2018

# Content-based Clustering and Visualization of Social Media Text Messages

Sydney A. Barnard
*Wright State University*

# CONTENT-BASED CLUSTERING AND VISUALIZATION OF SOCIAL MEDIA TEXT MESSAGES

A thesis submitted in partial fulfillment
of the requirements for the degree of
Master of Science

By

SYDNEY A. BARNARD
B.S.C.S., Wright State University, 2015

2018
Wright State University

WRIGHT STATE UNIVERSITY

GRADUATE SCHOOL

4/26/2018

I HEREBY RECOMMEND THAT THE THESIS PREPARED UNDER MY SUPERVISION BY Sydney A. Barnard ENTITLED Content-based Clustering and Visualization of Social Media Text Messages BE ACCEPTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF Master of Science.

_____
Soon M. Chung, Ph.D.
Thesis Director

_____
Mateen Rizki, Ph.D.
Chair, Department of Computer Science and Engineering

Committee on Final Examination

_____
Soon M. Chung, Ph.D.

_____
Nikolaos Bourbakis, Ph.D.

_____
Vincent A. Schmidt, Ph.D.

_____
Barry Milligan, Ph.D.
Interim Dean of the Graduate School

## ABSTRACT

Barnard, Sydney A. Department of Computer Science and Engineering, Wright State University, 2018. Content-based Clustering and Visualization of Social Media Text Messages.

Although Twitter has been around for more than ten years, crisis management agencies and first response personnel are not able to fully use the information this type of data provides during a crisis or natural disaster. This thesis addresses clustering and visualizing social media data by textual similarity, rather than by only time and location, as a tool for first responders. This thesis presents a tool that automatically clusters geotagged text data based on their content and displays the clusters and their locations on the map. It allows at-a-glance information to be displayed throughout the evolution of a crisis. For accurate clustering, we used silhouette coefficients to determine the number of clusters automatically. To visualize the topics (i.e., frequent words) within each cluster, we used the word cloud. This tool could be easily used by first response and official management personnel to quickly determine when a crisis is occurring, where it is concentrated, and what resources to best deploy to stabilize the situation.

# TABLE OF CONTENTS

**CHAPTER 1** ..... **1**

**INTRODUCTION** ..... **1**

**CHAPTER 2** ..... **3**

**BACKGROUND** ..... **3**

2.1 A History of Social Media and Crisis ..... 3

2.2 Other Attempts to Create a Platform for First Responders ..... 4

**CHAPTER 3** ..... **6**

**DATASET AND DATA MANIPULATION** ..... **6**

3.1 Dataset Used as an Example ..... 6

3.2 Overall System Architecture ..... 6

3.3 Data Processing Threads ..... 9

3.4 Sliding Time Frame ..... 14

3.5 Analysis of Time Frames ..... 16

**CHAPTER 4** ..... **21**

**VISUALIZATION OF DATA** ..... **21**

4.1 Visualization ..... 21

**CHAPTER 5** ..... **25**

**CONCLUSIONS AND FUTURE RESEARCH TOPICS** ..... **25**

REFERENCES ..... 29

## TABLE OF FIGURES

Figure 1: Process Diagram. ..... 7

Figure 2: The Silhouette coefficient and the number of clusters over the number of iterations in k-means. ..... 13

Figure 3: Sliding time frame ..... 15

Figure 4: Number of messages processed at each iteration ..... 16

Figure 5: Clustering time for each iteration ..... 17

Figure 6: Number of clusters for each iteration ..... 18

Figure 7: Silhouette coefficient over the number of iterations ..... 19

Figure 8: 180-second window time frame metrics ..... 20

Figure 9: 360-second window time frame metrics ..... 20

Figure 10: Web application visual interface ..... 22

Figure 11: Web application visual interface with a different selected word cloud and its associated geolocation points ..... 24

## ACKNOWLEDGEMENTS

I would like to express my thanks to my thesis advisor, Dr. Soon Chung, for his guidance in the completion of this thesis. My thanks also extend to Drs. Nikolaos Bourbakis and Vincent Schmidt, my thesis committee members. I want to thank my dearest friends Amy Foltz and Katie Thompson for reminding me to sometimes step back and enjoy life. Finally, I must express my profound gratitude to my wife Laura for her unwavering support and love throughout my academic career. Without all of them this accomplishment would not have been possible.

# CHAPTER 1
# INTRODUCTION

Most crisis response models rely on outdated communication modes to convey information to those affected (such as the 'top-down' approach from authorities), as well as to gather information, including emergency calls, and to establish a personnel presence on the ground. Newer platforms, such as social media, could provide not only a useful method of communication with those affected by an event, but also a means to capture information about an event as it is unfolding. This type of event information could be used by first responders to act more quickly in a crisis and decide what resources to deploy and where, as they could figure out what is going on before arriving at the scene. Another benefit could be earlier detection of a crisis, particularly for slow events such as an epidemic.

Schmidt and Binner [9] addressed the potential of social media for crisis evaluation by clustering social media data by time and geolocation and then displaying the resulting clusters on a geographical map, as a visual information tool for emergency management personnel. By providing a real-time view of microblog data, as a visual image of the text data and meta-data, emergency management agencies could monitor a situation and even inject responses during an event [9]. We have taken this idea and extended it to cluster the text messages by their contents, in addition to time and geolocation.

Clustering social media text messages, solely by time and geolocation, is based on the assumption that messages sent at the same time from the same location are related, but it may not always be the case. By clustering social media text messages by textual similarity, an emergency response agency would be able to see an overview of current

topics and see how these topics and their locations change over time. After a crisis, this information could be used for analyzing the causes and evaluating the effectiveness of the response. With enough datasets collected from actual crises, this model could be trained to recognize developing events on its own and deliver alerts to first responders.

In this research, we have developed a new tool that automatically clusters geotagged text data, of a format similar to Twitter data, and visualizes the clustering result. For content-based clustering, we used the k-means clustering algorithm, which is very efficient for the clustering of large text data, due to its relatively low computation requirement and high quality [19]. We also used the silhouette coefficient [7] in order to determine the number of clusters automatically, while maintaining an acceptable level of clustering accuracy. To visualize the topics within each cluster, we used the word cloud, which is an image composed of words appearing in a cluster where the size of each word in the image indicates its frequency.

This thesis is organized as follows: Chapter II reviews how social media data has been used for crisis management. In Chapter III, we describe the dataset chosen as a working example and then how the dataset is preprocessed and clustered. Chapter IV describes the visualization dashboard components and interactivity. Chapter V contains the conclusion and future research topics.

# CHAPTER 2
# BACKGROUND

## 2.1 A History of Social Media and Crisis

Since the publication of [9], there have been a few attempts to use new social media platforms for crisis management. Short Message Service (SMS) has been successfully harnessed for crisis management, particularly within closed communities. SMS alerts are sent to members of a college campus during certain situations including severe weather, dangerous chemical leak, active shooter situation, etc. Services, such as Campus Alerts, allows educational professionals to set up a service and register students to receive alerts and also provides a digital 'panic button' that teachers/staff can press in a crisis to alert first response personnel to their location [1].

Twitter and other microblogging platforms, such as Sino Weibo (similar platform for Chinese users), have been used during crisis situations. However, much of this use has been from civilians on the ground, rather than directed or harnessed by official personnel. During the Haiti earthquake in 2010, media sources used Twitter data in their reporting efforts when other forms of communication on the ground had been lost [5]. Facebook members can use the 'Facebook Safety Check' to notify friends and family that they are safe during a crisis; however, this service was activated manually by Facebook and was not available until reports of a crisis or a natural disaster had spread. Since the feature was released in 2014, it has been used more than ten times and has been used additionally for terrorist attacks in both Paris and Manchester [6]. In 2016, the feature was enhanced to be automatically enabled if enough people in an area are talking about an event [6].

An additional concern is that some social media users, who are not directly involved in the crisis, may repeat outdated information or converse about the topic over social media. For instance, during the Ebola outbreak in Africa in 2014, when a few travelers were tested for the disease in the United States, the topic was trending on Twitter in the United States even though none of them tested positive [8]. Moreover, misinformation is easily spread through social media platforms, even by well-intentioned users. During the Ebola outbreak, inaccurate claims about the nature of the disease and to which areas it had been spread were evident on Twitter [8].

## 2.2 Other Attempts to Create a Platform for First Responders

It was reported in [2] that official emergency personnel were not using the organically emerging local hashtags from Twitter during weather emergencies, and even if they were overwhelmed by a huge number of tweets on related topics, it was still possible to find a few official tweets if they looked at a narrowly affected location. In [2], they proposed search strategies to find official messages during a crisis, but did not provide any visual interface.

Another attempt was made to use geolocation to track crisis events using Sino Weibo [3]. They used textual clues as well as the user profile and the geolocation data for each message, in order to identify the location of a crisis. However, this method was not tested with live crisis data, and the textual clues rely on knowing what type of event is going on.

Another study attempted to collect data from Twitter, allow the user to setup categories with associated keywords, and then display the selected results on a heat map

[10]. Tweets from different categories are displayed with different Twitter icons, but the heat map is set up using all selected categories, thus creating affected areas that may contain more than one event or different types of events. Their approach also saves Twitter data over a period of seven days, so that the search timeframe can be widened or narrowed.

In [4], an emergency situation awareness system using social media is proposed, and its visualization tool can display Twitter data captured from an event on a map [4]. This map displays pins with colors, indicating the number of messages in a given set. A burst detection algorithm is used on the data to determine when an event is going on, and the tweets are then clustered by using additional features. It also provides a separate page with a time slider that shows trending words and associated tweets over time. However, it is not clear how responsive the system would be to live tweets as they come in and whether the geolocation map can indicate the type of events occurring without selecting each cluster.

# CHAPTER 3
# DATASET AND DATA MANIPULATION

## 3.1 Dataset Used as an Example

The dataset used in this visualization tool is the dataset from the 2011 IEEE VAST Challenge, related to the 2011 IEEE Conference on Visual Analytics, Science, and Technology (IEEE VAST). The mini-challenge 1 dataset (Geospatial and Micro-blogging Characterization of an Epidemic Spread) [11] is similar to that of a real-life crisis and in the same format as a microblog captured from Twitter. While we are not specifically using the data to answer the IEEE VAST Challenge, the resulting visualization interface could potentially be used to track the origin of the epidemic posed in the challenge. The dataset contains an ID, a timestamp, geospatial codes, and the message text. We consciously designed our visualization in such a way that it would be trivial to change the data source from the IEEE VAST sample data, presented in a comma-separated values (CSV) file, to live data captured from Twitter's advanced programming interface.

## 3.2 Overall System Architecture

All the technologies used for this project are open-source. The benefits of open-source solutions are that they are typically free and have a large number of developers contributing to their features, upkeep, and bug fixing. Particularly for a project that would benefit emergency response agencies, keeping costs at a minimum would allow more agencies to implement these solutions. We also took steps to keep processing speed as fast as possible, in order to handle a continuous stream of live data, and platforms and

tools were chosen and implemented with this in mind. If the data preparation and processing takes too long, the data could be outdated by the time it reaches the visual interface and not as useful to first responders.

In order to ensure the data could be either file data or live streaming data, we used a pipeline to transfer the data into different parts of the program. Kafka [13] (a distributed streaming platform used to build a real-time data pipeline) was used in conjunction with Zookeeper [14] (an open-source server) to pipe the sample data from a source, (a CSV file in this case, or alternatively a Twitter stream) into the clustering process, and finally to the visualizer. This process is illustrated in Figure 1, with Zookeeper and Kafka visible in the upper center. By using a pipeline, the data processing threads and the visualization threads are unconcerned with the source of the data. An additional benefit of this approach is that event data captured and stored in a compatible CSV format could be brought back into the program for additional analysis, as well as to train event management personnel and find ways to improve responses.

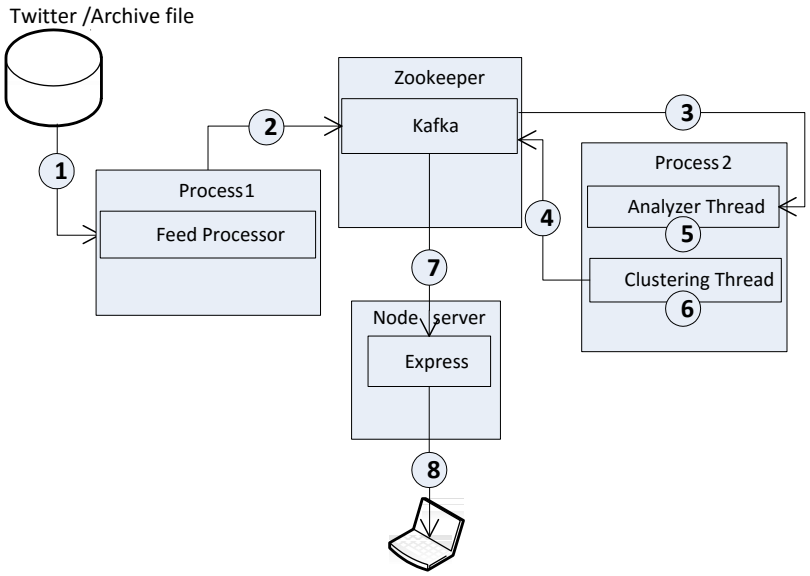


*Figure 1: Process Diagram.*

Schmidt and Binner's work took a 10,000-record sample from the IEEE VAST dataset [9]; but when we experimented with this method, too much data were lost, making the textual clustering ineffective or misleading. Additionally, this would be difficult to implement with live-stream data from a social media source in the case of a real crisis as a large number of messages would need to be captured before sampling. Instead we simulated a Twitter stream by reading data from the text document containing the IEEE VAST samples in order (eliminating the need to reorder the data by timestamp), and sending them to the data processing (i.e. analyzing and clustering) program in groups. This means, however, that clustering the messages first by time is meaningless.

The method used in [9] requires time gaps within the message data in order to cluster the messages. This is not adequate in a real-world scenario, particularly in a large or major city, where a large number of messages could be captured every minute. As the sample data's timestamp is only accurate to the minute, clustering real-time data by that method may result in the messages clustered minute-by-minute, which is not helpful for textual comparison.

We thought about clustering the data by time in a different way by counting the average number of messages per minute and splitting the data into groups, where the average number of messages changes significantly. However, we had two concerns with this approach. First, it may split 'conversations' in the data stream, which could lead to a less noticeable visual impact of the data. Second, we are faced with the difficulty of determining a good threshold value to split the messages, without having access to the whole data.

### 3.3 Data Processing Threads

The data processing program was written in Python to take advantage of many Python libraries that are available for large data processing. This program is split into two main threads — an analyzer thread and a clustering thread. The analyzer thread takes the text messages periodically provided by the Kafka pipeline and places the data into a shared list. The clustering thread accesses the shared list to retrieve the data for clustering. The access to the shared list is controlled by a semaphore, which guarantees the mutual exclusion between the analyzer and clustering threads on the data. Splitting the program into two concurrent threads allows us to process data in groups, while still being able to receive additional data. If the data source is changed to a live stream, the analyzer thread will continually receive messages from Twitter, instead of receiving messages in groups from the CSV processor. We do not have to wait for a specific number of messages to be received before proceeding, nor do we need to worry about using the reception of data as a trigger for the clustering step.

The python Natural Language Toolkit (NLTK) is used to preprocess the data., Stop-words and punctuation marks are removed from the text of each message. A lemmatizer reduces the inflectional forms of a word to its basic form, known as the lemma, by using a vocabulary and morphological analysis of words. As a result, plural words are changed into their singular forms (e.g., ‘wolves’ would be changed to ‘wolf’). A tokenizer is then used to split each text string into an array of words referred to as word tokens. A simple tokenizer was used to save processing power and memory space.

In the clustering thread, the data from the shared list is vectorized using a Term Frequency, Inverse Document Frequency (TF-IDF) vectorizer. It represents each message

as a vector of all terms, and then creates a numerical statistic to indicate the importance of each word in a message. TF-IDF provides the weight of a word by taking into account the frequency of the word within a document as well as the number of documents containing the word. Thus, if a certain word, like 'I', appears in many documents, it would have less weight than a relatively rare word like 'fire'.

The TF-IDF representation of a document d is:

$$d_{TF-IDF} = \left[ TF_1 \log\left(\frac{n}{DF_1}\right), TF_2 \log\left(\frac{n}{DF_2}\right), \ldots, TF_D \log\left(\frac{n}{DF_W}\right)\right]$$

where $TF_i$ is the term frequency of term $i$ in $d$, $DF_i$ is the number of documents containing term $i$, $W$ is the total number of unique terms in the dataset, and $n$ is the total number of documents. To account for the documents of different lengths, each document vector is normalized to a unit vector (i.e., $\|dTF-IDF\| = 1$).

The TF-IDF function used is from the Python scikit-learn library for machine learning and data mining. The resulting TF-IDF representations of messages are then clustered using k-means, which is also available in the scikit-learn library. We used k-means because it is very efficient for the clustering of large text data, due to its relatively low computation requirement and high quality [19]. The steps of k-means are as follows:

1. Select k initial cluster centroids, each of which represents a cluster.
2. For each document in the whole dataset, compute the similarity with each cluster centroid and assign the document to the closest (i.e., most similar) centroid.
3. Recalculate k centroids based on the documents assigned to them.
4. Repeat steps 2 and 3 until convergence.

Once the clustering is performed, the top ten words from each cluster are

determined. Each message in the shared list is labeled with a cluster id and its geolocation information, packaged in a message formatted with Java Script Object Notation (JSON), and then sent back to the Kafka pipeline (Figure 1(4)).

To measure the similarity between messages, we used cosine similarity which is most commonly used in text clustering [20]. For two documents $d_i$ and $d_j$, the similarity between them can be calculated as:

$$cos(d_i, d_j) = \frac{d_i \bullet d_j}{\|d_i\| \, \|d_j\|}$$

Since the document vectors are of unit length, the above equation is simplified to:

$$cos(d_i, d_j) = d_i \bullet d_j$$

The cosine value is 1 when two documents are identical and 0 if there is nothing in common between them [20].

In the clustering thread, an additional step is also performed, which calculates the silhouette coefficient. The silhouette coefficient is a comparative value based on the tightness and separation of the clusters created [7]. In general, it is a measure of how similar each object is to its own cluster, compared to other clusters, and it could be used to enhance the accuracy of clustering.

For each object *i*, its silhouette coefficient *s(i)* is defined as:

$$s(i) = \frac{b(i) - a(i)}{\max\{a(i), \ b(i)\}}$$

where *a(i)* is the average dissimilarity of *i* with all other objects within the same cluster, and *b(i)* is the lowest average dissimilarity of *i* to any other cluster that doesn't contain *i*. Thus, *s(i)* is always between -1 and 1 and when it is close 1, object *i* is appropriately clustered. On the other hand, if *s(i)* is close to -1, object *i* better be assigned to the cluster with the lowest average dissimilarity *b(i)*. If *s(i)* is close to 0, object *i* is on

the border between the two clusters [7].

When k-means is used, an important issue is how to determine the number of clusters, and we used the average silhouette coefficient of all messages (simply called silhouette coefficient) to determine the number of clusters to be used for each group of messages. For example, if the silhouette coefficient increases, the next group of messages will be clustered into one more cluster than the previous group. This is repeated until the silhouette coefficient stabilizes, which indicates that adding more clusters will not enhance the quality of the clusters. On the other hand, if the silhouette coefficient becomes higher than certain threshold value, the number of clusters could be decremented for the next group of messages, in order to reduce the clustering time.

Figure 2 shows the silhouette coefficient and the number of clusters produced for successive groups of messages by using k-means. We can see that the number of clusters increases with each iteration of k-means until it levels off around iteration 63, as the maximum number of clusters was set to 50. Similarly, we can see that the silhouette coefficient also trends upwards over time (with a few dips along the way).

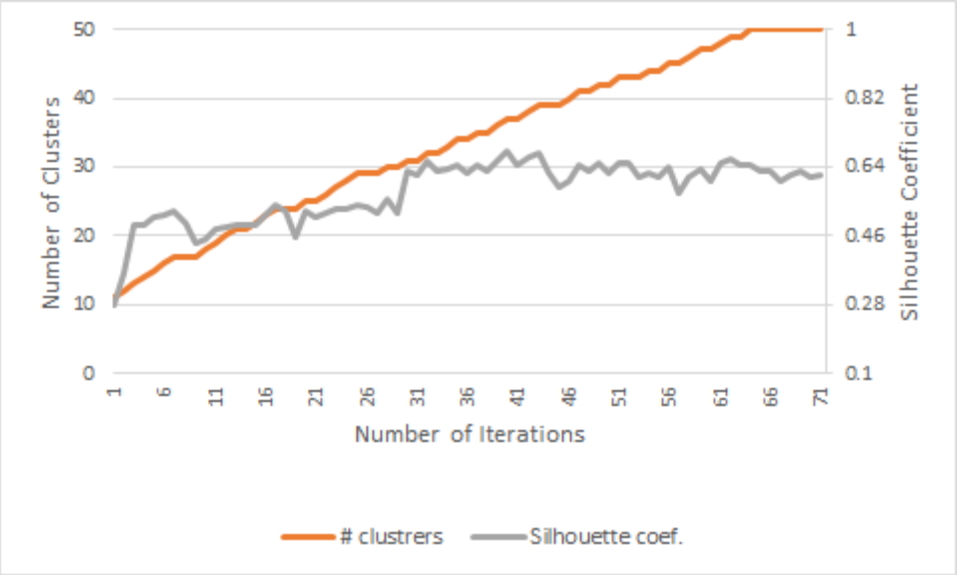


*Figure 2: The silhouette coefficient and the number of clusters over the number of iterations in k-means.*

To reduce the amount of time for the silhouette coefficient to stabilize, we implemented a method to obtain a reasonable silhouette coefficient in the first iteration. We start with a specific number of clusters — in this case, we chose 10 because it aids in a simpler visualization — and compute the silhouette coefficient for this clustering result. We then compare the silhouette coefficient against the target silhouette coefficient (we chose 0.5). If the silhouette coefficient is greater than or equal to the target value, we don't repeat the clustering. If the silhouette coefficient is less than the target value, we increase the number of clusters by 2 and cluster again. This is repeated on the first group of messages until either the target value is reached or our maximum number of clusters (we chose 50) is reached.

After the first iteration (i.e., the clustering of the first group of messages), the clustering and silhouette coefficient computation take place only once for each successive group of messages. For the second iteration, initially we use the number of clusters of the

first iteration. If the silhouette coefficient is better than the target value and also better than the previous silhouette coefficient, then we decrement the number of clusters by 1. On the other hand, if the silhouette coefficient is still less than our target value and the number of clusters is less than our maximum number of clusters, we increment the number of clusters by 1. The new number of clusters is then used for the next group of messages. This process is continued until it is halted.

An additional benefit of this method is that we can reduce the number of clusters for the next group of messages if the current silhouette coefficient is too high. This leads to an overall balance between the number of clusters and the silhouette coefficient representing the clustering accuracy.

### 3.4 Sliding Time Frame

In the clustering thread, two timeframes are setup to proceed the message analysis in increments. The first timeframe is the window timeframe. It determines how long the process waits before beginning the data clustering. The second time frame is the batch time frame, which is set to shorter than the window time frame. Once the batch time frame is setup, the clustering thread pulls the data from the shared list and removes data that is older than the current time minus the window time frame.

We can refer to Figure 3 for better understanding of this process. For example, the clustering thread may look at 2 minutes of data (the window time frame), but advance the window by 1 minute each time (the batch time frame). The first clustering will include all the messages within the first 2 minutes, which are represented by the message group $T_0$ in Figure 3. The process will then wait for the batch time frame of 1 minute, during which

additional messages are collected. After that 1 minute is up, any messages collected from the current time (which is at 3 minutes from the beginning) minus the 2-minute window time frame are kept, and any messages that fall outside of that is removed, resulting in the second clustering containing all the messages from 1 minute to 3 minutes (represented by the message group $T_1$ in Figure 3).

There are some advantages in processing microblogs using this sliding time frame: this way, conversations that are occurring through social media are not missed, even if they would have been at the boundary of a window time frame, as long as they are caught by the batch time frame. For example, messages related to an event may occur partway at the end of a window time frame and may not recognized clearly (within that time frame), but if they are grouped with continued messages during the following overlapped window time frame, the trend may be more apparent. Another important benefit is that the amount of the messages does not grow so large as to be prohibitive for processing, unlike the case where all the messages are saved and clustered once in a while, and most recent trends are not obscured by old ones. This approach also helps show the gradual movement or evolution of a crisis in progress by taking enough amounts of previous data into account while gradually removing outdated data.

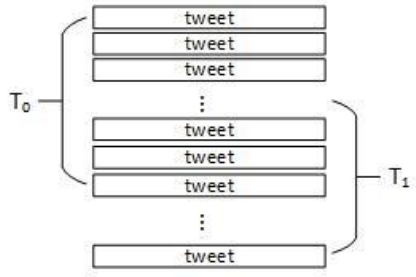


*Figure 3: Sliding time frame*

### 3.5 Analysis of Time Frames

We ran a few tests to see how the window time frame affects various metrics. We ran six tests with different window time frame values, incremented by 30 seconds for each run. We started with a 180-second window, then went up to 210 seconds, 240 seconds, and so on, until we reached 360 seconds. Each test used the same data input.

The first measurement we can look at is the number of messages processed at each iteration of clustering. As expected, in Figure 4 we can see that the number of messages processed at each iteration becomes larger as the window time frame increases. This is self-explanatory, as a longer time frame should have more messages to process, unless there was a time period during which no message was recorded, for some reason. We can also see here that the number of iterations taken to reach a stable number of messages increases as the window time frame increases (see iterations 1-12 in Figure 4), because it takes more iterations before the message queue, defined by the window size, is saturated.

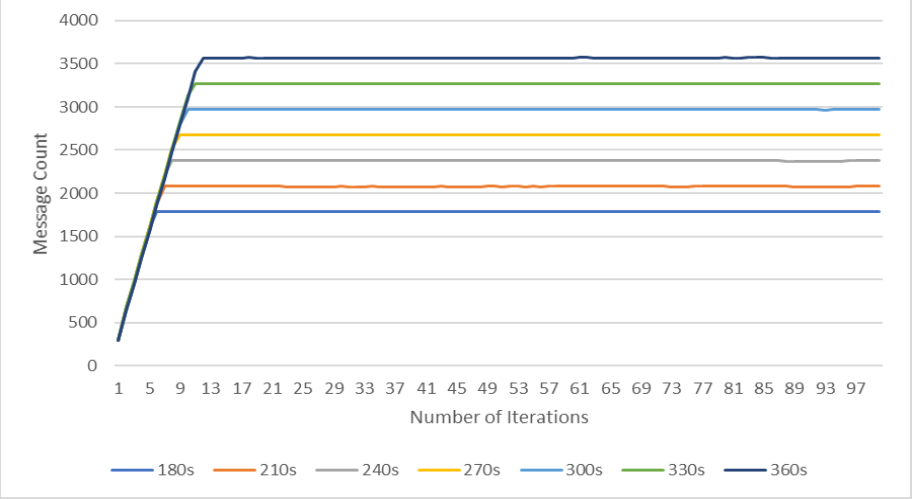


*Figure 4: Number of messages processed at each iteration*

The second measurement that we are interested in is the clustering time required for each iteration. Usually there are more messages within a larger window time frame: for example, within a 180-second window time frame, there are 1787 messages on average, whereas within a 360-second window time frame, there are 3566 messages. Thus, when a larger window time frame is used, a longer clustering time is expected, for each iteration. In Figure 5, we can see that this is generally the case, even though there are some spikes in the clustering time taken. This could be caused by a couple of things. One possibility is that the messages in that particular iteration are unusually dissimilar, so that it takes a longer time to cluster them. Another possibility is that, as this test was not run on a dedicated processing system, occasionally some background processes might have caused a slower reaction for the clustering process.

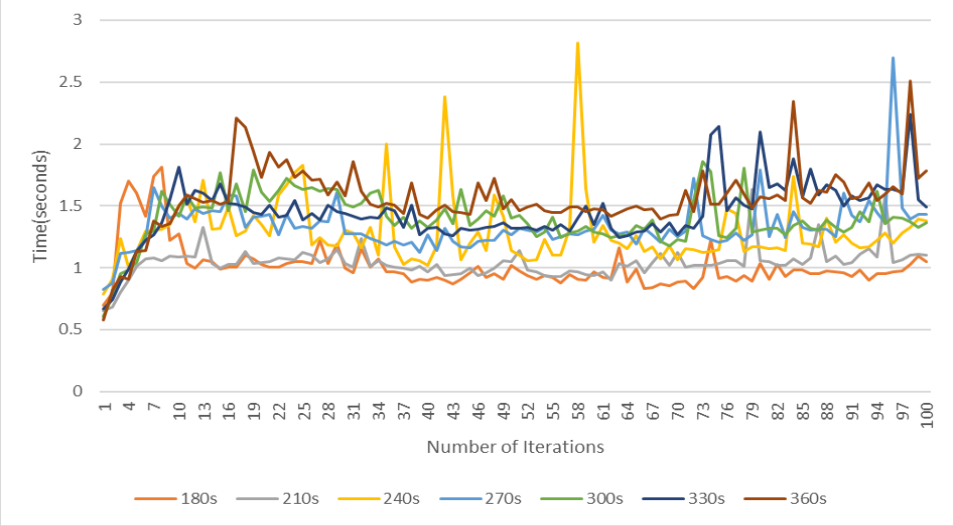


*Figure 5: Clustering time for each iteration*

The third metric that we recorded was the number of clusters for each iteration. Figure 6 shows that, at each iteration, the number of clusters is not sensitive to the

window time frame size, even though a larger window time frame usually contains more messages than smaller ones. One reason is that the number of clusters is adjusted based on the silhouette coefficient as long, as it is under the maximum of 50.

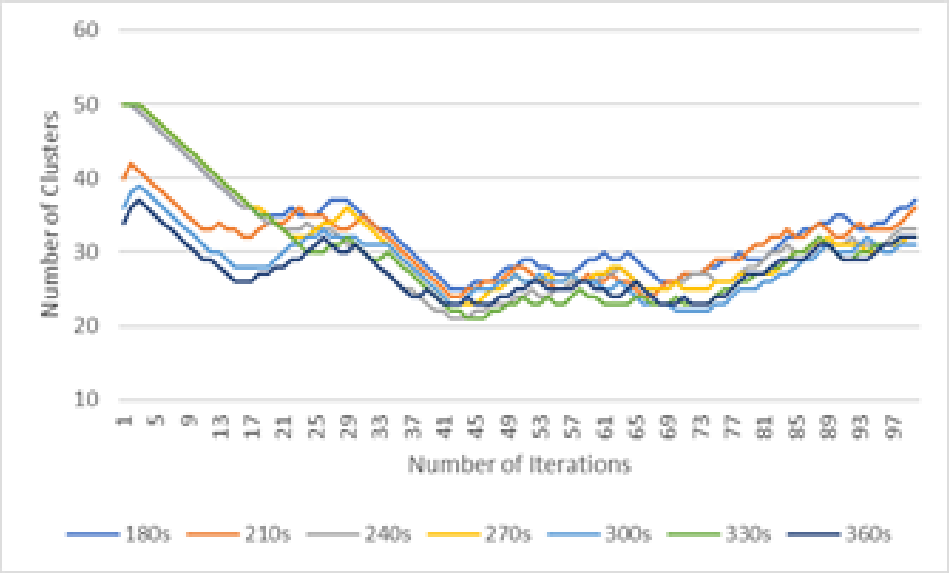


*Figure 6: Number of clusters for each iteration*

In Figure 7, we can see that the silhouette coefficient is fairly stable over all the window time frame values, running within a small range between 0.47 and 0.71. This is a pretty good indication that our method to stabilize the silhouette coefficient is working properly, as there are not huge swings in its value.

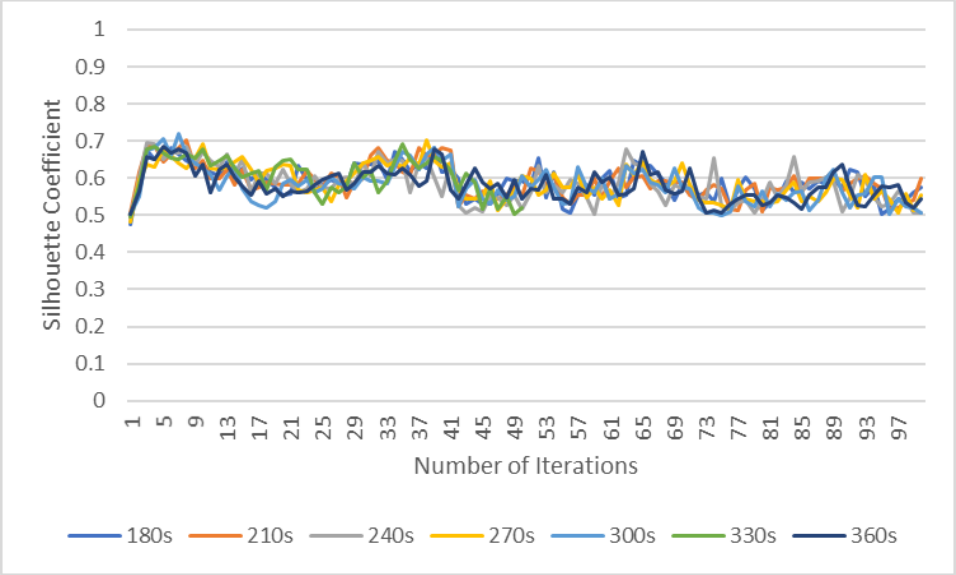


*Figure 7: Silhouette coefficient over the number of iterations*

If we look at the metrics just for the 180-second window time frame (shown in Figure 8) or just the 360-second window time frame (shown in Figure 9), we can see that the number of clusters is adjusted downwards over time, but it does increase whenever necessary to maintain an acceptable level of silhouette coefficient. We did not show the number of messages in these graphs as it was pretty stable (1786-1788 messages per iteration for the 180-second window time frame, and 3566-3570 messages per iteration for the 360-second window time frame). One interesting note is that the silhouette coefficient and the number of messages are more stable for the 360-second window time frame. This can be explained by the larger number of messages in the 360-second window time frame that overlap into the successive window time frame, which makes the textual similarity changes less from one iteration to the next one. The main drawback of the 360-second window time frame is that its clustering time is nearly the twice of that of the 180-second window time frame and its refresh rate of data on the screen is also

slower.

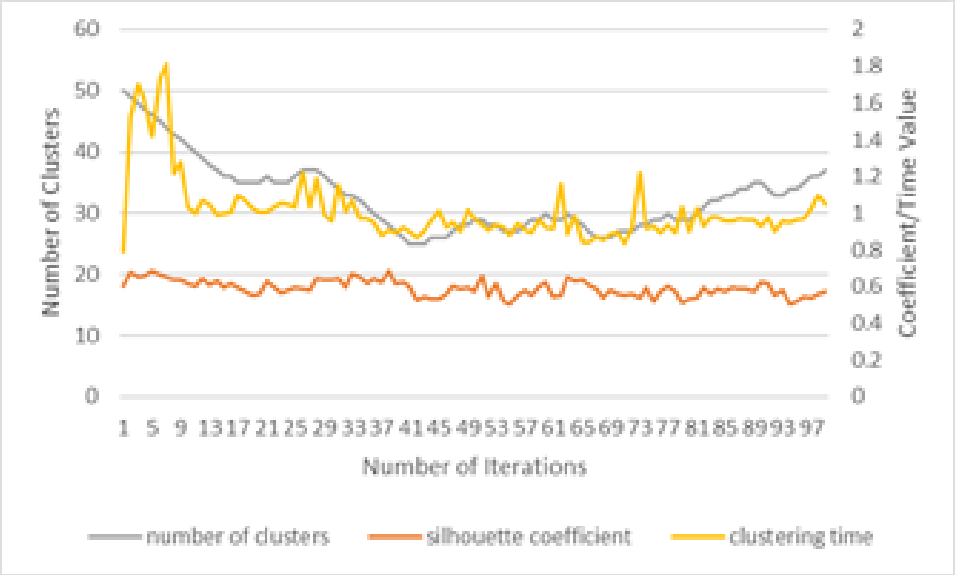


*Figure 8: 180-second window time frame metrics*

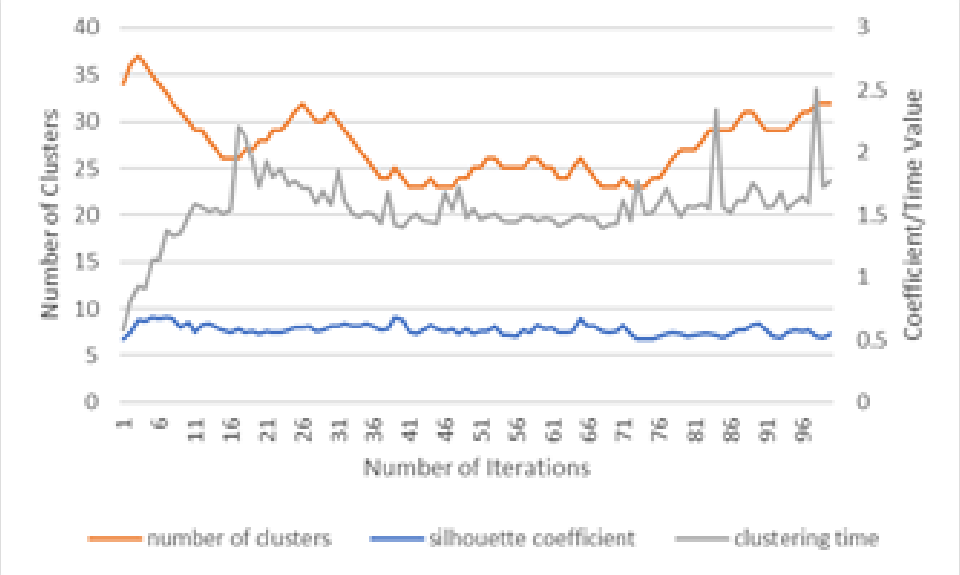


*Figure 9: 360-second window time frame metrics*

# CHAPTER 4
# VISUALIZATION OF DATA

## 4.1 Visualization

The primary purpose of visualizing the data is for it to be useful as a tool for first responders and emergency management personnel. They should be able to use the visual interface as a means to see changes in the social media topics over time, and to discern when and where their topics of interest (e.g., fire, flood, illness, etc.) appear. A useful visualization will allow a crisis professional to tell at-a-glance if a crisis is likely underway, to determine when and where it began, and what kind of a crisis it is.

The visualization we came up with to meet this goal is an interactive web application with a map on which individual message's geolocation can be shown, with a section on the left that contains a word cloud for each of the clusters that were generated by the data processing part of the system. As a new message group is processed, the display is refreshed with new word clouds, and the corresponding geolocation pins for the messages also change. Rather than showing all the messages on the map at once, showing the pins of a selected cluster makes the map more informative and less cluttered as shown in Figure 10.

If there is an event occurring, we would expect certain words related to the event to show up more prominently in the word clouds over multiple consecutive message groups. On the other hand, if no event is occurring, the world clouds should demonstrate more randomness, or contain words unrelated to any crisis situation. The word cloud is

rendered using an open-source library called JQCloud [15].

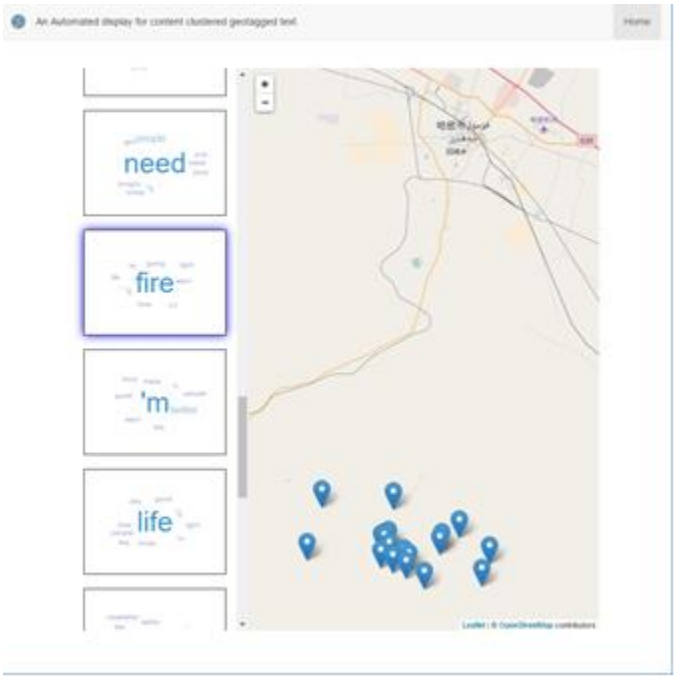


*Figure 10: Web application visual interface*

We used Express [16], which is a minimalist web framework, with Node.js [17] to provide a web interface for the visualization of data. Using a web application for the visualization of data provides more flexibility than using a program that must be installed; once the web server is setup, the visualization of data can be viewed from multiple platforms and even on mobile devices, such as cell phones and tablets. This allows the first responders to continue checking on current data even in transit to or at the site of an event.

In the server code, a Kafka consumer was set up. This allows the web application to connect to the Kafka server (Figure 1 component 7) in order to receive data from the

clustering thread. When the Kafka consumer receives data, the web interface is notified through a socket connection that new data has been received for display. This allows the data on the screen to refresh periodically as new data becomes available. The web application's JavaScript code then processes the new data.

The web interface itself uses Jade [18], a template language for HTML, with JavaScript providing the code behind. The JSON object sent by the clustering thread is parsed by the JavaScript in the web interface. Leaflet.js [14] was used to create an interactive map on the web application. As the sample IEEE VAST challenge dataset has geocodes that correspond to locations in China, all data points currently displayed are in China. If data with different geolocation tags are used, the data will be displayed accordingly, without any necessary reconfiguration. The map displays pins in each location that a message was sent, and can be dragged by the user to view different areas, and also zoomed in and out.

The top ten frequent words from each cluster are displayed as a word cloud along the side of the map. These clusters are dynamic — as many word clouds display as there are clusters generated by the k-means algorithm in the clustering thread. These word clouds are also interactive: If a user clicks on a cluster, the cluster's text messages are displayed with the pins on their geospatial locations. This helps emergency management personnel to see who sent messages about one of the topics highlighted by the word cloud. Figure 11 displays the pins for the geolocations of the cluster highlighted (with the most frequent word 'tonight'), which can be contrasted with the pins displayed in Figure 10 (with the most frequent word 'fire').

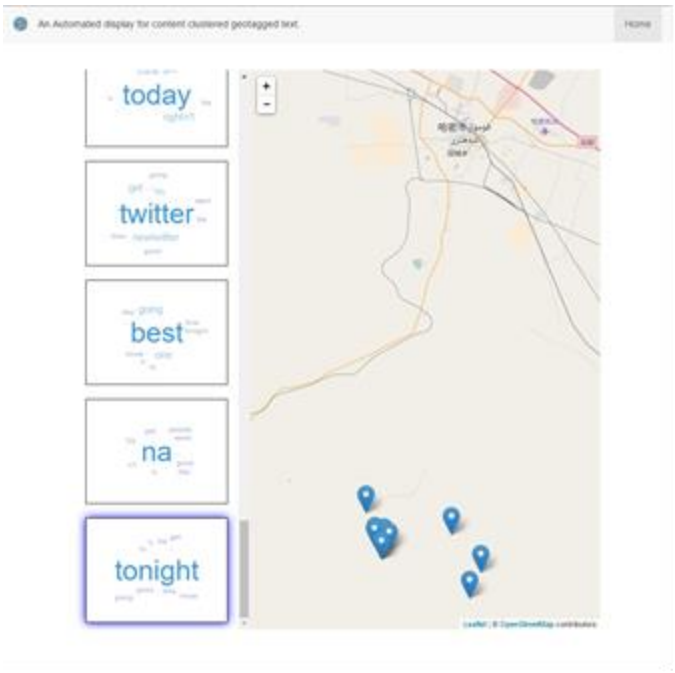


*Figure 11: Web application visual interface with a different selected word cloud and its associated geolocation points*

# CHAPTER 5
# CONCLUSIONS AND FUTURE RESEARCH TOPICS

For microblog or SMS data to be useful to emergency management agencies, a solution that presents an overview of information relevant to the management personnel, as well as to the location of the agency should be implemented.

We have developed a tool that clusters messages by their textual content within a time frame and displays the clusters and their locations on the map, in order to provide more information than just clustering messages by time and geolocation alone. For accurate clustering, we used the silhouette coefficient to determine the number of clusters automatically, at the same time to enhance the clustering accuracy. To visualize the topics (i.e., frequent words) within each cluster and their frequencies, we used the word cloud.

The main advantages of our tool are:

- It allows content-based clustering of text messages and displays the topics within each cluster.
- It can process a large number of text messages efficiently by using multithreaded concurrent processes.
- It is automated and easy to use because only few parameter values are required to be set by the user, such as the threshold for the silhouette coefficient (for clustering accuracy) and the length of window time frame (for clustering time interval), which are very intuitive.
- Its at-a-glance display of the content and locations of clusters can help quickly determine when a crisis is occurring, where it is concentrated, and what resources to deploy to stabilize the situation.

- It is very cost-effective as it is implemented using open-source software packages.

Our tool is very efficient, but it could be enhanced further to provide more useful features to emergency management personnel.

- More filtering of the text messages could be done in the preprocessing and analyzing process. Messages could be filtered based on the relevance to specific crisis-related topics, which could be provided by a list of crisis-related words. The tokenizer could also be changed from the general tokenizer currently in use to one that specifically filters social media data (considering the textual characteristics of a social media message, including filtering out or standardizing hashtags, handles and emojis).
- The full text message could be displayed when one of the individual geolocation pins is selected by the user. This may help provide additional information aside from the most commonly used words in a cluster.
- Relevant search terms could be added on the input by a user or selected from a list of relevant terms. This could be used to customize the program for specific personnel types.
- A history could be recorded that would allow for playback of changes over time. The text data as well as the clustering information could be stored in a database for later retrieval. This would allow crisis management personnel to replay the changes in the visualization over

the window and at the speed they select. This could also be used for future analysis to see if the crisis could have been detected earlier or to see how exactly people were using social media to communicate with each other and emergency personnel during a crisis.

- With enough real-life crisis data captured, the program could be improved to generate alerts when indications of a crisis are present in the data. We currently do not have enough crisis data to train our model to detect the onset of a crisis: More data would be needed for the program to be able to accurately detect a crisis. Personnel verification would still be necessary, particularly given the possibility for events that occurred in one area to be discussed in some other areas where no active crisis is ongoing, such as the case of the 2014 Ebola crisis [8].
- Once a crisis is in progress, it is possible that the hashtags used most frequently could be followed for the affected area, which may improve the accuracy of the data for first responders.

We are currently working on a method to hierarchically group the pins from each cluster on the map, such that as we zoom out of the map, the cluster will be reduced to fewer pins and as we zoom in, these pins will split into more and more pins until the individual message level is reached. This may allow more clusters to be shown on the map at the same time without losing information due to too much clutter. This could also allow the user to see the number of messages in each automatically clustered group and to see exactly how many messages about a particular topic were sent from a particular region, without counting them manually.

There are clearly many options to be explored and a lot of improvements that can be made. As the end goal is a tool that provides the greatest value possible to emergency management agencies, receiving feedback from those who would use it and gaining the most benefit would be useful for future improvements. Capturing additional data that could be used for further study, rather than using data invented for a challenge, would also be beneficial to testing and improving the system to respond to live data. Providing a tool that would be used to save lives, reduce suffering, and provide a faster, clearer response to a crisis is the ultimate goal.

## REFERENCES


[1] Campus Alerts® Emergency Mass Notification System and Apps, www.campusalerts.com, accessed May 20, 2017.

[2] K. A. Lachlan, P. R. Spence, X. Lin, K. Najarian, and M. D. Greco, "Social Media and Crisis Management: CERC, Search Strategies, and Twitter Content," Computers in Human Behavior, vol. 54. pp. 647−652, 2016.

[3] J. Ao, P. Zhang, and Y. Cao, "Estimating the Locations of Emergency Events from Twitter Streams," Procedia Computer Science, vol. 31, pp. 731−739, 2014.

[4] J. Yin, A. Lamper, M. Cameron, B. Robinson, and R. Power, "Using Social Media to Enhance Emergency Situation Awareness," IEEE Intelligent Systems, vol. 27, no. 6, 2012, pp. 52−59.

[5] M. Bunz, "In Haiti Earthquake Coverage, Social Media Gives Victim a Voice," The Guardian, Jan. 14, 2010, www.theguardian.com/media/pda/2010/jan/14/socialnetworking-haiti

[6] N. Gleit, S. Zeng, and P. Cottle, "Introducing Safety Check," Oct. 15, 2014, newsroom.fb.com/news/2014/10/introducing-safety-check

[7] P. J. Rousseeuw, "Silhouettes: A Graphical Aid to the Interpretation and Validation of Cluster Analysis," Journal of Computational and Applied Mathematics, vol. 20, 1987, pp. 53−65.

[8] V. Luckerson, "Fear, Misinformation, and Social Media Complicate Ebola Fight," TIME, Oct. 8, 2014, time.com/3479254/ebola-social-media

[9] V. A. Schmidt and J. M. Binner, “A Semi-Automated Display for Geotagged Text,” City Evacuations: An Interdisciplinary Approach, (Eds.) J. Preston et al., Springer, 2014, pp. 107−116.

[10] X. Chen, G. Elmes, X. Ye, and J. Chang, “Implementing a Real-time Twitter-based System for Resource Dispatch in Disaster Management,” GeoJournal, vol. 81, no. 6, pp. 863−873, 2016.

[11] 2011 IEEE Conference on Visual Analytics Science and Technology (VAST) Mini Challenge 1 Dataset: Geospatial and Microblogging Characterization of an Epidemic Spread. Available at hcil2.cs.umd.edu/newvarepository/benchmarks.php#VAST2011

[12] Apache Kafka, kafka.apache.org/

[13] Apache Zookeeper, zookeeper.apache.org/

[14] Leaflet: An Open-source JavaScript Library for Mobile-friendly Interactive Maps, leafletjs.com/

[15] JQCloud, github.com/lucaong/jQCloud

[16] Express, expressjs.com.

[17] Node.js, nodejs.org/en/

[18] Jade, www.jadeworld.com/

[19] Y. Zhao and G. Karypis, “Comparison of Agglomerative and Partitional Document Clustering Algorithms," Technical Report# TR 02-014, Department of Computer Science, U. of Minnesota, 2002.

[20] C. Luo, Y. Li, and S. M. Chung, “Text Document Clustering Using Neighbors,” Data & Knowledge Engineering, vol. 68, no. 11, Elsevier

Science, 2009, pp. 1271–1288.