 4.3-B. The controls on the upper left are responsible for controlling the displayed tweets and locations on the map. The controls include a date and time input where the user can input specific dates to display on the map. The scan range is responsible for controlling the range of tweets that are retrieved from the database. The maximum limit for retrieving the tweets would be 12 hours as the maximum would get tweets for 1 day. The lower left part is the Twitter feed where it displays the tweets that were gathered within the time range set by the user.

8.4 Integration of the Google Maps API

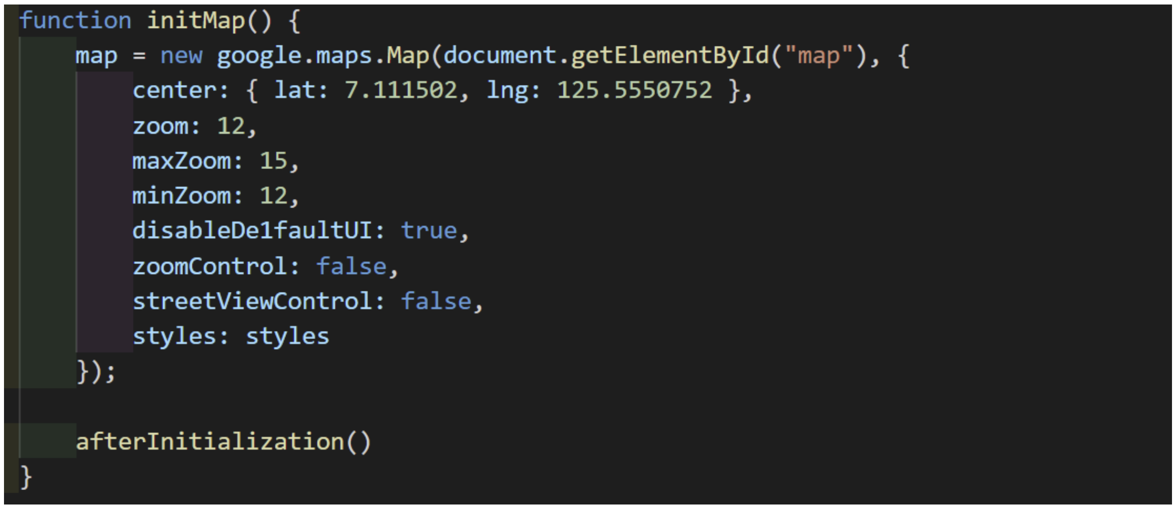
All of the processes for integrating the Google Maps API will be done on the frontend part of the developed tool.



```
<script
    src="https://maps.googleapis.com/maps/api/js?key={{ API_KEY }}&libraries=visualization&callback=initMap&v=weekly"
    async
></script>
```

Figure 4.4-A: Code snippet for embedding Google Maps API

For the integration of Google Maps API, the researchers have first applied for API access from the Google Developer Console from the Google Maps API website. After applying for access, the researchers have embedded the Google API Script to the HTML code as seen in figure 4.4-A. When the script for Google API is finished fetching data from their server, it will call a callback function that will be responsible for the initialization of the map.



```
function initMap() {
    map = new google.maps.Map(document.getElementById("map"), {
        center: { lat: 7.111502, lng: 125.5550752 },
        zoom: 12,
        maxZoom: 15,
        minZoom: 12,
        disableDe1faultUI: true,
        zoomControl: false,
        streetViewControl: false,
        styles: styles
    });

    afterInitialization()
}
```

Figure 4.4-B: Code snippet for initializing Google maps

For initializing the map, an instance of Google Maps will be stored in the `maps` variable. The instance of Google Maps will be centered on the longitude and latitude coordinates of Davao City, will have a default zoom of 12, a maximum and minimum zoom of 15 and 12 respectively, and removed default user interface, zoom controls and street controls. When the initialization process is finished, the function will call another function: `afterInitialization()` which will run functions with the already initialized map.

8.5 Testing Stage

In order to test the reliability of the developed system, the researchers will be tweeting simulated tweets from the developer account, which are related to flooding, and then check if the system has determined the location by using Named Entity Recognition and if the system has pinned the correct location on the map. For the design of the experiment, the researchers are using the research of Lim et.al (2007) but modified the feature sets to the granularity of the location since the researchers want to determine if the developed system is reliable in locating a flooded area using only the texts from the tweet.

For testing the reliability of the Named Entity Recognizer is, the researchers will be conducting experiments by combining the following features:

Legend	Features
A	City or Municipality
B	Barangay or Subdivision or Compound, District
C	Street
D	House Number

Figure 4.5-A: Table of features for the experiment

The researchers will combine the features in figure 4.5-A with the assumptions that the tweet has only one location, the tweet has a proper capitalization for named entities, and that the tweet is talking about a flood that is currently happening in Davao City.

Combination ID	Feature Combination

1	A + B + C+ D
2	A + B
3	B + C + D
4	B + C
5	C + D

Figure 4.5-B: Table of combinations of features for experiment

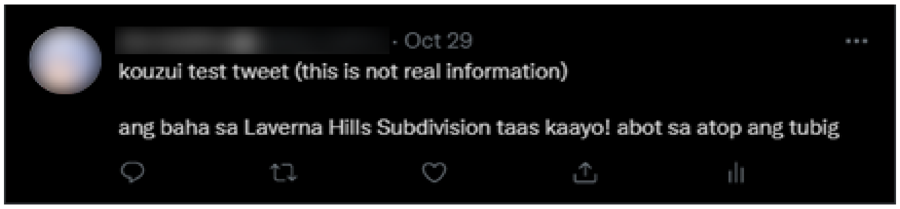
The combination can be seen in Figure 4.5-B. To be able to test the system, the researchers will be choosing 4 specific areas within Davao City that are susceptible to flooding as test cases, then test it by combining features from the feature combination and then compare the results.

8.6 Evaluation of Results

The researchers will count the number of correct, partially correct, and incorrect tags from the experiments and then look at the average in order to get the reliability of the developed system. For determining the places within Davao City that are mentioned in the tweets, the researchers will be using the tweet data from the 24th of October, 2021 where a flooding incident has happened in Davao City. The researchers will be basing the results on the named entity recognizer analysis data. The researchers have extracted the analyzed data from the aforementioned tweets through a CSV file then an analysis is made through the use of Microsoft Excel.

9 RESULTS AND DISCUSSION

The researchers have used the Twitter API in order to get the tweets from Twitter. The Twitter API v1 provides access to tweets from Twitter however since the researchers are using the free Twitter API v1, the fetched tweets are only limited to 7 days from the day of fetching the tweets. The Twitter API v1 also has a feature to search tweets that were made by a certain user; in this case, the researchers have used this feature to isolate tweets by only fetching tweets from the developer account made by the researchers.

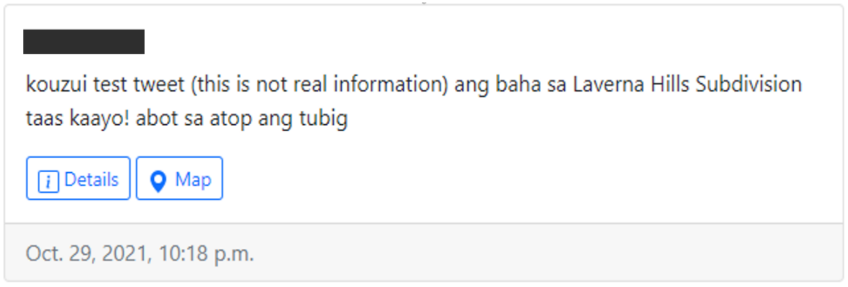

[redacted] · Oct 29

kouzui test tweet (this is not real information)

ang baha sa Laverna Hills Subdivision taas kaayo! abot sa atop ang tubig

Figure 5-A: Snippet of a tweet made from the researcher's Twitter account

For checking the reliability of the developed website, the researchers have first isolated the tweets by tweeting test cases on the developer account made by the researchers. As seen in figure 5-A, the researchers have made a disclaimer that the tweet is a test tweet then the tweet string regarding flood follows below.

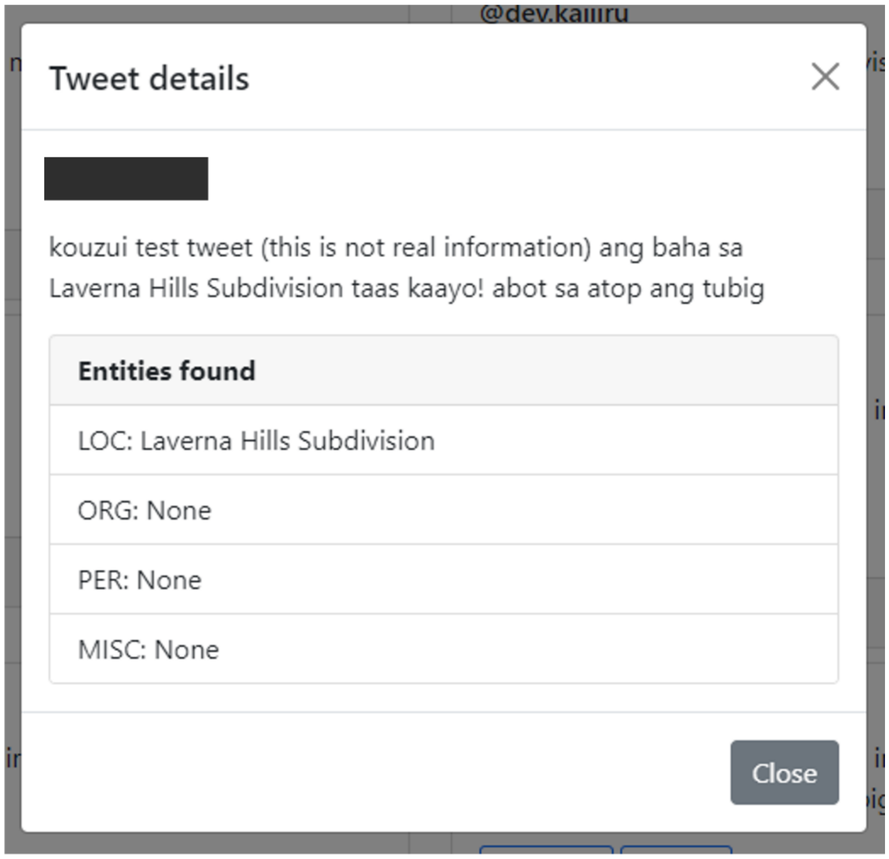

[redacted]

kouzui test tweet (this is not real information) ang baha sa Laverna Hills Subdivision taas kaayo! abot sa atop ang tubig

Oct. 29, 2021, 10:18 p.m.

Figure 5-B: Snippet of a tweet made from the researcher's Twitter account on the developed website

The developed tool by the researchers will run a script that will fetch tweets from Twitter every 10 minutes then save them in the database then display them on the web interface. If the developed tool detects a location entity, a map button will appear as seen in figure 5-B.



Tweet details

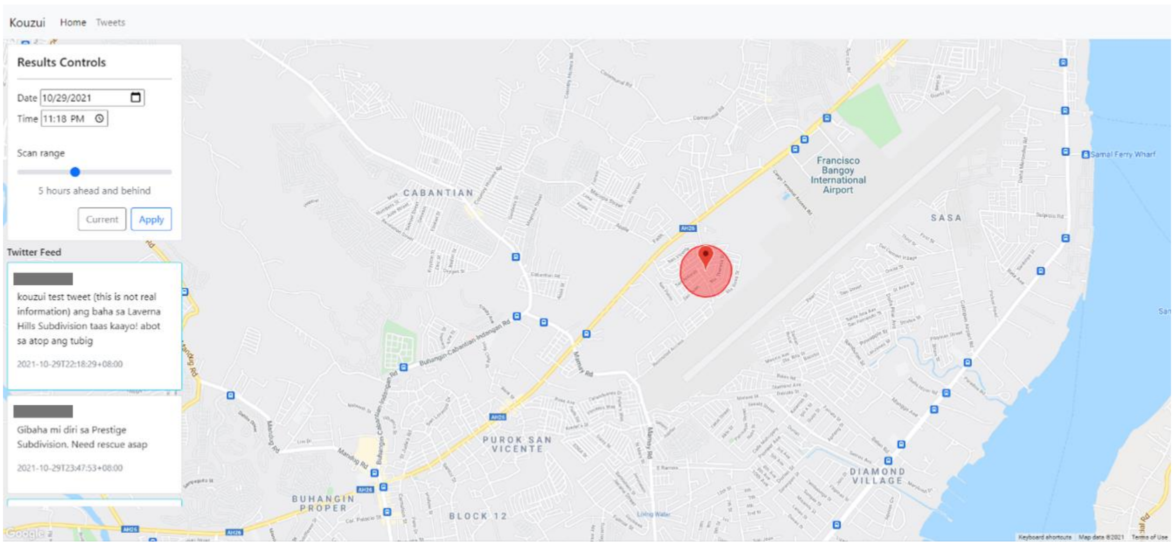
[redacted]

kouzui test tweet (this is not real information) ang baha sa Laverna Hills Subdivision taas kaayo! abot sa atop ang tubig

Entities found
LOC: Laverna Hills Subdivision
ORG: None
PER: None
MISC: None

Figure 5-C: List of entities detected from a tweet

The details button in figure 5-B will open a modal that will display the entities that were found by the named entity recognizer. There are 4 entities in which the researchers are interested so they will be displayed on the web interface for easy access.



Results Controls

Date 10/29/2021

Time 11:18 PM

Scan range

5 hours ahead and behind

Twitter Feed

[redacted]
kouzui test tweet (this is not real information) ang baha sa Laverna Hills Subdivision taas kaayo! abot sa atop ang tubig
2021-10-29T22:18:29+08:00

[redacted]
Gibaha mi diri sa Prestige Subdivision. Need rescue asap
2021-10-29T23:47:53+08:00

Figure 5-D: A location identified from the test tweet

If a location entity has been identified on the tweet, the coordinates will be queried by using the Places API, an API still under the Google Maps API. Once the coordinates are received, the Google Maps API will display a shaded circle on the specified location. This is under the assumption that the tweet only has one location mentioned. The smaller the circle, the more specific the place is. The specificity of the location is identified by the Places API, which is as follows: Rooftop, Range Interpolated, Geometric Center, and Approximate, with Rooftop being the most specific and the approximate being the broadest.

In order to determine the basis on which a tweet is considered relevant during flooding disasters, the researchers will look into the results of the data from the gathered tweets from October 25, 2021, where heavy rain has struck Davao City and caused floods in different parts within the city. The researchers have collected 1000 datasets from Twitter that day and then used the system to determine the location, and then tallied the results.

For testing the reliability of the developed system, the researchers have devised 12 strings of texts; 4 texts with the languages of English, Tagalog, and Bisaya, with features from figure 4.5-A, then work by removing features to experiment with the combination of features seen on figure 4.5-B.

Text ID	English Text	Tagalog Text	Bisaya Text	Location
1	News flash: A flood is currently happening in 98 Roxas Avenue, Poblacion District, Davao City.	News Flash: Sa kasalukuyan, isang baha ang nangyayari sa 98 Roxas Avenue,	Adunay baha nga nagapanghitabo sa may 98 Roxas Avenue, Poblacion District, Davao City	98 Roxas Avenue, Poblacion District, Davao City

		Poblacion District, Davao City		
2	People are asking for rescue efforts in 147 San Mateo, Buhangin, Davao City. There are flooded areas over there!	Ang mga tao mula sa 147 San Mateo, Buhangin, Davao City ay nasalanta ng baha at nanghihingi ng tulong sa mga rescuer upang sila ay maligtas mula sa panganib dulot ng baha.	Nanginahanglan mi ug tabang. Magparescue mi diri sa 147 San Mateo, Buhangin, Davao City.	147 San Mateo, Buhangin, Davao City
3	Flood is happening! The waters in 109, Lunar Street, Matina Crossing, Talomo, Davao City are getting higher.	Tumataas ang tubig dulot ng baha sa may 109, Lunar Street, Matina Crossing, Talomo, Davao City.	Ang tubig sa may 109 Lunar Street, Matina Crossing, Talomo, Davao City, taga tuhod na.	109, Lunar Street, Matina Crossing, Talomo, Davao City
4	Flood water in 105 Jacinto Street, Poblacion District, Davao City is high.	Abot bewang na baha ang kasalukuyang nananalanta sa 105 Jacinto Street, Poblacion District, Davao City.	Ang tubig baha sa 105 Jacinto Street, Poblacion District, Davao City, taas na.	105 Jacinto Street, Poblacion District, Davao City

Figure 5-D: Table of simulated texts to be analyzed by the researchers

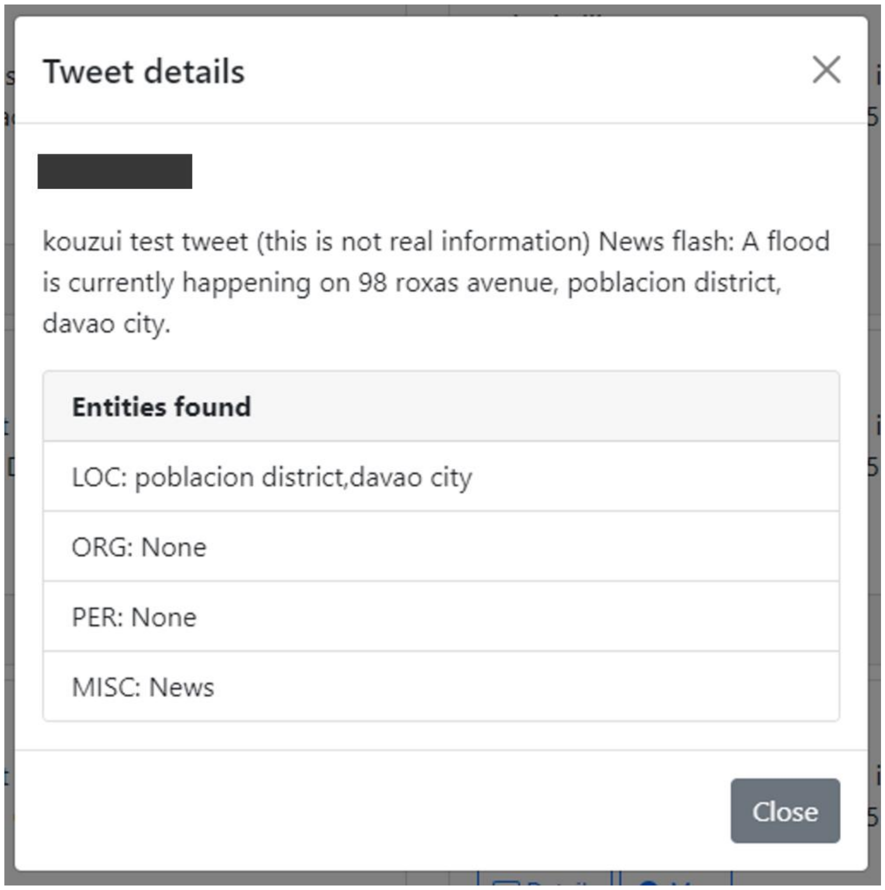
		English			**Tagalog**			**Bisaya**		
Text ID	**Combination ID**	**C**	**P**	**I**	**C**	**P**	**I**	**C**	**P**	**I**
1	1		✓			✓			✓	
1	2	✓			✓			✓		

1	3		✓			✓			✓	
1	4	✓			✓			✓		
1	5	✓				✓			✓	
2	1		✓			✓			✓	
2	2	✓				✓		✓		
2	3		✓			✓			✓	
2	4	✓				✓		✓		
2	5		✓			✓			✓	
3	1		✓			✓			✓	
3	2	✓			✓				✓	
3	3		✓			✓			✓	
3	4	✓			✓				✓	
3	5			✓		✓			✓	
4	1		✓			✓			✓	
4	2	✓		✓	✓			✓		
4	3		✓			✓			✓	

4	4	✓			✓			✓		
4	5		✓			✓			✓	
Total		9	10	1	6	14	0	6	14	0

Figure 5-E: Location entity recognition results from the simulated texts

In the table presented on figure 5-E, the researchers have analyzed the processed data by the tool developed if the entity recognizer has correctly tagged the location that was mentioned in the text. The texts in English have a significantly higher correct tagged location while the Tagalog and Bisaya tweets have the same scores.



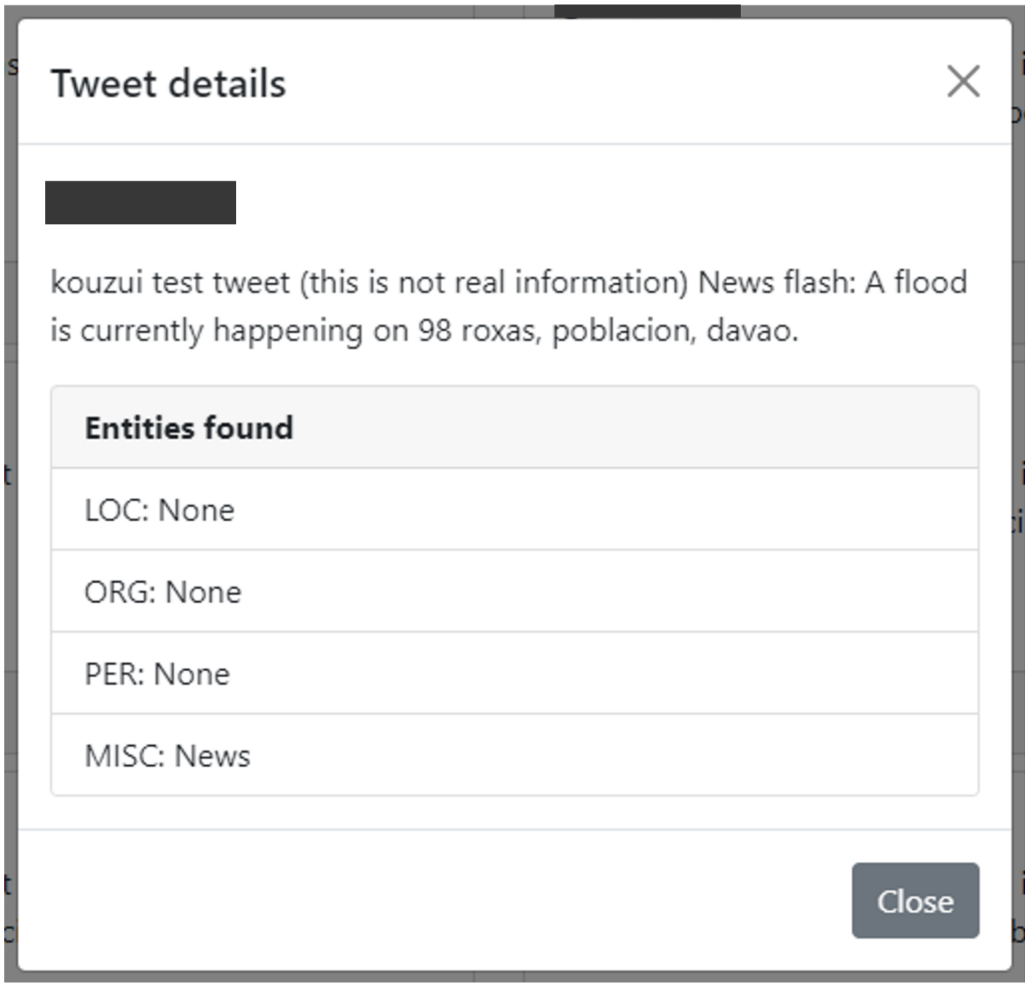
Tweet details

[redacted]

kouzui test tweet (this is not real information) News flash: A flood is currently happening on 98 roxas avenue, poblacion district, davao city.

Entities found
LOC: poblacion district,davao city
ORG: None
PER: None
MISC: News

Figure 5-F-1: A sample tweet without location signifier


Tweet details

[redacted]

kouzui test tweet (this is not real information) News flash: A flood is currently happening on 98 roxas, poblacion, davao.

Entities found
LOC: None
ORG: None
PER: None
MISC: News

Figure 5-F-2: A sample tweet with location signifier.

The researchers define the relevance of the tweets in this study as tweets that pertain to flooding that is happening that has information that is supplemental to determine where a flood is happening, in particular a location. To be able to tag a tweet relevant, a tweet must have the keyword "Baha" or "Flood" used in a context that pertains to a flooding natural disaster.. The tweet must have a location signifier such as "district", "city", "street", "municipality", etc. in order to properly tag an entity. As seen on figure 5-F-1, the named entity recognition system has successfully extracted a partial location while on the figure 5-F-2, no named entities were found.

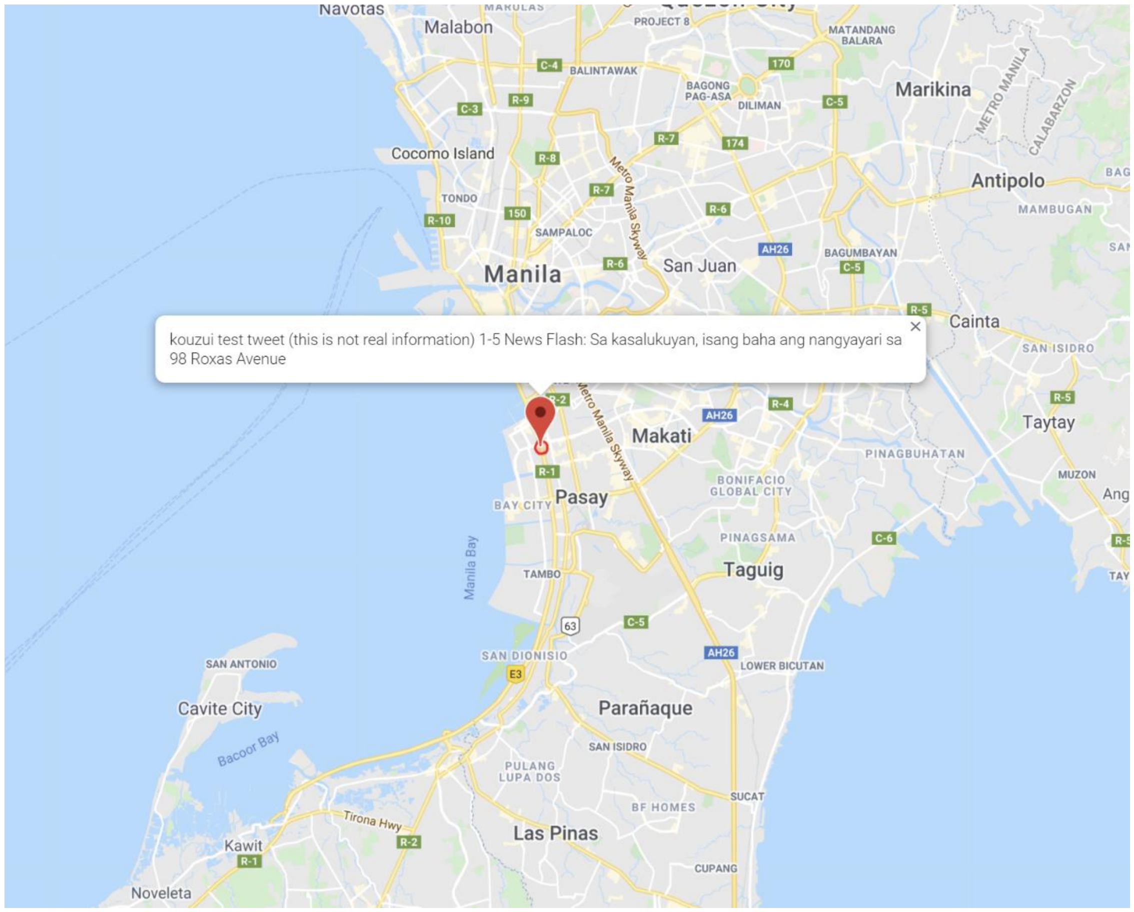

kouzui test tweet (this is not real information) 1-5 News Flash: Sa kasalukuyan, isang baha ang nangyayari sa 98 Roxas Avenue

Figure 5-G: Interpreted location from a house number and street name

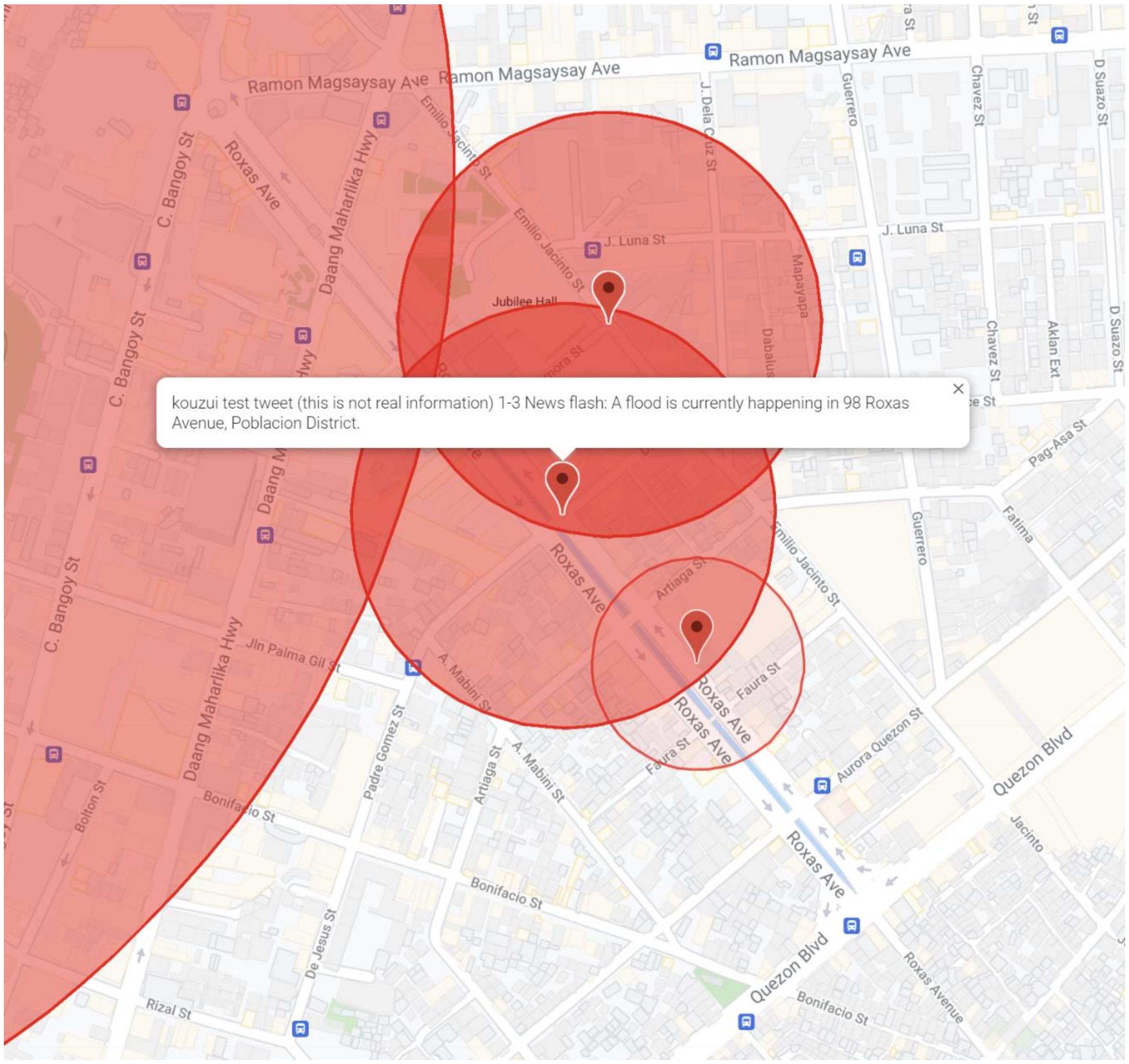
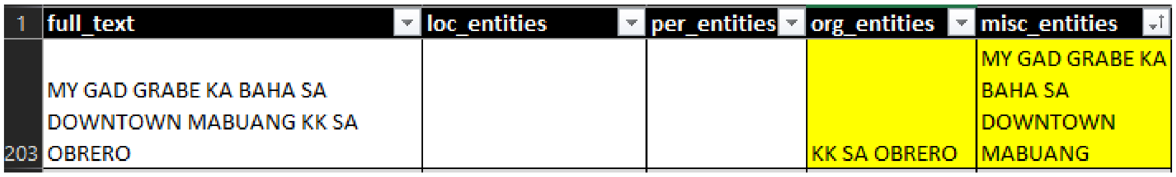

kouzui test tweet (this is not real information) 1-3 News flash: A flood is currently happening in 98 Roxas Avenue, Poblacion District.

Figure 5-H: Interpreted location from a house number, street name, and district name

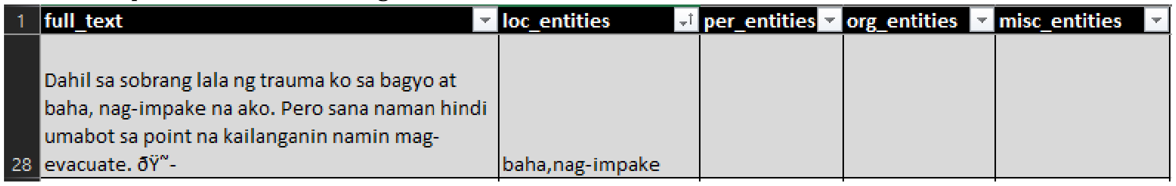
In terms of geo-locating a tweet by geocoding a mentioned location, the developed geo-visualization tool by the researchers works correctly if the mentioned location from a tweet is very specific. In figure 5-G, the tweet with a location of "98 Roxas Avenue" was interpreted and geo-visualized as a location in Manila whereas in figure 5-H, the tweet with a location of "98 Roxas Avenue, Poblacion District, was pinned within Davao City.



1	full_text	loc_entities	per_entities	org_entities	misc_entities
203	MY GAD GRABE KA BAHA SA DOWNTOWN MABUANG KK SA OBRERO			KK SA OBRERO	MY GAD GRABE KA BAHA SA DOWNTOWN MABUANG

Figure 5-I: Named Entity Recognition results from a tweet

In the analysis of the most mentioned locations within Davao City, the named entity recognition system implemented by the researchers for finding locations within the tweets has not properly tagged some of the locations. An example found in figure 5-I have not tagged a location as the loc_entities column is blank whereas a location has been partially tagged correctly as an organization entity and miscellaneous entity. The researchers point out the possibilities that the named entity recognizer did not work correctly when a tweet is in all-capitalized form, and the word "Obrero" is a vague entity as it does not have a location identifier such as "street" of "city", and the word "downtown" is a colloquial name for a place which can be vague.



1	full_text	loc_entities	per_entities	org_entities	misc_entities
28	Dahil sa sobrang lala ng trauma ko sa bagyo at baha, nag-impake na ako. Pero sana naman hindi umabot sa point na kailanganin namin mag-evacuate. ðŸ˜-	baha,nag-impake			

Figure 5-J: Wrong tag from the Named Entity Recognition system

The named entity recognition has however tagged "Davao" correctly as a location entity in tweets that mentions "Davao" however the named entity recognition system has also tagged some words as a location entity even though it is not a location. As seen in figure 5-J, "*baha, nag-impake*" has been tagged as a location even though it is not a location.

10 CONCLUSION AND RECOMMENDATION

10.1 Conclusion

From the findings of the study, the researchers have concluded that the more a retrieved tweet is specific to its location, the more reliable the geo-visualization will be, but the assumption should be that the tweet only mentions one location. Test tweets created by the researchers have helped confirm this, displaying a specific location to a street-level point of view. These locations are highlighted within a red circular outline, which intensifies in its hue the more tweets are originating from said location. However, these are rare occurrences, as addresses are rarely given out within public tweets, so the researchers have encountered less specific tweets, which result in relatively inaccurate and broad estimations of their geo-visualized location. The named entity recognition system by the researchers has not properly worked yet with Filipino tweets as there are no specific training models yet for Filipino texts that are available for Spacy NER.

10.2 Recommendation

Based on the findings of this paper, the researchers recommend that future researchers should experiment with different forms of NER. If a Filipino training model, a NER specialized in Filipino words, locations, and occupations, is released, then the researchers suggest using that as a basis. It is also possible that future researchers could train their own form of NER via machine learning. It is also recommended that future researchers take into account open-source methods to either bolster or hasten their creation of a NER.

This paper also recommends that researchers look for alternative mapping API should Google Maps prove to be unwieldy for their purposes, like OpenStreetMaps. Also, in regards to discerning the authenticity of gathered tweets, it is recommended that future researchers look into other forms of discernment. One such suggestion is through the detection of hashtags on gathered tweets and whether or not a majority of them share that hashtag. Though, this falls secondary to a properly trained NER.

11 REFERENCES

Cadag, J. R., Petal, M., Luna, E., Gaillard, J. C., Pambid, L., & Santos, G. (2017). Hidden disasters: Recurrent flooding impacts on educational continuity in the Philippines. International Journal of Disaster Risk Reduction. https://doi.org/10.1016/j.ijdrr.2017.07.016

Mooney, S. J., Westreich, D. J., & El-Sayed, A. M. (2015). Epidemiology in the era of big data. Epidemiology, 26(3), 390.

Tan, A. (1999). Text Mining: The state of the art and the challenges. In Proceedings of the PAKDD 1999 Workshop on Knowledge Discovery from Advanced Databases, 65–70.

What Is Text Mining? A Beginner's Guide. (n.d.). MonkeyLearn. Retrieved April 20, 2021, from https://monkeylearn.com/text-mining/

What is Text Mining? (2021, January 26). IBM. https://www.ibm.com/cloud/learn/text-mining

Varun, R. (n.d.). Text Mining Vs Text Analytics | ExcelR. Excelr.Com. Retrieved April 25, 2021, from https://www.excelr.com/text-mining-vs-text-analytics

Salloum, S., Al-Emran, M., & Shaalan, K. (2017). Mining Social Media Text: Extracting Knowledge from Facebook. International Journal of Computing and Digital Systems, 6(2), 73–81. https://doi.org/10.12785/IJCDS/060203

Automated Data Collection Terms. (2010, April 15). Facebook. https://www.facebook.com/apps/site_scraping_tos_terms.php

Pascual, F. (2019, November 20). What Is Text Analytics? MonkeyLearn Blog. https://monkeylearn.com/blog/what-is-text-analytics/

Storie, J. (2016). What is Geovisualization? 6.

Torget, A. J., Mihalcea, R., Christensen, J., & McGhee, G. (2011). MAPPING TEXTS: COMBINING TEXT-MINING AND GEO-VISUALIZATION TO UNLOCK THE RESEARCH POTENTIAL OF HISTORICAL NEWSPAPERS. 53.

Varun, R. (n.d.). Text Mining Vs Text Analytics | ExcelR. Excelr.Com. Retrieved April 25, 2021, from https://www.excelr.com/text-mining-vs-text-analytics

Sutton, S., & Swickard, K. (2020). Text Mining 101. The Serials Librarian, 78(1-4), 3-8.

Bird, S., Klein, E., & Loper, E. (2009). Natural language processing with Python: analyzing text with the natural language toolkit. " O'Reilly Media, Inc.".

Django introduction—Learn web development | MDN. (n.d.). Retrieved May 27, 2021, from https://developer.mozilla.org/en-US/docs/Learn/Server-side/Django/Introduction

Ibrahim, O. A., & Mohsen, K. J. (2014). Design and implementation an online location based services using Google maps for android mobile. International Journal of Computer Networks and Communications Security (CNCS), 2(3), 113-118.

Ablazo, A. (2019). Designing a Web Application using the Twitter streaming API as a Tracking Tool for Weather Updates [Master's thesis]. Ateneo de Davao University.

Overview | Elevation API. (n.d.). Google Developers. Retrieved June 5, 2021, from https://developers.google.com/maps/documentation/elevation/overview

Khan, S. (2021). Google maps heatmap- How to visualize location history | VWO. Blog. https://vwo.com/blog/google-maps-heatmap/

Heatmap layer. (n.d.). Google Developers. https://developers.google.com/maps/documentation/javascript/heatmaplayer

What is bootstrap: A beginners guide. (2016). Become a designer, developer or data analyst from scratch. https://careerfoundry.com/en/blog/web-development/what-is-bootstrap-a-beginners-guide/

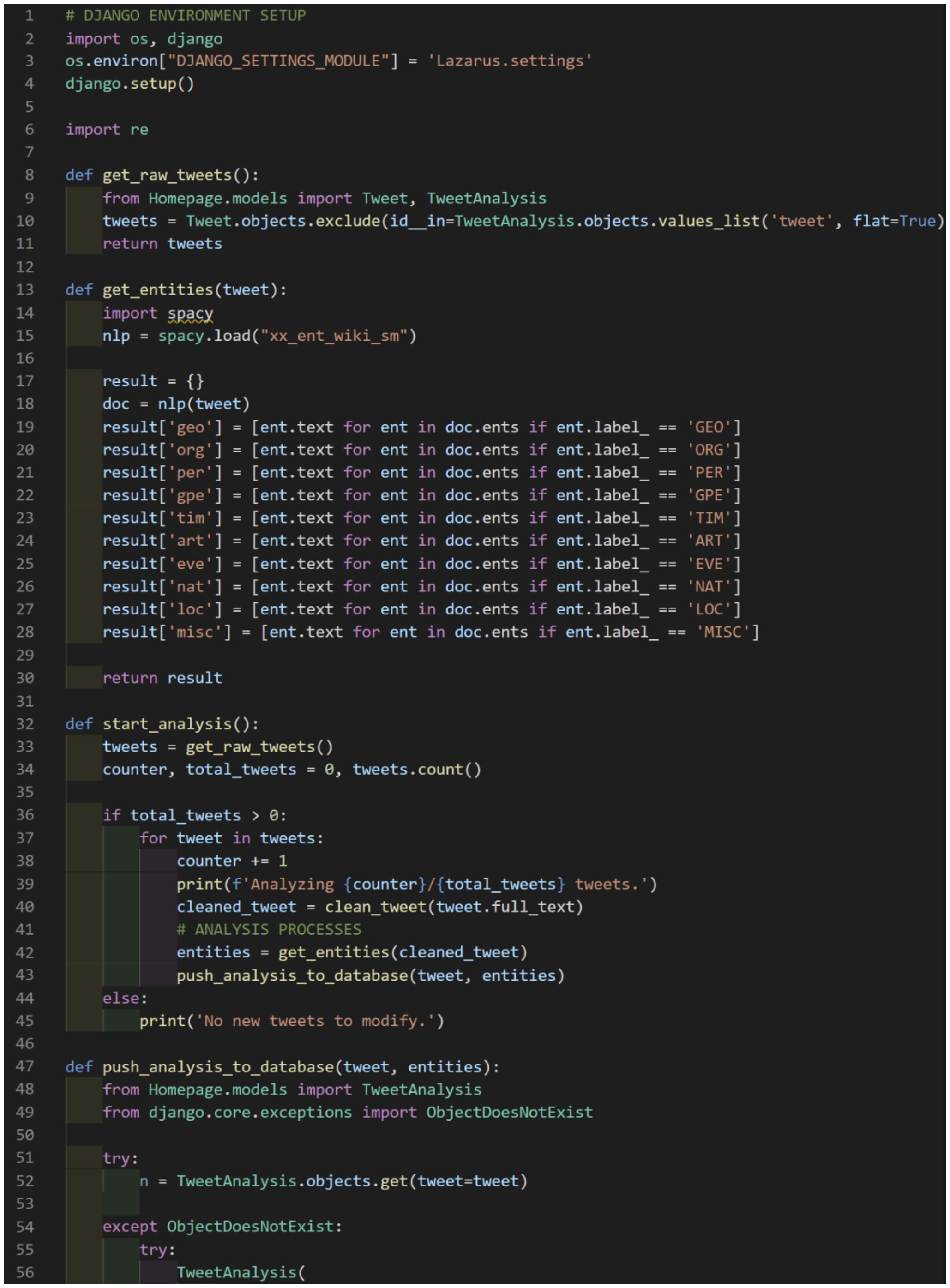
Mansouri, A., Affendey, L. S., & Mamat, A. (2008). Named entity recognition approaches. *International Journal of Computer Science and Network Security*, *8*(2), 339-344.

Lim, L. E., New, J. C., Ngo, M. A., Sy, M., & Lim, N. R. (2007). A Named-Entity Recognizer for Filipino Texts. *Proceedings of the 4th NNLPRS*.

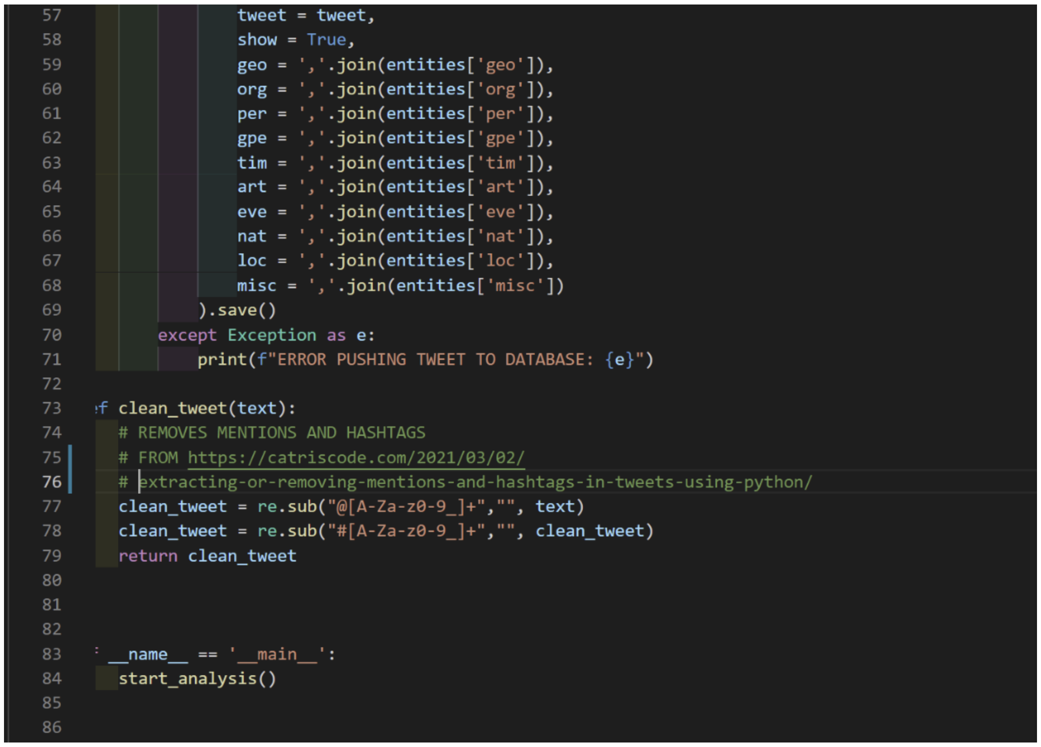
Spacy 101: Everything you need to know · spacy usage documentation. (n.d.). Spacy. https://spacy.io/usage/spacy-101

12 APPENDIX

12.1 TweetAnalysis.py

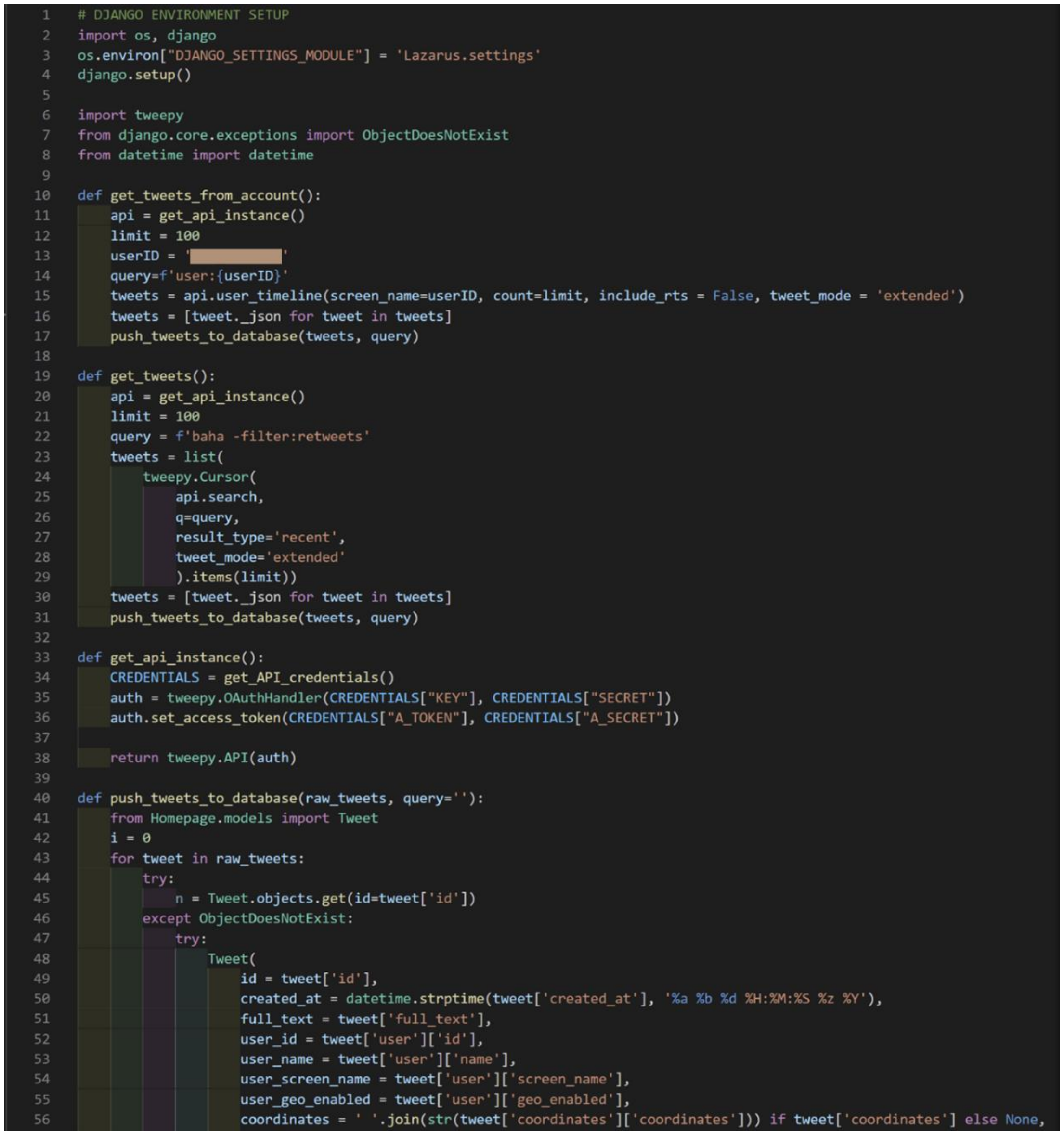


```python
# DJANGO ENVIRONMENT SETUP
import os, django
os.environ["DJANGO_SETTINGS_MODULE"] = 'Lazarus.settings'
django.setup()

import re

def get_raw_tweets():
    from Homepage.models import Tweet, TweetAnalysis
    tweets = Tweet.objects.exclude(id__in=TweetAnalysis.objects.values_list('tweet', flat=True)
    return tweets

def get_entities(tweet):
    import spacy
    nlp = spacy.load("xx_ent_wiki_sm")

    result = {}
    doc = nlp(tweet)
    result['geo'] = [ent.text for ent in doc.ents if ent.label_ == 'GEO']
    result['org'] = [ent.text for ent in doc.ents if ent.label_ == 'ORG']
    result['per'] = [ent.text for ent in doc.ents if ent.label_ == 'PER']
    result['gpe'] = [ent.text for ent in doc.ents if ent.label_ == 'GPE']
    result['tim'] = [ent.text for ent in doc.ents if ent.label_ == 'TIM']
    result['art'] = [ent.text for ent in doc.ents if ent.label_ == 'ART']
    result['eve'] = [ent.text for ent in doc.ents if ent.label_ == 'EVE']
    result['nat'] = [ent.text for ent in doc.ents if ent.label_ == 'NAT']
    result['loc'] = [ent.text for ent in doc.ents if ent.label_ == 'LOC']
    result['misc'] = [ent.text for ent in doc.ents if ent.label_ == 'MISC']

    return result

def start_analysis():
    tweets = get_raw_tweets()
    counter, total_tweets = 0, tweets.count()

    if total_tweets > 0:
        for tweet in tweets:
            counter += 1
            print(f'Analyzing {counter}/{total_tweets} tweets.')
            cleaned_tweet = clean_tweet(tweet.full_text)
            # ANALYSIS PROCESSES
            entities = get_entities(cleaned_tweet)
            push_analysis_to_database(tweet, entities)
    else:
        print('No new tweets to modify.')

def push_analysis_to_database(tweet, entities):
    from Homepage.models import TweetAnalysis
    from django.core.exceptions import ObjectDoesNotExist

    try:
        n = TweetAnalysis.objects.get(tweet=tweet)

    except ObjectDoesNotExist:
        try:
            TweetAnalysis(
```

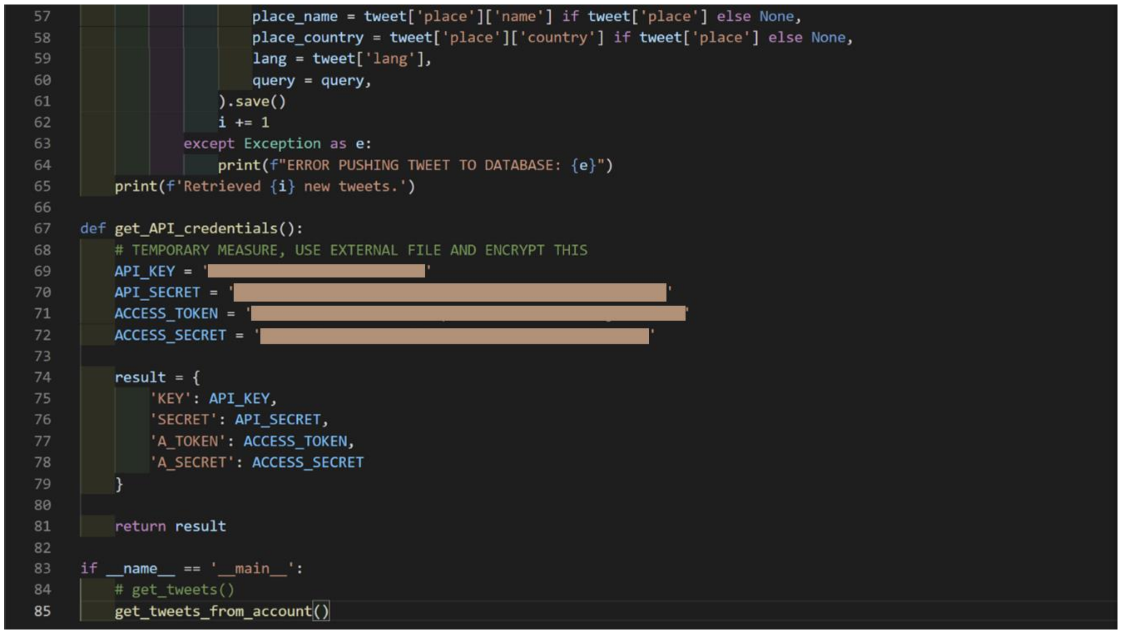


```
                tweet = tweet,
                show = True,
                geo = ','.join(entities['geo']),
                org = ','.join(entities['org']),
                per = ','.join(entities['per']),
                gpe = ','.join(entities['gpe']),
                tim = ','.join(entities['tim']),
                art = ','.join(entities['art']),
                eve = ','.join(entities['eve']),
                nat = ','.join(entities['nat']),
                loc = ','.join(entities['loc']),
                misc = ','.join(entities['misc'])
            ).save()
        except Exception as e:
            print(f"ERROR PUSHING TWEET TO DATABASE: {e}")

f clean_tweet(text):
  # REMOVES MENTIONS AND HASHTAGS
  # FROM https://catriscode.com/2021/03/02/
  # extracting-or-removing-mentions-and-hashtags-in-tweets-using-python/
  clean_tweet = re.sub("@[A-Za-z0-9_]+","", text)
  clean_tweet = re.sub("#[A-Za-z0-9_]+","", clean_tweet)
  return clean_tweet



 __name__ == '__main__':
  start_analysis()


```

12.2 TweetRetriever.py

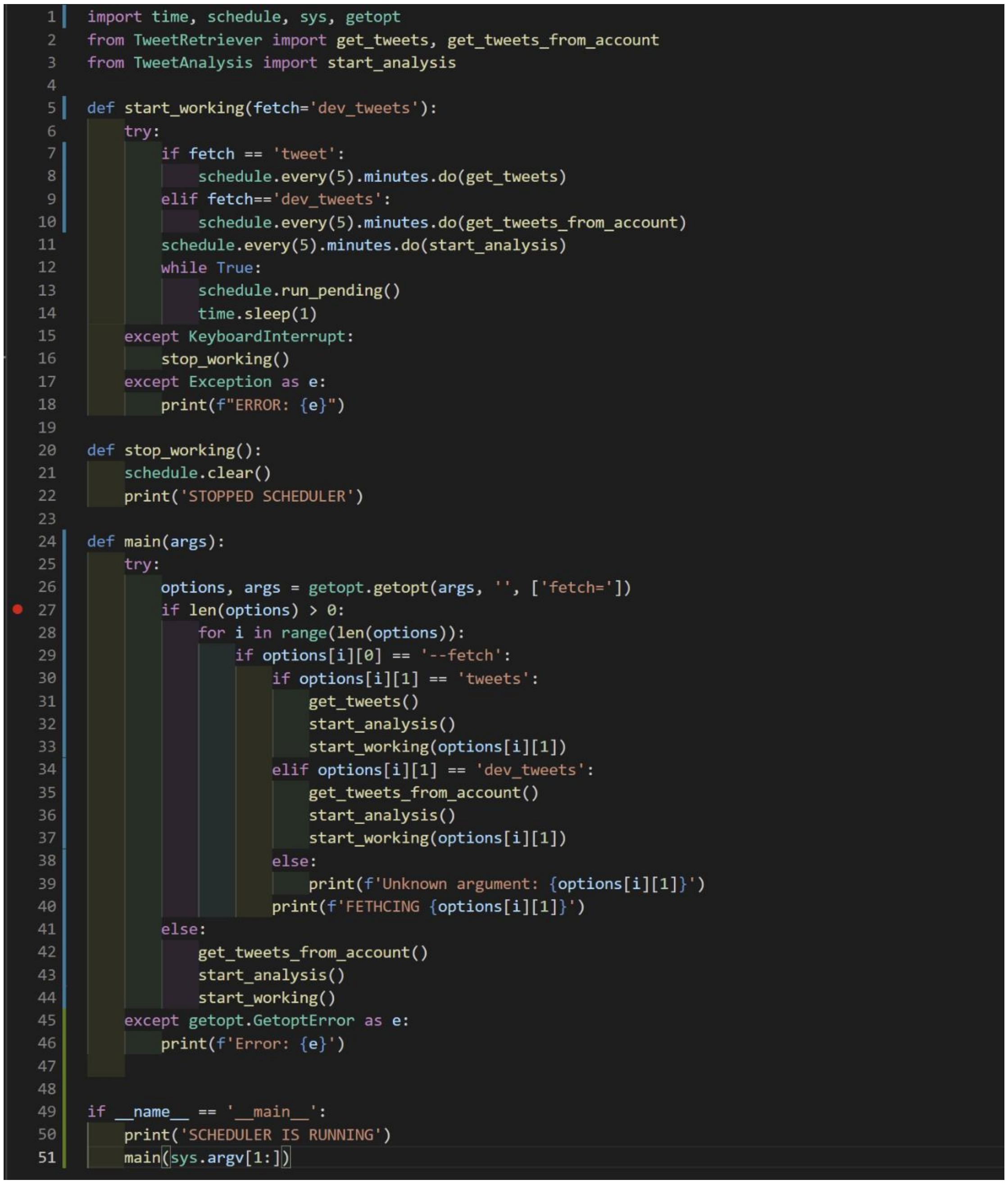


```python
# DJANGO ENVIRONMENT SETUP
import os, django
os.environ["DJANGO_SETTINGS_MODULE"] = 'Lazarus.settings'
django.setup()

import tweepy
from django.core.exceptions import ObjectDoesNotExist
from datetime import datetime

def get_tweets_from_account():
    api = get_api_instance()
    limit = 100
    userID = '[redacted]'
    query=f'user:{userID}'
    tweets = api.user_timeline(screen_name=userID, count=limit, include_rts = False, tweet_mode = 'extended')
    tweets = [tweet._json for tweet in tweets]
    push_tweets_to_database(tweets, query)

def get_tweets():
    api = get_api_instance()
    limit = 100
    query = f'baha -filter:retweets'
    tweets = list(
        tweepy.Cursor(
            api.search,
            q=query,
            result_type='recent',
            tweet_mode='extended'
            ).items(limit))
    tweets = [tweet._json for tweet in tweets]
    push_tweets_to_database(tweets, query)

def get_api_instance():
    CREDENTIALS = get_API_credentials()
    auth = tweepy.OAuthHandler(CREDENTIALS["KEY"], CREDENTIALS["SECRET"])
    auth.set_access_token(CREDENTIALS["A_TOKEN"], CREDENTIALS["A_SECRET"])

    return tweepy.API(auth)

def push_tweets_to_database(raw_tweets, query=''):
    from Homepage.models import Tweet
    i = 0
    for tweet in raw_tweets:
        try:
            n = Tweet.objects.get(id=tweet['id'])
        except ObjectDoesNotExist:
            try:
                Tweet(
                    id = tweet['id'],
                    created_at = datetime.strptime(tweet['created_at'], '%a %b %d %H:%M:%S %z %Y'),
                    full_text = tweet['full_text'],
                    user_id = tweet['user']['id'],
                    user_name = tweet['user']['name'],
                    user_screen_name = tweet['user']['screen_name'],
                    user_geo_enabled = tweet['user']['geo_enabled'],
                    coordinates = ' '.join(str(tweet['coordinates']['coordinates'])) if tweet['coordinates'] else None,
```



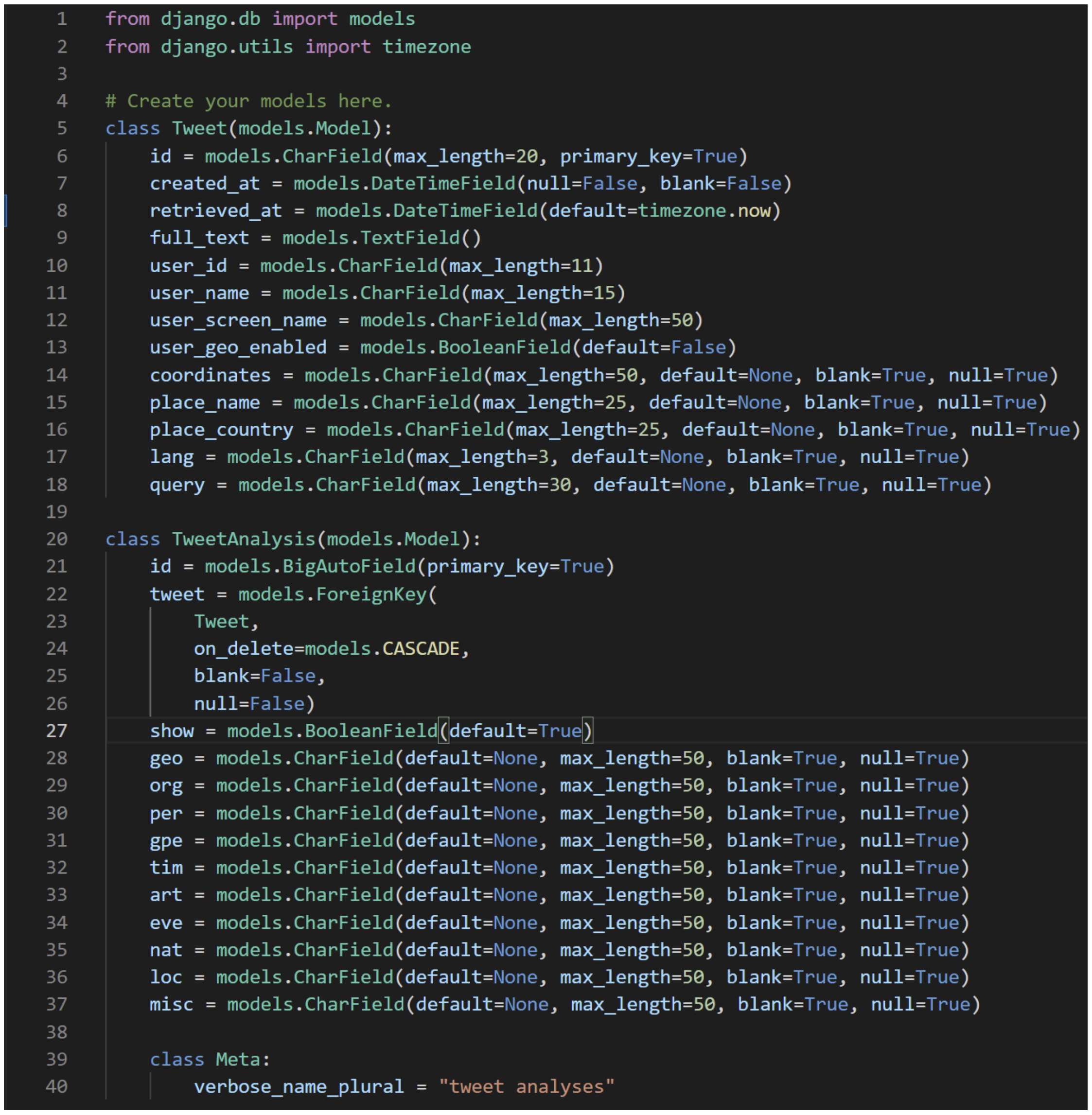
```python
                    place_name = tweet['place']['name'] if tweet['place'] else None,
                    place_country = tweet['place']['country'] if tweet['place'] else None,
                    lang = tweet['lang'],
                    query = query,
                ).save()
                i += 1
            except Exception as e:
                print(f"ERROR PUSHING TWEET TO DATABASE: {e}")
    print(f'Retrieved {i} new tweets.')

def get_API_credentials():
    # TEMPORARY MEASURE, USE EXTERNAL FILE AND ENCRYPT THIS
    API_KEY = '[redacted]'
    API_SECRET = '[redacted]'
    ACCESS_TOKEN = '[redacted]'
    ACCESS_SECRET = '[redacted]'

    result = {
        'KEY': API_KEY,
        'SECRET': API_SECRET,
        'A_TOKEN': ACCESS_TOKEN,
        'A_SECRET': ACCESS_SECRET
    }

    return result

if __name__ == '__main__':
    # get_tweets()
    get_tweets_from_account()
```

12.3 Jobs.py



```python
import time, schedule, sys, getopt
from TweetRetriever import get_tweets, get_tweets_from_account
from TweetAnalysis import start_analysis

def start_working(fetch='dev_tweets'):
    try:
        if fetch == 'tweet':
            schedule.every(5).minutes.do(get_tweets)
        elif fetch=='dev_tweets':
            schedule.every(5).minutes.do(get_tweets_from_account)
        schedule.every(5).minutes.do(start_analysis)
        while True:
            schedule.run_pending()
            time.sleep(1)
    except KeyboardInterrupt:
        stop_working()
    except Exception as e:
        print(f"ERROR: {e}")

def stop_working():
    schedule.clear()
    print('STOPPED SCHEDULER')

def main(args):
    try:
        options, args = getopt.getopt(args, '', ['fetch='])
        if len(options) > 0:
            for i in range(len(options)):
                if options[i][0] == '--fetch':
                    if options[i][1] == 'tweets':
                        get_tweets()
                        start_analysis()
                        start_working(options[i][1])
                    elif options[i][1] == 'dev_tweets':
                        get_tweets_from_account()
                        start_analysis()
                        start_working(options[i][1])
                    else:
                        print(f'Unknown argument: {options[i][1]}')
                    print(f'FETHCING {options[i][1]}')
        else:
            get_tweets_from_account()
            start_analysis()
            start_working()
    except getopt.GetoptError as e:
        print(f'Error: {e}')


if __name__ == '__main__':
    print('SCHEDULER IS RUNNING')
    main(sys.argv[1:])
```

12.4 Homepage/models.py



```python
from django.db import models
from django.utils import timezone

# Create your models here.
class Tweet(models.Model):
    id = models.CharField(max_length=20, primary_key=True)
    created_at = models.DateTimeField(null=False, blank=False)
    retrieved_at = models.DateTimeField(default=timezone.now)
    full_text = models.TextField()
    user_id = models.CharField(max_length=11)
    user_name = models.CharField(max_length=15)
    user_screen_name = models.CharField(max_length=50)
    user_geo_enabled = models.BooleanField(default=False)
    coordinates = models.CharField(max_length=50, default=None, blank=True, null=True)
    place_name = models.CharField(max_length=25, default=None, blank=True, null=True)
    place_country = models.CharField(max_length=25, default=None, blank=True, null=True)
    lang = models.CharField(max_length=3, default=None, blank=True, null=True)
    query = models.CharField(max_length=30, default=None, blank=True, null=True)

class TweetAnalysis(models.Model):
    id = models.BigAutoField(primary_key=True)
    tweet = models.ForeignKey(
        Tweet,
        on_delete=models.CASCADE,
        blank=False,
        null=False)
    show = models.BooleanField(default=True)
    geo = models.CharField(default=None, max_length=50, blank=True, null=True)
    org = models.CharField(default=None, max_length=50, blank=True, null=True)
    per = models.CharField(default=None, max_length=50, blank=True, null=True)
    gpe = models.CharField(default=None, max_length=50, blank=True, null=True)
    tim = models.CharField(default=None, max_length=50, blank=True, null=True)
    art = models.CharField(default=None, max_length=50, blank=True, null=True)
    eve = models.CharField(default=None, max_length=50, blank=True, null=True)
    nat = models.CharField(default=None, max_length=50, blank=True, null=True)
    loc = models.CharField(default=None, max_length=50, blank=True, null=True)
    misc = models.CharField(default=None, max_length=50, blank=True, null=True)

    class Meta:
        verbose_name_plural = "tweet analyses"
```