Metrohelper: A Real-time Web-based System for Metro Incident Detection Using Social Media

Chih Fang Chen

Thesis submitted to the Faculty of the

Virginia Polytechnic Institute and State University

in partial fulfillment of the requirements for the degree of

Master of Science

in

Computer Science & Applications

Chang Tien Lu, Chair

Jin Hee Cho

Chandan K. Reddy

May 4, 2022

Falls Church, Virginia

Keywords: Information Retrieval, Twitter, Social Media, Data Mining

Metrohelper: A Real-time Web-based System for Metro Incident Detection Using Social Media

Chih Fang Chen

(ABSTRACT)

In recent years the usage of public transit services has been rapidly increased, thanks to huge progress on network technologies. However, the disruptions in modern public transit services also increased, due to aging infrastructure, non-comprehensive system design and the needs for maintenance. Any disruptions happened in current transit networks can cause to major disasters on passengers who use these networks for their daily commutes. Although we have lots of usage on transit network, still most current disruptions detection systems either lack of network coverage or did not have real-time system. The goal of this thesis was to create a system that can leverage Twitter data to help in detecting service disruptions in their early stage. This work involves a web applications which contains front-end, back-end and database, along with data mining techniques that obtain Tweets from a live Twitter stream related to the Washington Metropolitan Area Transit Authority (WMATA) metro system. The fundamental features of the system includes real-time incidents panel, historical events review, activities search near specific metro station and recent news review, which allowing people to have more relatively information based on their needs. After the initial functionalities is being settled, we further developed storytelling and sentiment analysis applications, which allowed people have more comprehensive information about the incidents that are happened around metro stations. Also, with the emergency report we developed, the developer can have immediate notification when an urgent event occurred. After fully

testified the system's case study on storytelling, sentiment analysis and emergency report, the outcomes are extreme convincing and trustworthy.

Metrohelper: A Real-time Web-based System for Metro Incident Detection Using Social Media

Chih Fang Chen

(GENERAL AUDIENCE ABSTRACT)

As public transit network become more and more accessible, people around the world rely on these network for their daily commutes. It is clearly that service disruptions among these system will affect passengers severely, especially when there are more and more people using it. This thesis is dedicated to build a web application that will not only allowing people to search latest information, but also assisting on the early detection of the disruptions. In this work we have developed an web application which has easy to use user interface, along with data mining techniques that connected with live data from Twitter to identify these disruptions. Our website is a real-time platform that contains real-time incidents panel, historical events review, activities search near specific metro station and recent news review based on latest tweets and news. By collecting live data from Twitter and various news website, we further developed storytelling and sentiment analysis features. For storytelling, we applied a machine learning model to help us clustering the related tweets/news, after summarize and track the evolution of tweets/news, we converted into stories and displayed it with interactive timelines. For sentiment analysis, we integrated a machine learning model which will scaled the emotional strength of tweets/news, then show the feelings of particular tweets/news. Additionally, we create an emergency report functionality, since it is important for the authority to where and when the incidents happened as soon as possible. The outcome of the system has been well-testify based on the daily case studies, and the results not only meet the ground truth, but also provide with various information.

Dedication

Dedicated to Chin Tsai Chang and Chien Hung Chen

Acknowledgments

Firstly, I would like to thank the CS department and Dr. Chang Tien Lu for his guidance, funding and supporting me as GRA through my Master's degree.

I would like to thank Kai Hsiang Cheng, Che Hsien Liao and Stephen Sun for their supports and companies through my Master's degree.

I would also like to thank Dr. Sattar Mansi for giving me the chance to work with him on Finance & Engineering project.

I would also like to thank Dr. Jin Hee Cho and Dr. Chandan K. Reddy to be my graduate committees.

Finally, I would like to thank my family and my close friends for their support. I wouldn't be where I am today without them.

Contents

List of Figures

List of Tables

List of Abbreviations

CNN Convolutional Neural Network

DBSCAN Density-Based Spatial Clustering of Applications with Noise

NLP Natural Language Processing

NLTK Natural Language Toolkit

VADER Valence Aware Dictionary for Sentiment Reasoning

WMATA Washington Metropolitan Area Transit Authority

Chapter 1

Introduction

1.1 Background

For all metropolitan cities, public transit networks have became more and more crucial throughout these years. These networks bring in huge amounts of daily users or customers, due to their accessibility and conveniences. More importantly, they connected the suburbs and outskirts of the city to the main metropolitan area. This make them the priority option for commuters and popular mode for travelers on their transportation. According to Washington DC Metropolitan Transit Authority report back in 2019, there were approximately 630,000 daily ridership [13]. Recent news also told that Washington DC Metropolitan Transit Authority is now having several changes for wooing back old riders and attracting new ones [1]. With the disruptions in service will not only forced old customers to find alternative for their transportation, but also hindered new customer to familiar with the system. This could eventually drive customers away, causing the negative reputations on the transit agency.

1.2 Motivation & Goal

1.2.1 Motivation

Technological advancements has lead to escalating use of social media applications in modern world. People can easily post and report nearby events and we can treat these as our data points. For social media platform, like Twitter, has 186 million daily active users with more than 500 million daily tweets in 2021. This make Twitter one of the most influenced social media platform, which became a major platform where topical information on various incidents can be found and extracted. In our work, we're focusing on three measures on Twitter:

1. Time

2. Location

3. Entity-related information about the incident.

By extracting the information from Twitter, we can take these data and use it for our advantage. Users on Twitter can be treated as human sensors. Figure 1.1 shows a sample of how these Twitter data can be aggregated with metro systems that provide disruptions based on location. With the increasing customers in ridership and users on Twitter, we can have a better coverage on incidents that happened near metro stations. Since modern reporting systems are relied on manual inspections and in-person complaints, this creates a gap between the actual time of the incident and time it was reported.

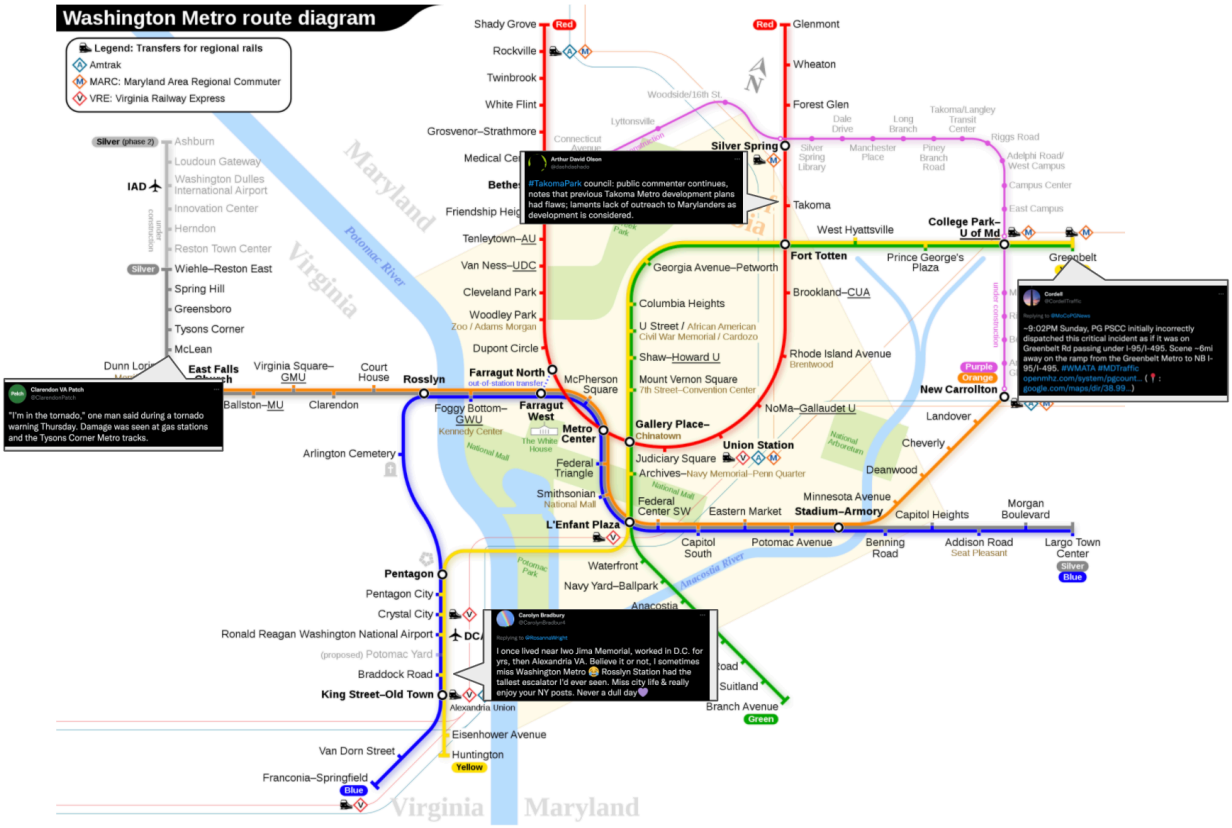

Figure 1.1: An Example of How Twitter Data Can Be Aggregated With Metro Systems and Provide Disruptions Based On Location
Source: https://en.wikipedia.org/wiki/Purple$_{L}ine_{(}Maryland$)

1.2.2 Goal

In order to solve this asynchronization among public transit networks, our approach is to develop a real-time system that adopt the use of Twitter data. By integrating incidents along with its geographical information into google map, we can have better understanding about the activities happened on particular area or station. To demonstrate the solution, we choose WMATA as our target. In this work, we utilized this approach to further create diverse features:

- **real-time Incidents Panel**: we construct a dataflow that will extract WMATA

related tweets from Twitter, displaying the latest 5 incidents on a real-time panel. This feature will be further discussed in subsection 3.2.

- **Storytelling**: we adapt DBSCAN that will clustered day/week tweets, converting into stories that give user to have a broad view among incidents. This feature will be further discussed in subsection 3.4.

- **Sentiment Analysis**: we use vader sentiment analysis tool from NLTK to help us create model that can differentiate people's feeling on particular event.This feature will be further discussed in subsection 3.5.

- **Emergency Report**: we provide emergency alerts to the authorities, once any potential threat is being found. This feature will be further discussed in subsection 3.6.

During our implementation, we found out that public news is also an important source of information, since most authorities announced their policy and decree through regional news. In other words, people who used public transit network can have detailed information on large scale events which driven us to develop features about news. For these features will be discussed in section 3, along with Twitter features.

At the end we provide additional case studies on historical events and real-time incidents caught by our system, which can further testified our system's validation.

1.3 Contributions

In this work, we developed a web application, for displaying and identifying metro disruptions from Twitter data. The major contributions of our work are:

1. **Social Media Mining**: Based on targeting public transit network, acquiring data from Twitter. Extracting disruption related Tweets using keyword detection. The details of the incidents is then stored and tracked.

2. **Incidents Clustering & Displayed**: Cluster a group of incidents which are related to each other on one single event. Convert the result into a story, allowing user to have better understanding for particular incident's timeline.

3. **Sentiment Analysis on Incident**: Adapt and implement algorithm to scaling person's feeling on particular incident, based on intensity of vocabulary he/she used on the post. Provide user a complementary view of events.

4. **Easy-To-Use Web Application for Public Transportation**: Create clean and comprehensive website, use it as an additional resource. With the real-time functionalities, people can easily catch up the latest event/news anywhere they go.

5. **Case Study**: Proof the validation of system by performing case studies among incidents clustering, storytelling and sentiment analysis to justify the effectiveness of our approach.

1.4 Thesis Structure

- **Chapter 1**: Provides a brief introduction on our motivation and goal and how our work can contribute to current public transit network.

- **Chapter 2**: Provides a literature review of past work done for algorithms, system design and database storage for live data. Also provides a review of work done for data clustering and data classification using well-trained non-GPU model and techniques being used to extract geographical information from Twitter stream.

- **Chapter 3**: Dives into the global view of how our system and database is being designed. This chapter will provide details and design for each of our website's features along with its data-flow and system design that applied during development.

- **Chapter 4**: Provides case studies that corresponding to real-time incidents and real-world events, and showing the overall outcome of our website.

- **Chapter 5**: Summarizes the results for our system study along with the future work that might help our system become more reliable for daily commuters.

Chapter 2

Review of Literature

There has been a lot of work related to real-time system and event detection from social media. This literature review has been divided into two sections, *System Design for Real-time Information* and *Database Storage for Real-time Dataset*.

2.1 System Design for Real-time Information

Some previous work on developing system for real-time information usage. Zulfiqar et al. [14] developed a system for detect security incidents within live Twitter stream using Dynamic Query Expansion which dedicated on tracking the incident's update during the period of time. Osborne et al. [9] construct the system called *ReDites*, which gives the update among crisis to its end-user. Gupta et al.[5] construct a web-based system for assessing credibility of content on Twitter for real-time events that used as a Chrome extension tool allowing user to have credibility score among Tweets. Chen et al.[2] Implementing a real-time Twitter-based system for resource dispatch in disaster management which helps authorities on decision making when catastrophe happened. There were also works which built the pipeline to utilizing real-time data to detect disease outbreaks and traffic incidents [4, 8, 12, 15].

Although previous studies have proofing their validation among their works, still there is room for us to try new design style on the system side. The goal of our work targeting on create a real-time platform that includes *real-time incidents detection*, *storytelling* and

emergency report that related to metro system disruptions. Also, most of the works focus on overall events, our work dedicated on aligning the information between citizens and authorities among metro incidents. Combining these unique features, make our work different from others.

2.2 Database Storage for Real-time Dataset

Database is critical for all work that related to processing real-time and gigantic dataset. However, most of work did not pay attention on the database their using. Gupta et al.[5] used *MySQL* as their database to store the result credibility score for Tweets. Chen et al.[2] used *SQL Server* as their geo-database to store geo-tagged data. Sharma et al.[11] also used *MySQL* as their database to store their Twitter recommendations. However, both MySQL and SQL Server are not designed to serve as a real-time database which the performance will decrease when its being used to solve real-time problem. In this work, we adapt two real-time database *MongoDB* and *Firebase* which both of them were designed to serve real-time applications. Thus, the overall result on query time will outperform MySQL and SQL Server. Also, our work use MongoDB for most of our features which is related to near real-time computing and Firebase for real-time incident update.

Chapter 3

WMATA System

In order to deal with real-time incidents, it is important to construct a system. This chapter will go over each feature's details, starting from data acquisition, pipeline setup to result display, along with framework and tools that were used.

3.1 Global View of System

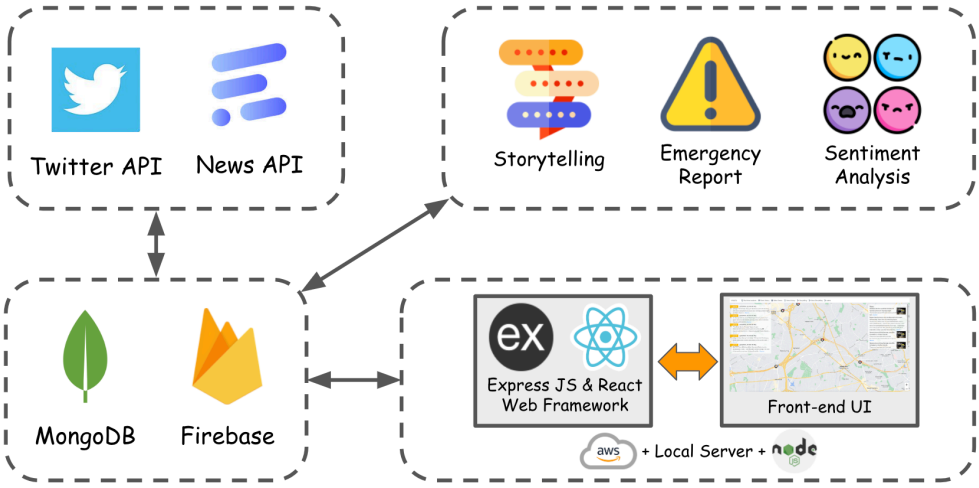

Figure 3.1: Global View of System

This section gives a global view of the structure among different portions, Figure 3.1 shows the overall view of the system; Data Acquisition, Data Storage, Daily Update and Web Application.

- **Data Acquisition**: Collecting real-time incidents from Twitter, Obtaining latest news from News API.
- **Data Storage**: Adopting MongoDB and Firebase as system's storage for different database purposes which will be further discussed in section 3.2.
- **Real-time & Daily Update**: Executing real-time and daily scripts like, storytelling, emergency report and sentiment analysis, not only to have the newest result on events, but also provide the latest analysis among incidents.
- **Web Application**: Setting up our user interface with Express JS and React JS as front-end. along with back-end, includes Amazon Web Services, Node JS and local server, etc.

In this work, sections like data acquisition, real-time & daily update and web application will exchange their information through data storage section which is treated as the intersection or hub in the system.

Also, In order to have a better management on our work, we used *Docker*, *Docker-compose* to separate the set up of front-end, back-end and other scripts. Docker and Docker-compose will treats different sections as a specific container which changes in different section will not effect other sections. This helps on turning the application into a microservice usage which allowing us to develop functionality independently.

3.2 Twitter Mining

Since this system is dealing with real-time events, we will be using the Python library called *Tweepy* which is Twitter Streaming API. Tweepy allows people to access real-time public

Tweets by getting needed information through settled query parameters. For different cities around the world, there are different hashtags, slogans and influential users. There are tags like *#wmata*, *#dcmetro*, and users like *@MetroHero Alerts*. We took these local tags to help us capture the related information for particular city.

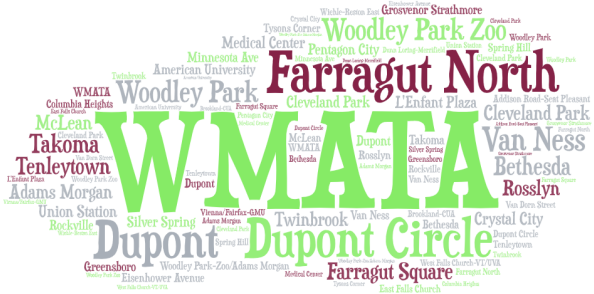

Figure 3.2: Words Used for Twitter Stream

In order to get relevant Tweets about WMATA from a generalized Twitter stream, a list of WMATA related keywords were passed to Tweepy streamer. The list contains variant name of stations and station localities along with their abbreviations. This step helps us to remove noise Tweets from the stream. Figure 3.2 shows part of words that being used in Twitter Stream.

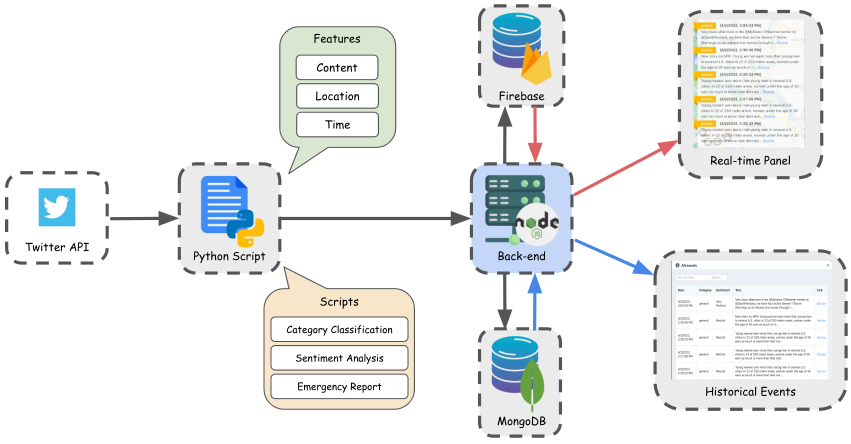

Figure 3.3: Data Pipeline of Twitter Stream

Figure 3.3 shows the entire pipeline of how data is being processed from Twitter stream.

When Tweet(s) first captured by Tweepy, they will go over a pre-processing script which extract needed features:

- Tweet Content
- Geographical Location
- Published Time

Since Tweets can contain emojis, special characters and symbols, by applied simple filter allowing us to have clean plain text of Tweet. Also, adopting remover to exclude any possible re-Tweets, cause most re-Tweets are duplicated due to their relations with original Tweet. However, the sentiment analysis tool we used have the ability to distinguished emojis, so the clean text will be stored in different database table. For more detail methodology will be presented in section 3.5.

For events in this work, we categorized them into three different type, *general*, *security* and *delay*. For events that are tagged *general* are information that related to WMATA but not related to actual incidents. For *security* and *delay* are the disruptions we were tried to bring up when they happened.

Twitter stream returns geo-tagged Tweets that contain *geo*, *place* and *coordinates* object. However, not all Tweets contain geo-location information means they are not geo-tagged. For Tweets that contain geo-location, we assigned them to the according stations. For Tweets that do not provide geographical information will be assigned to multiple stations based on locality in Tweet. Figure 3.4 shows how Tweet without geo-location is being assigned to stations.

Once all needed information is being extracted, they will be sent to two different databases, MongoDB and Firebase.

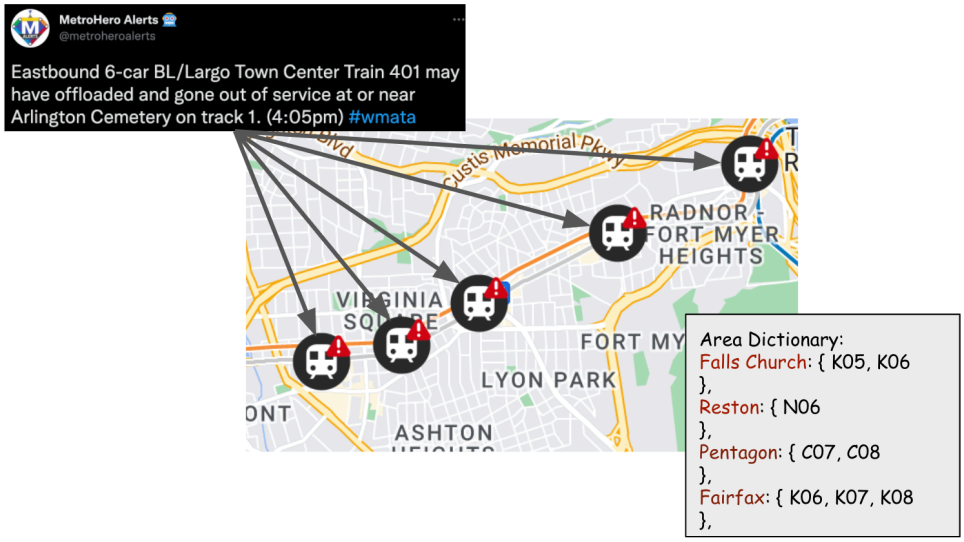

Figure 3.4: Location Match

- **Firebase**: Firebase is famous by its well-performed real-time database, it allow applications to approach cross-platform data within real-time after joining NoSQL cloud-storage. This service also enables developers to work without internet connectivity. The data can still being cached in the memory of local server when the system is offline and start synchronizing after the internet connectivity. This make Firebase one of the best choice for this work, due to its flexibility on real-time data and feasibility for various data types.

- **MongoDB**: MongoDB is schema less database, which one collection holds different documents, and number of field, content and size of document can differ from one document to another. With its ease of scale-out and document oriented features, developers can easily expand and store different types of information, which is a great fit for our system.

For visualization in the front-end, real-time panel will display latest five incidents from Firebase, historical events panel will shows the overall incidents with its detail information.

3.3 News Retrieval

Local news also plays an important part for people lives in different cities, especially for metropolitan cities. In order to obtain news, we use News API for gathering resources. By passing the WMATA related list, as section 3.2 mentioned, we can acquired news targeting to particular place.

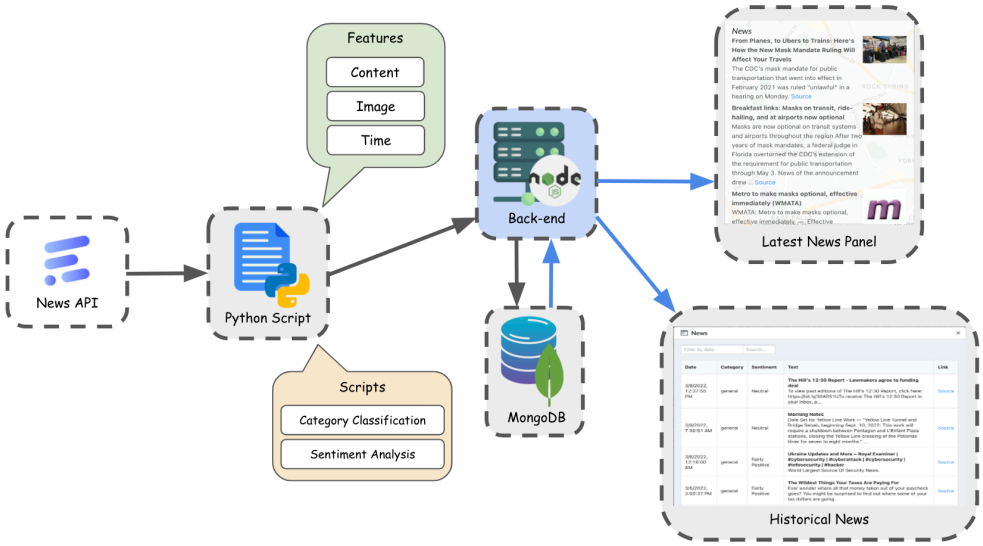

Figure 3.5: Data Pipeline for News Retrieval

Figure 3.5 shows the pipeline of news. News scripts will run once a day to capture the latest news as soon as possible. When news first crawled, it will go over a pre-processing script which extract needed features:

- News Content
- Image
- Published Time

For visualization in the front-end, latest news panel will show nearest 3 days news, historical

events panel will shows the overall news with its detail information.

3.4 Storytelling

Since some Tweets or news are related to one single events, so if we can gathered them and convert them into a story timeline, this will not only help authorities to have broad view of series incidents, but also let customers know the entire progress of the event.

In this section, we have to portions to help us, data pre-processing and data clustering:

3.4.1 Data Pre-Processing

First we need to normalized Tweets and news by converted their content into same dimensional size vector. With Natural Language Toolkit, we can easily conduct this transition. Figure 3.6 below shows how NLP pre-processing is being done by NLTK.

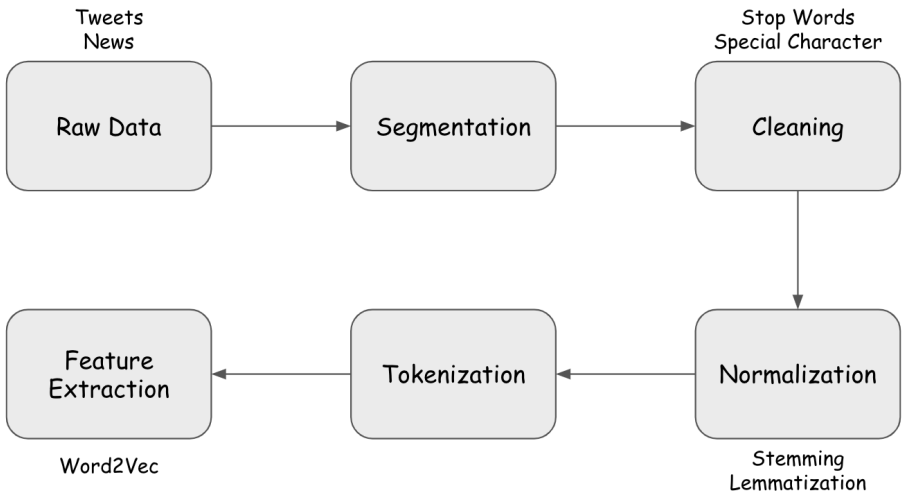

Figure 3.6: NLP Pre-Processing Procedure By Natural Language Toolkit

Step 1. **Raw Data**: In this step, the data is being queried from MongoDB based on different

purposes, daily Tweets to give a view of recent events, weekly Tweets and news for having global view among weeks.

Step 2. **Segmentation**: Text segmentation is the process of dividing written text into meaningful units. The term applies both to artificial processes implemented in computers, and mental processes used by humans when reading text.

Step 3. **Cleaning**: By rearranged human language into a format that machine models can understand, cleaning can be performed using simple scripts that eliminates stopwords and special characters. This elimination will not change the overall meaning of text, but only removes the meaningless words.

Step 4. **Normalization**: Stemming is one of normalization process of reducing a word to its stem or root format. For example, words like “branched”, “branching” and “branches”. They all can be reduced to the same word “branch”. After all, these three words have same meaning. Lemmatization is technique which is used to reduce words to a normalized form. By using dictionary to map different variants of word back to its root format. For example, words like ”is”, ”was” and ”were” can be convert back to their root ”be”. By conducting these two methods, we can reduce complexity while retaining the essence meaning of words.

Step 5. **Tokenization**: Tokenization is a way of separating a piece of text into smaller units called tokens. Tokens can be either words, characters, or subwords which can be corresponding to three different tokenization. In this work, word tokenization is being used, which splits a piece of word text into individual words based on a certain delimiter. Different word-level tokens are formed, depends on delimiters. Figure 3.7 shows sample of pre-processing pipeline from step 1 to step 5.

Step 6. **Feature Extraction**: In this work, pre-trained word2vec model is used which can

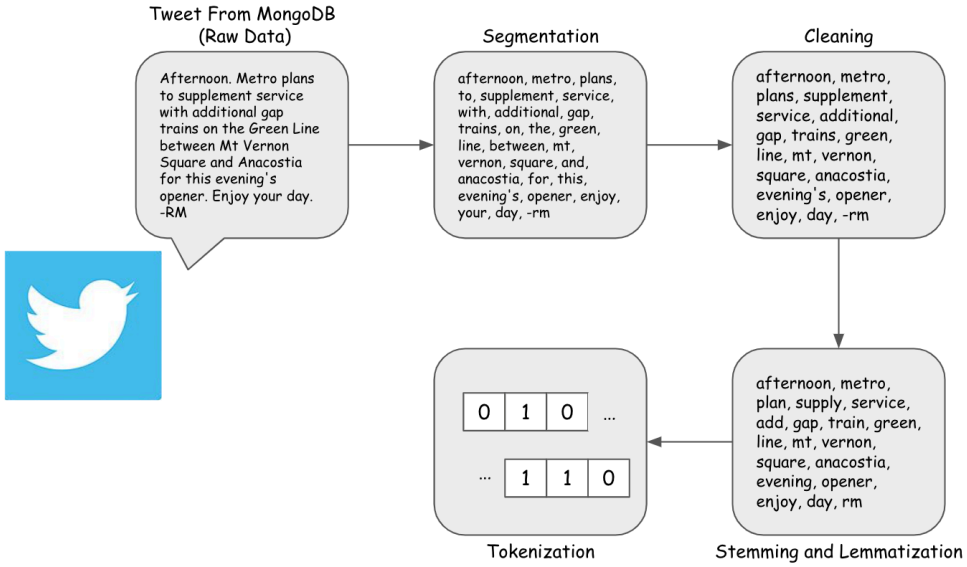

Figure 3.7: Data Cleaning Sample

detect synonymous words or suggest additional words for a partial sentence. As the tokenized word implies, word2vec represents each distinct word with a particular list of numbers. By adapting CNN afterward, we extract the feature into a dot in 300-dimensional space, which can be further clustered.

3.4.2 Data Clustering

Secondly, by applying cluster algorithm, we can clustered feature vectors according to cosine similarity among each others. Scikit-Learn is a Python library that provide variety of clustering package. In this work, we use method called *DBSACN*, which is one of the clustering algorithm provided from Scikit-Learn. DBSCAN has an ability to find arbitrary shaped clusters, especially on finding clusters with noise. This make DBSCAN perfect suit in this work. The main idea of DBSCAN is that a point belongs to a cluster if it is close to many points from the specific cluster. There are two key parameters of DBSCAN:

- **eps**: This parameter provide the radius of the circle which the origin is a point. And if the distance between two points are less than or equal to eps, they are considered to be neighbors.
- **minPts**: This parameter gives the condition of being a cluster, a cluster is being defined is there are minimum number of neighbor points within the radius.

With these two parameters, points are distinguished as core point, border point or noise:

- **Core Point**: Within radius *eps*, if there are at least $minPts$ number of points (including the point itself), a point is defined as core point.
- **Border Point**: Within radius *eps*, if it can be reached by the core point, but the neighbor points are less than $minPts$, a point is defined as border point.
- **Noise**: A point is an noise if it is not a core point and can not reached by any core points.

Figure 3.8 shows the example of how DBSCAN is being conducted. By giving minPts as 3. Red points are core points because within radius eps, there are at least 3 points within their neighborhood. Yellow points are border points because they can be reached by a core point and have less than 3 points within their neighborhood. Finally, blue points are noise, since they can not be reached from core point.

According to offical document[6], DBSCAN start with arbitrary point and retrieves all points density-reachable from that starting point, along with parameter Eps and $MinPts$. By providing $SetofPoints$, Eps and $MinPts$, DBSCAN can start clustering points based on given parameters. Algorithm 1 shows how DBSCAN is being conducted.

Once DBSCAN starting to clustered points, it will loop through each unvisited points, until all points has been visited. Based on the condition of Eps and $MinPts$, DBSCAN will

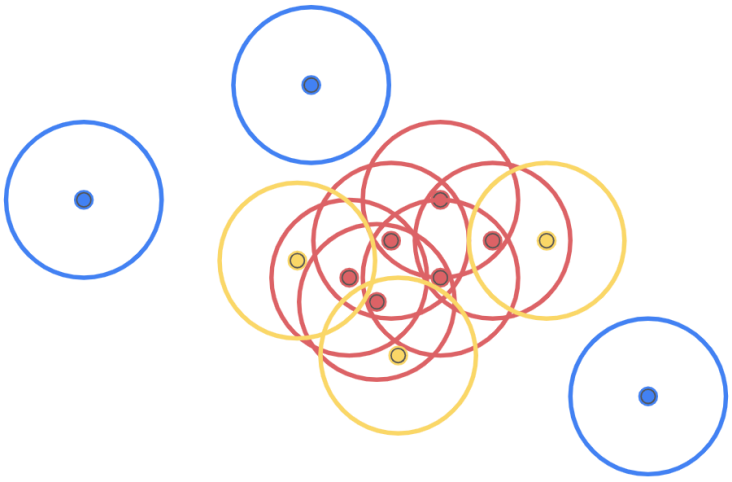

Figure 3.8: DBSCAN visualization

differentiating points as *corepoint*, *borderpoint* or *noise*. By gathering the neighborhood points, DBSCAN will eventually assign every points to its belonging cluster.

Figure 3.9 shows the pipeline for how system conduct storytelling. By querying daily Tweets, weekly Tweets and weekly news, and utilizing data pre-processing and data clustering techniques, the outcome will have series of stories.

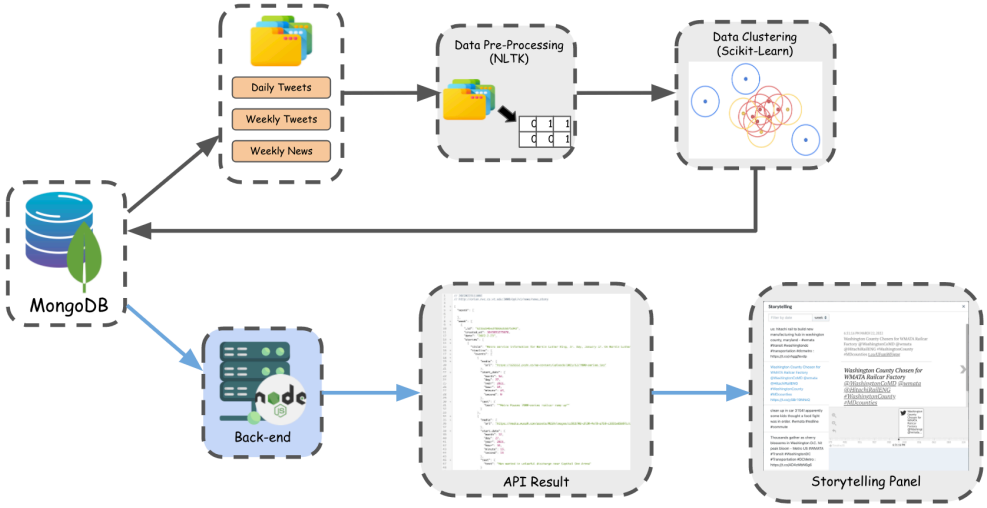

Figure 3.9: Storytelling

3.5 Sentimental Analysis

Figure 3.10 shows how the process of how system conduct sentiment analysis. Once the system receive Tweets or news it will convert content into corresponding scaling sentences, then display along with its Tweet or news.

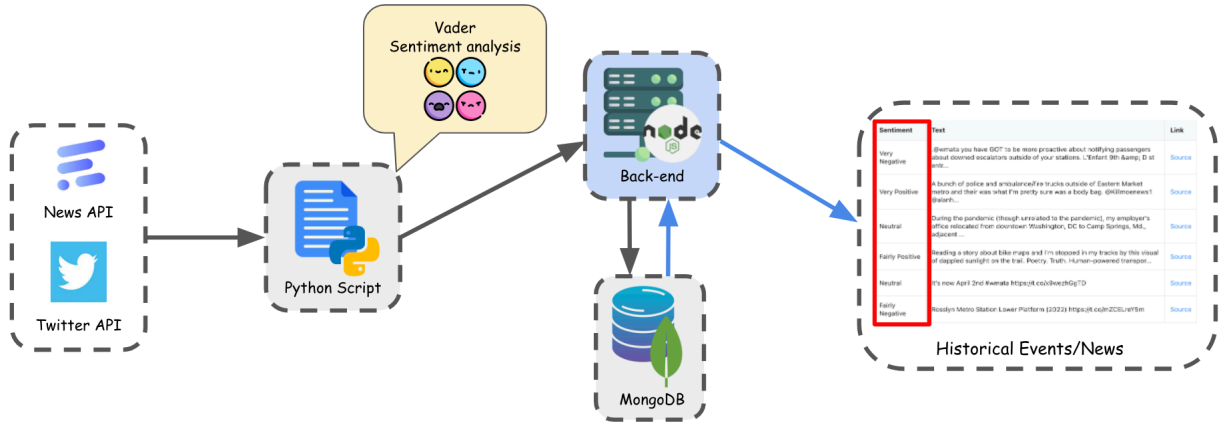

Figure 3.10: Sentiment Analysis

Every Tweet or news contains emotional orientation, which can represent the overall feeling for Tweet or news. By utilizing this characteristic, people can quickly understand the intensity among information.

Table 3.1: Example of *Positive*, *Neutral* and *Negative* Words

Positive	Neutral	Negative
bad	I	like
worst	you	best
hate	it	love
sad	he	happy
injure	she	delight
stupid	they	smart
blame	and	admire
aggressive	or	patient
bomb	the	cheer
gun	what	bless

In this work, we used library in NLTK called *sentiment*. By adapting module, $VADER$, in *sentiment*, we can have sentiment analysis that scale based on the polarity among Tweet or news. $VADER$ is model used for text sentiment analysis that is sensitive to both polarity and intensity of emotion. Sentiment scores is being calculated by mapping lexical features to emotion intensities. This score can be obtained by summing up the intensity of each word in text, which contains four categories; $Negative$, $Neutral$, $Positive$ and $Compound$. Table 3.1 shows the example of words in different categories.

$Compound$ is the normalization result among all other score, which is the score that we used in this work. Table 3.2 shows the scaling table of compound score.

Table 3.2: Scaling Table of Compound Score That Scaling From -1 to 1

Compound Score C	Corresponding Sentence
$-1 \leq C < -0.7$	Extremely Negative
$-0.7 \leq C < -0.4$	Very Negative
$-0.4 \leq C < -0.1$	Fairly Negative
$-0.1 \leq C \leq 0.1$	Neutral
$0.1 < C \leq 0.4$	Fairly Positive
$0.4 < C \leq 0.7$	Very Positive
$0.7 < C \leq 1.0$	Extremely Positive

For VADER sentiment analysis in NLTK, it is intelligent enough to understand the basic context of sentences. The latest update for sentiment analysis also support on emojis. By converting emojis into sentences that represent its meaning, we can treat these emojis as other text in sentence. Figure 3.11 shows the example of dictionary that can convert emoji into corresponding sentence.

Figure 3.11: Dictionary for Emoji Transition

3.6 Emergency Report for Twitter

Since modern incidents mostly are reported manually, there is a gap between the exact incident happened time and the authority got the report of incident. However, with our system we can detect any potential threat once there are being found. By integrating the emergency sensor in our system, we can easily report the incidents right after its being published. Figure 3.12 shows the process of emergency sensor. Threat list contains various dangerous word which can indicate any potential threat. Once the threat is being found, it will sent a alert to the authority. Table 3.3 shows the example of threat list that is being used in our system.

Table 3.3: Threat List Used In Emergency Sensor

Threat List
Mask
Gun
Fire
Knife
Shoot
Stab
Crash
Threat
Bomb

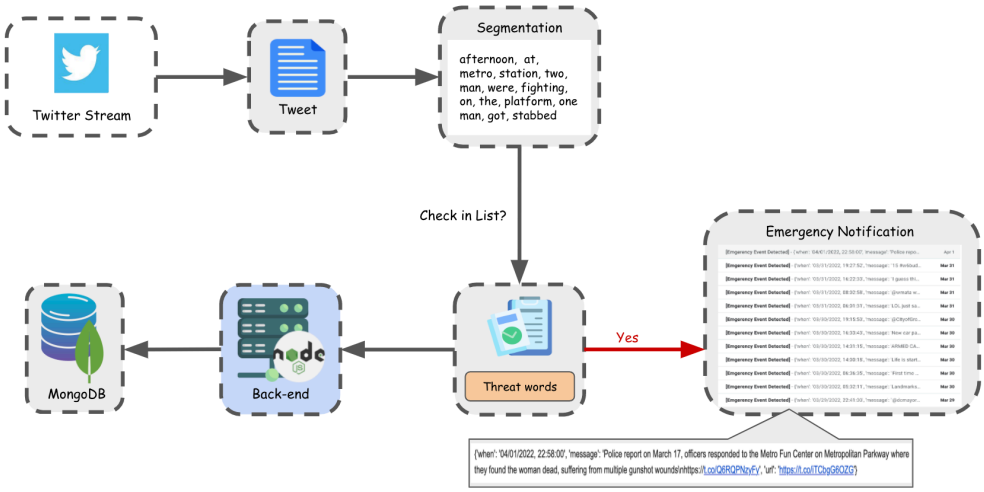

Figure 3.12: Emergency Report

Currently, the alert are sent as email, but the authority can connect this feature to their original report system, which is not only very handy to use, but also having latest situation among incidents.

Algorithm 1: DBSCAN Algorithm Pseudocode

```
DBSCAN(SetofPoints, Eps, MinPts)
Cluster = 0
for each unvisited point P in SetofPoints do
    Mark P as Visited
    NeighborPts = RegionQuery(P, Eps)
    if NeighborPts.size < MinPts then
        Mark P as Noise
    else
        Cluster = Next Cluster
        ExpandCluster(P, NeighborPts, Cluster, Eps, MinPts)
    end
end

RegionQuery(P, Eps)
return All Points within P's Eps − Neightborhood (Including P)

ExpandCluster(P, NeighborPts, Cluster, Eps, MinPts)
Add P to Cluster
for each P^i in NeighborPts do
    if P_i Not Visited then
        Mark P_i as Visited
        NeighborPts_i = RegionQuery(P_i, Eps)
        if NeighborPts_i.size ≥ MinPts then
            NeighborPts = NeighborPts Join With NeighborPts_i
        end
    end
    if P_i Not In Any Group then
        Add P_i to Cluster
    end
end
```

Chapter 4

Case Study & Results

4.1 Case Study

In this section, three cases are found in order to demonstrate our system. All cases are being detected by our system, which is the pipeline of tweets sorted from latest to earliest. Each update and notifications pop on real time panel and events history for particular station(s). For all incidents that happened, a notification is sent. After the end of day, the system will upload new storytelling, which is the combination of storyline among incidents. These cases show the validation and correctness of our system on detecting and updating events.

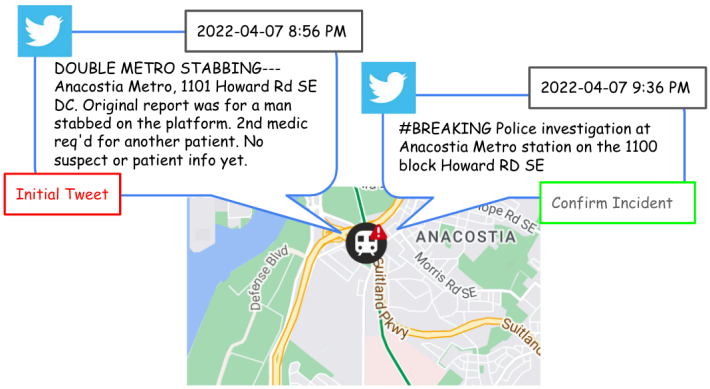

Figure 4.1: Anacostia Metro Stabbing

1. **Anacostia Metro Stabbing 04/07/2022**

   The initial source of incident was found on Twitter at 8:58 pm, when a user tweeted

about a man is being stabbed on the platform at Anacostia Metro Station. The real-time incident panel later updated a following tweet at 9:36 pm, saying that the police is now investigating at the station. After the first Tweet, the first news was reported on FOX news DC at 10:01 pm, which is a hour later. Figure 4.1 gives the information from the incidents was being found till the police start the investigation.

2. **Gallery Place Train Halted 04/05/2022**

   The application notified a situation tweeted by a passenger on the train, which spill, mess, and cracked window were found on train at 12:43 pm. The follow up Tweet said the train then later stop at Gallery Place at 12:51 pm for police to investigate the situations. For the first authority to response this situation is at 1:03 pm, which is a 20 minutes delay on incidents. Figure 4.2 shows the progress from original Tweet to official on the event.

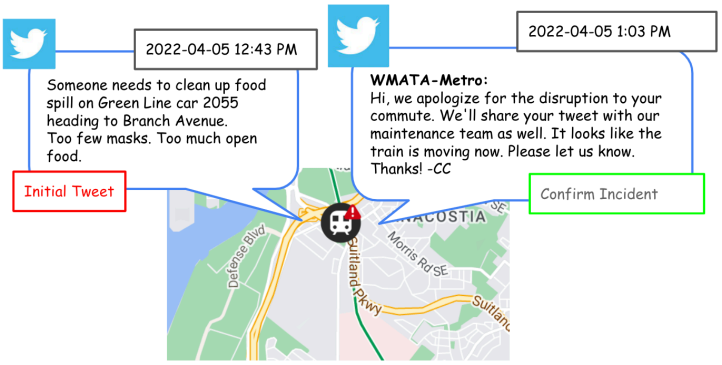

Figure 4.2: Gallery Place Train Halted

3. **Congress Heights Car Accident 04/03/2022**

   The initial Tweet is found at 1:40 AM mentioned a severe metro bus accident that happened near Congress Heights Metro station. The incident is too critical that a 10-year-old child died after being sent to hospital. However, the official news with comprehensive information is not being announced until 10:40 AM. Obviously, the

response time is way delay when accidents happened in the weekend. Figure 4.3 gives the timeline from first tweet to first official report.

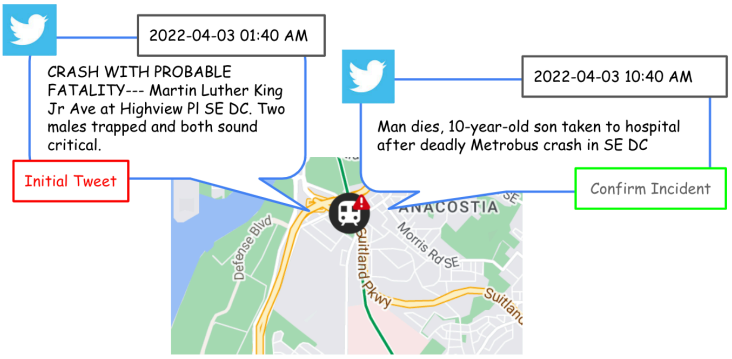

Figure 4.3: Congress Heights Car Accident

4.2 Results

In this section, we will have comparison between our system with [**REFERENCE**] on improvement we make. For experiment purpose, the simple system about [**REFERENCE**] is being built to help us demonstrate the validation of our improvement.

1. **Twitter Stream Modification**

   Originally, the paper create a Twitter stream that contains any information about WMATA. Based on statistical report [**REFERENCE**], the average re-Tweets number for each Tweet is 1690.46 cases. With the amount of re-Tweets being caught by the system, it need huge process time on digest these data. For system-wise is hard for system to process these data and do the scripts at the same time. For database-wise it create huge amount of information that normal application can not handle. By clear out the re-Tweets, it will not only reduce the process time on analyzing critical Tweet, but also store the information that is more important than duplicate Tweets. Table

4.1 shows the statistical comparison among these two approaches.

Table 4.1: Comparison Before and After Twitter Modification

	RISECURE(Before)	Our System(After)
Average Daily Tweets	561 cases	172 cases
Average Daily Data Size	2.5 MB	0.8 MB

2. **Database Modification**

Since both system performed real-time panel, it bring up the critical factor of need for speeding up the query time. In our system, we adapt the database from MongoDB to Firebase. The major reason we do the transition is not only their both NoSQL database which is more easily to reconstruct the pipeline, but also Firebase is dedicated on real-time performance and website applications which will be the better solution when we are dealing with real-time data. Table 4.2 shows the comparison between MongoDB and Firebase on server response time among different queries.

Table 4.2: Comparison Before and After Database Modification

	RISECURE(MongoDB)	Our System(Firebase)
Server Response Time	0.221 sec	0.176 sec
	0.219 sec	0.167 sec
	0.220 sec	0.183 sec
	0.231 sec	0.191 sec
	0.211 sec	0.159 sec
Average Response Time	0.2204 sec	0.1752 sec

3. **Daily Routine on Information Update**

Daily jobs on updating Storytelling and catching the latest news is critical in our system. Also, according to Twitter official document on Twitter Stream usage, once the stream is disconnected it will not connect back on its own, which is a issue that we need to accomplish. By applying Jenkins in our work, we can easily set up a routine jobs that automatically run an update for all features. Jenkins is an open

source DevOps tool that helps developer implement CI/CD workflows, which in this case it can help us execute scripts when it meet the arranged time. Table 4.3 gives the features that will be update by Jenkins based on different period of time.

Table 4.3: Daily Update Features That Executed by Jenkins

Features	Period (hour/time)
Refresh for Twitter Stream	2
Daily News Acquisition	24
Daily/Weekly Storytelling(Twitter)	24
Weekly Storytelling(News)	24

Chapter 5

Conclusions & Future Work

5.1 Conclusions

The main purpose of this thesis is to develop a web application to not only identifying or detecting metro disruptions from Twitter data, but also allowing its user to have a overall picture among incidents. The work for this website was broken down into four main different features. The first features involved data mining technique to conduct real-time panel that keep updating Tweets and news. The second features involved building a pipeline for converting Tweets and news into storylines, group of Tweets/news that are related to particular incidents. The third features involved building another pipeline for analyzing emotional expression of specific Tweet/news. The fourth features involved adapting built-in function in Twitter stream to send alerts once potential threats were being found.

By giving a case studies among features, we found storytelling effectively group Tweets which are related to same events together. Also, the sentiment analysis for Tweets and news is correctly reflecting author's feeling on incidents. For emergency report, the system properly sent the alert when the potential threats are being found. Since we improve most of system design from Zulfiqar et al.[14], we conduct the comparison between two systems. For Twitter stream modifications, our system excluded re-Tweets, which the average daily tweets and average daily data size are both 3 times less than the original one. For database modification, our system replace MongoDB with Firebase, which the average response time

for daily Tweets also faster than the original one. Although the response time for daily Tweets is only 0.05 second, still if we are dealing with the weekly or monthly computing among Tweets, the response will become significantly.

5.2 Future Work

Since our sources which acquired in our website are all came from internet, social media and news, by changing the keyword in section 3.2 and section 3.3, this application can be adapted into different metro metropolitan cities. For the pipeline among section 3.4, section 3.5 and section 3.6, they are all depend on the incoming Tweets and news, so there is no modification needed after we changing the sources.

In section 3.4, we designed a pipeline for clustering the related Tweets into a storylines that shows information of incidents in timely manner. Previous work have developed a well-tested method to predict and detect among transportation events[3, 7, 10]. Since current work did not involve functionality for foreseeing, by adapting the methodology they presented can enhance our system.

In section 3.6, we develop our own emergency report for potential threat. Since every metro metropolitan cities have there own reporting system when incidents were found, once we obtain initial APIs for connecting their reporting system, our emergency report can easily integrate with the original reporting system. Also, the emergency report in our system is currently using threat list on detecting potential threats. By developing the features for adjusting this threat list, user can update new keywords for additional needs. Also, previous work, Smith et al. [12] purposed a framework for flood risk management, by converting its designed into metro incidents allowing us to provide comprehensive update among incidents.

Final task left would be developing search function for allowing user to query based on keyword. There are large amount of data the system acquired from section 3.2 and section 3.3, so it can be hard for anyone to find information about certain words. Once we have this update, users can easily find incidents that are related to the words they want to know, which can also be useful for authority on looking back the incidents.

Bibliography

[1] Jake Blumgart. Big changes coming to d.c.'s transit to boost ridership. https://shorturl.at/iCMYZ, August 2021.

[2] Xiannian Chen, Gregory Elmes, Xinyue Ye, and Jinhua Chang. Implementing a real-time twitter-based system for resource dispatch in disaster management. *GeoJournal*, 81(6):863–873, 2016.

[3] Kaiqun Fu, Taoran Ji, Liang Zhao, and Chang-Tien Lu. Titan: A spatiotemporal feature learning framework for traffic incident duration prediction. In *Proceedings of the 27th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, pages 329–338, 2019.

[4] Yiming Gu, Zhen Sean Qian, and Feng Chen. From twitter to detector: Real-time traffic incident detection using social media data. *Transportation research part C: emerging technologies*, 67:321–342, 2016.

[5] Aditi Gupta, Ponnurangam Kumaraguru, Carlos Castillo, Patrick Meier, et al. Tweetcred: A real-time web-based system for assessing credibility of content on twitter. *arXiv preprint arXiv:1405.5490*, 2014.

[6] Clayton Hutto and Eric Gilbert. Vader: A parsimonious rule-based model for sentiment analysis of social media text. In *Proceedings of the international AAAI conference on web and social media*, volume 8, pages 216–225, 2014.

[7] Taoran Ji, Kaiqun Fu, Nathan Self, Chang-Tien Lu, and Naren Ramakrishnan. Multi-task learning for transit service disruption detection. In *2018 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*, pages 634–641. IEEE, 2018.

[8] Stuart E Middleton, Lee Middleton, and Stefano Modafferi. Real-time crisis mapping

of natural disasters using social media. *IEEE Intelligent Systems*, 29(2):9–17, 2013.

[9] Miles Osborne, Sean Moran, Richard McCreadie, Alexander Von Lunen, Martin Sykora, Elizabeth Cano, Neil Ireson, Craig Macdonald, Iadh Ounis, Yulan He, et al. Real-time detection, tracking, and monitoring of automatically discovered events in social media. In *Proceedings of 52nd annual meeting of the association for computational linguistics: system demonstrations*, pages 37–42, 2014.

[10] Dong-Woo Seo and Soo-Yong Shin. Methods using social media and search queries to predict infectious disease outbreaks. *Healthcare informatics research*, 23(4):343–348, 2017.

[11] Aneesh Sharma, Jerry Jiang, Praveen Bommannavar, Brian Larson, and Jimmy Lin. Graphjet: Real-time content recommendations at twitter. *Proceedings of the VLDB Endowment*, 9(13):1281–1292, 2016.

[12] Luke Smith, Qiuhua Liang, Phil James, and Wen Lin. Assessing the utility of social media as a data source for flood risk management using a real-time modelling framework. *Journal of Flood Risk Management*, 10(3):370–380, 2017.

[13] WMATA. Metrorail ridership grew by 20,000 trips per weekday in 2019. https://www.wmata.com/about/news/2019-Metrorail-ridership.cfm, January 2020.

[14] Omer Zulfiqar, Yi-Chun Chang, Po-Han Chen, Kaiqun Fu, Chang-Tien Lu, David Solnick, and Yanlin Li. Risecure: metro incidents and threat detection using social media. In *2020 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*, pages 531–535. IEEE, 2020.

[15] Ovidiu Șerban, Nicholas Thapen, Brendan Maginnis, Chris Hankin, and Virginia Foot. Real-time processing of social media with sentinel: A syndromic surveillance system incorporating deep learning for health classification. *Information Processing & Management*, 56(3):1166–1184, 2019.