Mass amounts of data exist on social media networks, much of which has been used to analyze brand performance. As has been reported in the popular press, social media is increasingly being used to share information in times of social and political uprising. In this research, we have developed new methods for visualizing and characterizing the emergence of events using the data available on social networks. We have developed methods to cluster Twitter posts in real-time using a number of different criteria as well as a web service to facilitate creation of mashups using posts and clusters we have captured. As a demonstration, we have created an iPad application with custom visuals designed to characterize these events. Our research contributes new analysis methods along with usable tools to assist future work in this developing research area.
VISUALIZING AND CHARACTERIZING
REAL WORLD EVENTS ON TWITTER

A Thesis

Submitted to the
Faculty of Miami University
in partial fulfillment of
the requirements for the degree of
Master of Computer Science
Department of Computer Science and Software Engineering
by
David James Morell Jr.
Miami University
Oxford, Ohio
2012

Advisor__________________________
Dr. Jerry Gannod

Reader___________________________
Dr. Keith Frikken

Reader___________________________
Dr. James Kiper
### Contents

1 Introduction .......................................................................................................................... 1
   1.1 Contributions.................................................................................................................. 3
   1.2 Thesis Statement............................................................................................................ 3

2 Background and Related Work ......................................................................................... 4
   2.1 Clustering Approaches ................................................................................................. 4
   2.2 Streaming Algorithms for Clustering ........................................................................... 5
   2.3 Brand Intelligence Application .................................................................................... 5
   2.4 Prediction Market to Predict Oscar Winners ............................................................... 9
   2.5 Event Identification in Social Media .......................................................................... 10
   2.6 Twitriss ....................................................................................................................... 11

3 Approach ............................................................................................................................ 12
   3.1 Use Case ...................................................................................................................... 12
   3.2 Clustering Algorithm .................................................................................................. 13
   3.3 Workflow ..................................................................................................................... 17

4 Design and Implementation ............................................................................................... 19
   4.1 Overall Architecture ................................................................................................... 19
   4.2 Web Service ................................................................................................................ 21
   4.3 iPad Visualizations ...................................................................................................... 21
      4.3.1 Cluster Map .......................................................................................................... 22
      4.3.2 Post View ............................................................................................................. 23
      4.3.3 Advanced Clustering Settings ............................................................................ 26

5 Evaluation ........................................................................................................................... 29
   5.1 Montreal Riot .............................................................................................................. 29
   5.2 Cluster Accuracy ......................................................................................................... 30
   5.3 Clustering Parameter Trials ....................................................................................... 31

6 Conclusion .......................................................................................................................... 34
   6.1 Future Work ............................................................................................................... 34

References .............................................................................................................................. 36
List of Figures

Figure 1: Brand Intelligence Infrastructure ........................................... 6
Figure 2: Brand Intelligence Meter View ................................................. 7
Figure 3: Brand Intelligence Graph View ................................................. 8
Figure 4: Cluster Structure .................................................................. 14
Figure 5: Cluster Processor Sequence ..................................................... 14
Figure 6: Developing Event Flow ............................................................ 17
Figure 7: Overall Architecture ............................................................... 19
Figure 8: Cluster Map .......................................................................... 22
Figure 9: Cluster Preferences ................................................................. 23
Figure 10: Post View ............................................................................ 24
Figure 11: Social Graph ........................................................................ 26
Figure 12: Advanced Settings ................................................................. 27
Figure 13: Montreal Map Zoomed ............................................................ 29
Figure 14: Montreal Map Accuracy .......................................................... 30
Figure 15: Parameter Trials ................................................................. 31
Chapter 1

Introduction

With the number of people using social media continuing to grow, businesses, governments, and individuals are looking for ways to analyze social data to meet their needs. It seems that many significant societal events are preceded by changes online. In fact, authorities have begun to place partial blame to past events on social media [1]. One such case is the BART subway protest in 2011 [2]. In this event protesters organized online to rally against a recent shooting in the subway. Social media was such a factor in this situation that BART authorities shutdown cell service in the subway to interfere with the planning of the protest. In another case the London Riots are said to have begun with campaigns on Facebook, BlackBerry Messenger, and Twitter [3]. In a third case Osama Bin Laden’s death was first made public on Twitter. A man in Abbottabad, Pakistan live-tweeted the bangs he heard along with information about the helicopter overhead [4]. So whether social media is used to organize these events or whether it is the first place to detect them, it is clear that social media plays a key role when events are emerging.

Before we began this project we obtained some preliminary results. We conducted experiments on the BART subway protest [2] and the London Riots [5] and manually analyzed these events. We were limited to the amount of historical data we could collect by the API we used, but we were able to see
an increase in posts about these events when they occurred. This encouraged us to design a system to automatically monitor, store, and display the data surrounding events.

To further motivate our problem, on August 23rd 2011, the Intelligence Advanced Research Projects Activity (IARPA) released a very relevant solicitation [6]. They were looking for methods to anticipate or detect significant societal events in advance. Their synopsis states that there are many detectable changes in communication that are observable before a crisis, riot, or instability. They will be evaluating the technical innovation on the basis of the warnings it can deliver about real-world events. Finding this solicitation helps to motivate our thesis because our work could directly contribute to such a solution. IARPA is a large government organization with the motto “Be The Future,” and is focusing resources on this problem very recently. This reinforces the value of our research area.

Our project can be very useful for governments or leaders and begin to create a system to detect, predict, and visualize events. These predictions could allow people to prepare for and even take steps to prevent something bad from happening. In the case of a peaceful situation, businesses or organizations could use that prediction to contribute to the event, for marketing or other purposes. In both cases a tool that can help visualize and characterize events is useful and can help contribute to the future work of predicting events.

We are proposing new analysis methods to characterize the emergence of events before they occur. This task involves processing real-time data from Twitter. We have determined features of the data to monitor and cluster posts. We group posts based on their location, timestamp and text similarity. Connections are established between users based on retweets, user mentions,
and replies to help visualize the social network graph. Whether we are visualizing a dangerous flash mob, a riot occurring, or a peaceful event, our system can be used as a tool to find the cues and patterns that define when an event is imminent. It can also provide insight into how the event is progressing, and whether related social media posts are increasing or dying down.

1.1 Contributions
Our thesis research has provided many contributions to both the research community and organizations. Our research has provided a new analysis approach for social media data in order to cluster posts related to real-time events. We have also provided new and innovative tools that demonstrate our approach, including an iPad application for visualization of our systems output. We believe we have provided original methods and tools to perform a task that has not been addressed in past research.

1.2 Thesis Statement
We hypothesize that given the data on social media networks, we can cluster, characterize, and visualize real world events in ways that provide insight into how the events are emerging and progressing.
Chapter 2

Background and Related Work

2.1 Clustering Approaches

Many clustering approaches exist to help make sense out of large datasets. Jain, Murty, and Flynn provide a good survey of data clustering techniques in [7]. The paper provides good background for designing a clustering algorithm. It states that clusters can be represented in a variety of ways: the centroid values of the cluster features can be stored, a classification tree can be created, or conjunctive logic expressions can be used (e.g. \([X > 3][Y < 6]\)). It also outlines the key steps of clustering: to determine a similarity computation, create a grouping process, and create a cluster representation. A classification tree does not make much sense for our system since each cluster is independent of another. Conjunctive logic expressions would also not be the best representation for our system, because clusters have overlapping features. For example, time is one of our features and two clusters may have posts from the same time range. Because of this, our system uses the centroid method of representing clusters. This allows us to easily keep ‘average’ cluster values to compare the similarity of new posts. Section 2.4 discusses a clustering algorithm used specifically for events on social media, and Section 3.2 details our similarity computation and grouping process.
2.2 Streaming Algorithms for Clustering

O’Callaghan, Mishra, Meyerson, and Guha present techniques for clustering streaming datasets in [8]. Streaming data algorithms have becoming increasingly important as datasets become larger, and as streaming data becomes more available. Common clustering approaches rely on static datasets that they can either store in memory or iterate over many times. Streaming algorithms are useful when a continuous stream of data is arriving and can only be processed when it arrives with minimal processing. These algorithms can also be applied to datasets that are so large on disk that the only feasible way to handle them is to process them like a stream.

The main concept behind using this approach for streaming is to fully process each item as it arrives, into a cluster representation. For clustering based on one metric this means clustering the first k items on that metric, then purging memory of everything except the mean cluster values and their weights. When the next k items arrive, they are clustered with the mean values in memory. The process repeats itself so that the only thing stored in memory is the mean values of the clusters, but not the data items themselves. The algorithm we have designed for clustering follows a similar approach. One difference is we have more than one metric to base our clusters on. Because of this we calculate an aggregate similarity between the incoming items and the existing clusters.

2.3 Brand Intelligence Application

In previous work we developed a project that was focused on determining sentiment about brands or topics on social media [9]. Our goal was to
accurately represent this sentiment about brands for business professionals, and to visualize how these brands compared to one another. Many of these ideas originated in a capstone course with about 15 students. In an independent research project I continued working and developing the concepts with Robert Howard. The product itself consists of a server that consumed social media data on the web and categorized it based on positive/negative/neutral sentiment. It included an iPad application that visualized the data from the server.

As can be seen in Figure 1, a FeedReader would receive request posts from the social APIs and ask the SentimentClassifier to determine the sentiment of the text. Then all the data about the posts and sentiment was stored in a database. When the iPad application was launched it would request this data from the Servlet. The Servlet would send custom XML to the iPad application with the formatted data from the database that it requested. This research
project explored social media feeds as sources of data that could be analyzed in a variety of ways.

Since visualization is an important component of our thesis, we will discuss that visuals used in this past work. Figure 2 shows the main view of the application. Universities are considered the brands in this scenario and are listed on the left hand side of the capsules. Each capsule represents the positive/negative/neutral share of posts about that brand on the corresponding social network. The pie chart represents the total share of posts about each brand, as indicated by the color next to each brands name. The last element on this screen is the word cloud, which shows the most prominent words for all the posts about these brands. The general idea of this screen was to allow a brand manager to assess the quantity of the posts
compared to competitors, and also the sentiment of those posts. The word cloud also provides a way to see if anything new or interesting is being said for this product category. The dashboard nature of this screen allows for all this information to be consumed at once, without the need to navigate around an application.

A second screen of this application maintains the pie chart and word cloud, a graph instead of the capsules. This graph allows the user to choose two brands and directly compare the quantity of positive/negative/neutral posts over time. The color of the line represents which brand was selected. While this project was more specifically focused on brands, it sparked an interest in our new thesis project.

The research we are proposing may sound similar on the surface, but it is quite different in the specific research problems we have addressed
including the implementation details. The past project did minimal processing; it focused on collecting data and determined the sentiment. Our thesis work collects data from the same types of sources, but is faced with many different challenges. We have designed custom data structures, clustering methods, and analysis approaches for relating posts, and determining how to visualize them most effectively. Also while both projects use the iPad to display the data, the task of visualizing them was entirely different. The past project was visualizing the sentiment about a specific brand and how it compared to other brands. In contrast, our thesis work has visualized the intensity of posts that may be indicating a looming event, the locations of these posts, and the social graph relations of the authors. We have explored new techniques to display the data in the most useful and insightful ways.

2.4 Prediction Market to Predict Oscar Winners

Bothos, Apostolou, and Mentzas [10] proposed a method for predicting the winners of the Oscars using only data collected from social media. They defined agents to monitor on each network, one for IMDb, Twitter, Flixster, etc. They then defined a market where the agents could bid on their predictions. A very high price in the prediction market would indicate a prediction for a certain result, because it shows that many agents have bid on that prediction. And since their approach is modeled with money on the market, when an agent gets a prediction correct, it will gain money. More money for an agent gives it more power to bid and more weight in future decisions. Each agent processes the data in any actual ratings from a site (e.g. 3 stars) and also processes the text in the comment using sentiment analysis. Therefore the system does a lot of what we are trying to accomplish,
it searches social media for data and processes that data along with machine learning to generate a prediction. One distinction between our system and theirs is that they are predicting the outcome of an event that is planned, and our goal is to characterize events that emerge without notice. Their system essentially accumulates user opinions about certain topics to guess what will win an award. Our system will need to depict the characteristics of an emerging event, where it is occurring, the intensity of the event, and who are the key contributors. We also are not looking at data that has star ratings or can be objectively categorized as winner or loser. The paper on this topic is one of the only we found about event prediction on social media, and has recently been accepted for publication in IEEE Intelligent Systems. This indicates that real-time event analysis and prediction has not been successfully done before and is a current topic of interest.

2.5 Event Identification in Social Media

Becker, Naaman, and Gravano present some very promising techniques in [11]. The goal of their paper is to provide local event browsing and search. Their system is based on geocoded data, timestamps, and the text of the data they collected. They have defined the problem as a clustering problem, and create clusters based on the features of the data they are examining. Their system clusters posts based on how close they are in time, location, and semantic similarity of their text. By defining these similarity measures between features they are able to group posts which are most likely relating to the same event. They also use machine learning to train the system and determine the threshold levels for each cluster. These techniques and the threshold approach can be applied to our problem, and many of our algorithms are derived closely from their work. However, our data and
features are updated in real-time from a streaming Twitter feed. Our algorithms and data structures are designed to work in this real-time environment. Also because of this we do not have any historical data to train a machine learner on, so some fundamental modifications have been necessary. In a real-time system it is much more difficult to perform the supervised learning that they perform. Still, they have an effective framework that we have learned from and adapted to our specific problem.

2.6 Twitris

At Wright State University Jadhav et al. have designed a tweet mining system called Twitris [12]. Their application gathers posts from Twitter and performs semantic analysis to extrapolate knowledge about events and entities. Using the semantic web, it identifies and tags known entities in the posts it collects. With these tags they hope to be able to answer questions like “Where is Obama now?” We became aware of this research late in our project, so it does not have a large impact on the design of our system. However, it is related in the sense that it gathers event centric data from Twitter, and unique it the fact that it applies semantic web techniques to the problem.
Chapter 3

Approach

3.1 Use Case

Our system has been designed to detect and visualizing emerging events. One such use case for the system would be when an event occurs or begins to develop. Let’s say that people start tweeting about the event, either to get others to join, or just to inform others what is occurring. When many people start posting about this event, we want our system to detect that those posts are related. This involves comparing the posts based on a few characteristics such as: when they were posted, where they were posted from, and what they are saying. In this case, the users are most likely posting from a very similar location, around the same time, and mentioning some keywords that are distinct from the rest of the posts in that area. It is clear that these three features are important, but we also want our system to be flexible enough to pick up some posts that aren’t similar in one of those dimensions. Some events can occur over a few days or months, some people may be posting about an event but aren’t physically present. These are points to consider and ideally our system would be able to capture all of these cases.

Once our system captures these events it will visualize them in the iPad application. The main screen of the application shows a map of where the most recent clusters are located. Let’s say the user is a law enforcement
officer and finds a cluster nearby. They tap on it to drill down into more information. They will then be presented by three views, a map, graph, and histogram. The map shows them where each individual post is coming from. This could be useful to see specifically where active areas of the event are located, or just to get a better idea of the location and distribution of the event. Each post can be viewed to determine when it was posted, who the author was, and the text it contains. Posts can be individually analyzed to get a better idea of exactly what is going on. Secondly, the histogram shows the rate of posts over the lifetime of the event. This allows the user to see if the event is getting larger, or dying down. If the event continues to grow the officer may want to call in backup, if it is dying down maybe that wouldn't be necessary. Lastly the social network graph is displayed. This shows how each user is connected to other users. It is useful for seeing who is most involved in the event. The officer could work on contacting some of the most connected users to learn more about the event or try to prevent it from getting out of hand.

At this point in our research it is not possible for the officer to define specific keywords to follow, we are still looking for general events around the world. It may be the case that the officer would like to specify new words to monitor after seeing the topic of the event. This could be implemented but would make the application much more focused on one area, or one event, while the current architecture provides a more general landscape of events all over.

### 3.2 Clustering Algorithm
Figure 4: Cluster Structure

Figure 4 shows the logical structure of a cluster in our system. This structure represents one cluster or event in our system. The ‘terms’ variable is a map containing terms that are frequent in the cluster as keys, with the corresponding term frequency as the value. As new posts are clustered they are compared to all clusters in memory.

Figure 5: Cluster Processor Sequence

Figure 5 shows a post arriving and being compared to all clusters in memory. This comparison is not trivial and required a custom algorithm to be designed that would work in this real-time system and still maintain the integrity of the clusters. One problem is that our dataset is not finite, most clustering techniques require all the data to be present before clustering. It would not be feasible for us to collect continuously and then cluster all the posts occasionally, because of the volume of posts that are arriving, and due
to the fact that our cluster data would never truly be real-time. For these reasons each post must be clustered when it arrives.

Similarity is measured between a post and a cluster using a few metrics. One is the location of the post compared to the location of the centroid of the cluster. The distance similarity is calculated as $1 - \frac{H(p\text{Loc}, c\text{Loc})}{\text{maxKilometers}}$ where $p\text{Loc}$ and $c\text{Loc}$ are the post and cluster latitudes and longitudes. $\text{MaxKilometers}$ is the farthest distance we consider similar, if posts are farther than this distance then 0 is returned. $H()$ is the Haversine distance [15]. This measure is an accepted geographic distance measure that computes the ground distance between two latitude and longitude coordinates. All of our similarity metrics are on a scale from 0 to 1. Identical locations would receive a similarity of 1, and two locations across the globe would receive a similarity of 0. Locations in between are scaled between 0 and 1. Another metric used is time similarity. The concept behind this metric is that posts are more likely to be about the same event if they are closer together in time. Posts that are more than one month apart will receive a 0, and posts tweeted at the same time will receive a 1. The times in between are scaled between 0 and 1. The final metric used to cluster posts is the term similarity. First the post is tokenized and normalized, most punctuation is removed, and it is put into lowercase. Links are not modified because changing the case of short links will change their destination. For example, ‘http://bit.ly/2V6CFi’ is not equivalent to ‘http://bit.ly/2v6cfi’. A list of common and non-meaningful words is used to exclude as much noise as possible. Over-normalization does not improve performance of such a measure. In [16], Rosa et al. determined that beyond whitespace tokenizing, lowercasing, and excluding certain terms, more normalization does not improve performance in a significant way. Each tokenized term is then compared against the cluster.
One terms similarity is equal to the frequency of that term in the cluster divided by the number of posts in that cluster. This ensures that cluster size does not affect the similarity. A posts similarity is defined as the average similarity of all its terms.

Once the geographic similarity, term similarity, and time similarity are calculated they are aggregated based on weights. These weights differ based on whether a post had an exact location, whether it was estimated from the author’s profile, or whether it was not available. Modifying these weights changes how posts are clustered and can result in clusters that are tighter or broader in time, geographic, or term similarity.

The new post now has an aggregated similarity for each cluster. The cluster that has the maximum similarity is where the post will end up. In the case that this maximum similarity does not meet our minimum cluster threshold, a new cluster will be created with only that post. This ensures that when a post is not strongly related to any current cluster it can form a new cluster. In either case the cluster that receives the post updates its centroid data and updates the social graph. Its social graph is updated if this post was retweeted from another post in the cluster, mentions another post in the cluster, or replies to another post in the cluster. With the current limitations to the Twitter API and the design of our system it is not possible to add a graph edge solely based on Twitter friendship. This information is not transmitted to us by the API and would require additional API calls for every post we receive. This would greatly exceed the API limits in place by Twitter.

To manage memory and noise, clusters are periodically purged from the system. Thresholds are in place that force old or small clusters to be deleted from memory. The original posts are not deleted from our database, but the cluster structures are removed. Currently clusters are considered old after 48
hours, and considered small with less than 15 posts. If a cluster is old and small it will be cleared from memory. This process of clearing old clusters occurs every 5 hours. This greatly reduces the overhead of clustering new posts and storing clusters in memory that are not significant.

### 3.3 Workflow

Social media is a great place for opinion mining about brands or topics [13], but we haven't seen its content monitored to characterize real world events as they occur. Since many events have been attributed to either beginning or evolving on Twitter, our thesis focuses on analyzing and visualizing this data. The event identification described in Section 2.4 provided us with a basis for clustering related posts. Our goal was to group posts that are about the same real world event. We used information available on Twitter to group these posts based on a variety of features such as time, location, and text content.

![Diagram of Developing Event Flow](image)

**Figure 6: Developing Event Flow**

Figure 6 shows the intended flow of the system. It can begin with either an event occurring, or event planning starting online. In both cases these tweets arrive at our server from Twitter. The posts are clustered based on similarity. After establishing clusters of posts, we provide an interface for
learning and making decisions about the events we identify. Our goal for this project was to create the framework necessary to gather data from Twitter, cluster the posts, and visualize them in real-time. Our focus was not on finding perfect clustering thresholds or using machine learning to identify events, but instead to demonstrate the system that can effectively visualize these clusters, and be easily adjusted to use any clustering parameters.

In order to visualize the clusters, we must show key information about the clusters. Some key facts are the location of the cluster, number of posts, rate of posts per hour, and connections between users in that cluster. The location of the cluster may determine the importance to a user. A specific user may only be interested in clusters nearby, whereas a government may be interested in large events anywhere. The rate of posts can help indicate whether a cluster is growing or shrinking, and how fast. The social network graph demonstrates the connections between authors, and which have the most impact on the event. It may be helpful to contact these authors or monitor their future activity. The tools we have created visualizes these aspects of the event and allow for the user to customize their views based on where they would like to focus their attention. Some key visuals that we include are a map, histogram of the post activity, and a social network graph. Details of these view and their customizations can be found in the following section.
Chapter 4

Design and Implementation

We have designed and implemented a complex framework that handles the workflow from receiving the posts, to analyzing and clustering, to visualizing. Our system was designed to be flexible, meaning that the parameter values we discuss can be further optimized to increase the accuracy of our system.

4.1 Overall Architecture

![Diagram of Overall Architecture]

*Figure 7: Overall Architecture*
The overall architecture of the system can be seen in Figure 7. On startup our system initializes a connection with the Twitter Streaming API [14]. This API is provided by Twitter and allows real-time streaming of posts based on a keyword query. For our system we filter on keywords related to events that might occur. For example: protest, boycott, occupy, riot, gather, mob. Twitter does not allow us to receive all posts, but for our purpose the filtering actually assists our system in identifying events. The keywords we chose were based on data we collected about the BART subway protests, and the London Riots.

When the StreamReader receives a post from Twitter, it stores that post and all relevant information in our Database. At this time it extracts the ‘location’ field from the authors profile and geocodes it using Bing Maps Geocode Service. This latitude and longitude is then stored in our database as an estimated location when an author does not geo-tag their post with GPS location data. It should be noted that some users do not indicate a location in their profile and others do not refer to a legitimate location, but these coordinates still serve as a good estimate when the geo-tag is not present. Overall the system does not rely on every location being exact, but that we have the best accuracy possible when grouping and displaying posts. Locations from GPS data are given more weight than locations obtained through the author’s profile.

After storing the post and related information in our database, the StreamReader sends the data to the ClusterProcessor as well. The ClusterProcessor clusters posts in real-time to determine if a new post belongs in a cluster already in memory or not. Full description of the ClusterProcessor can be found in Section 3.2.
On the other end of the system is an iPad application that visualizes the data we have collected and processed. This contacts a custom web service we designed to retrieve this data, and displays the visuals we have designed.

4.2 Web Service
A JSON web service has been designed and implemented for this project that is loosely coupled to the iPad application. This is an important factor that maintains flexibility of our system, allowing applications to be designed on various platforms without major modification of the server and clustering system. The service supports operations to get the clusters, get the social network graph for a specific cluster, and to get the posts for a specific cluster. When the list of clusters is requested, the caller can specify the number of clusters it would like to receive, and the service will return a list of clusters. Each cluster will include the average location, average time of posts, number of posts, and most frequent terms in that cluster.

When more data is required about one specific cluster, the caller can request the posts from just that cluster. This will return all the posts in that cluster including all related data about the post and author. If the social network graph is needed, the service can send the graph edges within a cluster when requested.

4.3 iPad Visualizations
We chose to visualize our data on a mobile device because we believe it is the most flexible. To analyze events in real-time, as they emerge, it makes sense to have a tablet. Government officials or law enforcement officers could use it more easily than a PC based application. In fact anyone who is monitoring
such events and is not at a desk would benefit from having this application on a tablet. With that being said, the backend system and web service are platform independent of the visualizations. Therefore any platform can be used to create new or supplemental visuals displaying the same computed data.

4.3.1 Cluster Map

Figure 8: Cluster Map

Figure 8 shows the screen that is displayed when the application is launched. This screen is designed to show which clusters are present in the database. The map can be easily scrolled or zoomed for different perspectives or to view clusters in another part of the world. The legend in the bottom left of the
screen shows that the color of the pin indicates the size of the cluster. Having color differences allows for the user to quickly see which clusters are larger than others. If a cluster is tapped, a callout view displays some of the top terms in that cluster and the exact cluster size, as seen in Figure 8. When the blue arrow is pressed the screen will transition into a view that shows details about the posts in that specific cluster.

![Cluster Preferences](image)

**Figure 9: Cluster Preferences**

The clusters view can be customized by clicking the ‘Info’ button in the top right of the application. This allows for the user to determine which clusters they are interested in seeing. The default values are shown in Figure 9. The user has a choice of showing the most recent clusters, or the largest clusters. The most recent clusters are those with a centroid time closest to the current time. The largest clusters are simply the clusters with the most posts regardless of their centroid time. The number of clusters to display can also be specified, which limits the view to that number of clusters. The minimum cluster size is important because if ‘Most Recent’ is selected with a low minimum cluster size, then small, insignificant clusters will be displayed.

### 4.3.2 Post View
The post view is reached by clicking the blue arrow on a cluster. It can be seen in Figure 10. The map has shrunk to half the screen size and now displays the posts in the selected cluster. As can be seen in the legend on the bottom left, red pins indicate an exact GPS location was present, and purple indicates that the location was estimated from the author’s profile. Tapping on a pin will show the author’s username on Twitter and the full text of the post.

The top right displays a social network graph. Connections in this graph are inferred from user mentions, retweets, and replies. The username is located in the graph as well to identify how authors interact with each other. Authors with more followers have larger nodes in this graph. Color is used to indicate how many posts the author has posted for the life of their
account, red represents more than 1 millions and green represents more than 10,000 posts. Blue represent less than 10,000 posts. By combining this information into one social graph this allows a user of our application to see which authors are having the most affect on a cluster. It could potentially show the central entity or key contributors. It should be noted that very advanced algorithms exist for determining user influence on social networks. Romero, Galuba, Asur, and Huberman propose one such technique in [17].

Our influence calculations are similar to those they describe, which are based on how often the tweets are retweeted or forwarded. Our visuals display this in the graph edges.

On the top right is a histogram showing the rate of posts throughout the life of the cluster. This allows a user to determine whether a cluster is increasing or decreasing in activity. It also allows for a user to see how the event progressed historically. A slider beneath the bar chart can be slide back to change the posts that are displayed in the map and social graph. This is one way that a user can see what an event looked like earlier in time, and how it progressed to the present time.

The map can also be zoomed and moved to eliminate clusters from the social network graph. When a post goes off the visible map, it is also removed from the social network graph. So this view allows for the event to be examined in the time and space dimensions.
4.3.3 Advanced Clustering Settings

The small grey arrows in the bottom corners of the social network graph and the histogram allow for either of these visuals to be viewed in full screen. As can be seen in Figure 11, this is especially helpful when a social graph is crowded or complex, because the extra screen space allows it to be examined more closely.

The small grey arrow also works to return the view to its original size. Once out of full screen, there is a ‘Clusters’ button in the top left of the toolbar to leave this cluster and return the view of all the clusters.
In order to facilitate testing of the clustering algorithm itself, we designed advanced configurable settings. Figure 12 shows the user interface for changing these settings. It is composed of a grid of parameter boxes on top of a list view of previously used settings. The list contains a special value ‘Live Clustering’. When that is selected the system will display the clustering of live tweets, essentially how it was designed to operate. The parameter boxes will display the current values in use and are not editable. As you can see some items in the list are date ranges. These represent a clustering of posts that were posted in that date range. Selecting one of those ranges will change the parameter boxes to represent the values used for that manual clustering process. Finally, selecting the option ‘New Cluster Params’ will make the boxes editable and the user can enter a date range and clustering values to perform a manual re-clustering of posts in that date range.

The grid of boxes that is shaded in purple represents the clustering weights for each similarity. The location value of a post can potentially be ‘Exact’, ‘Estimated’, or ‘None’. In each case a user may want to weigh these similarities differently. So horizontally these should sum to 1, and represent the weight of time, terms, and geographic similarity for a post. The
‘MinSimilarity’ parameter ensures that a post is not grouped with a cluster if it has a very low similarity. So if a post has a most-similar cluster that is still below this threshold, a brand new cluster will be created for that post instead. The ‘maxMinutes’ and ‘maxKilometers’ values are used when calculating time and geographic similarity. As previously discussed these metrics are scaled from 0 to 1 and these parameters will define when to determine similarity of 0. In other words any post more than ‘maxMinutes’ away from the cluster will get a 0 for its time similarity measure. Likewise any post more than ‘maxKilometers’ away from the cluster centroid location will get a 0 for its geographic similarity measure.

These settings were not designed for the end user since they require the server to perform the time consuming and costly task of re-clustering, in addition to the live clustering it is always performing. These settings are primarily for testing purposes. One example use could be externally determining when an event occurred, and then trying to cluster that date range with new parameters. This allows for comparison of the results and fine-tuning of the clustering parameters. If better values are found the system could be manually reconfigured to use those parameters for the live clustering.
Chapter 5

Evaluation

5.1 Montreal Riot

On Thursday March 15, 2012 a protest occurred in Montreal, Canada [18]. The protesters were marching for anti-police brutality. The event quickly turned into riot with over 1000 people showing up, over 200 people arrested, and 10 police officers injured. We were first made aware of this event through our application. It was very exciting to see the system working for the exact purpose it was created. Figure 12 shows a zoomed in map of this cluster in our application. For clarity it should be noted that there was a lot of noise in this cluster as well in other areas, but clearly a tight bunch was in Montreal.
and related to this event. It is also interesting to notice that most of the posts in this area are represented by red pins, meaning they were geo-tagged with GPS location. This reinforces our algorithm’s increased weight for exact locations versus estimated locations from the author’s profile. It could even indicate that GPS locations should be given an even higher weight than we used, or that clusters with more GPS locations should be displayed more prominently.

5.2 Cluster Accuracy

Figure 14 shows a zoomed out view of the same Montreal event. The screenshot has been annotated to show which posts actually are relevant to the event. Relevancy was determined manually by checking the text of each post. The red polygon encloses the posts that are mainly about the Montreal event. The orange circle shows where the event occurred. The reason that other posts down near New York were included in this cluster is because they are nearby the location and also had text referring to riots or mobs. Unfortunately they do not appear to be talking specifically about the Montreal riots. It remains an open problem of how to include posts in the
entire polygon, without including other posts very close to the orange circle that are not related. This figure helps to demonstrate how related posts can be in odd patterns, there are related posts far away and non-related posts nearby. When these groups are similar enough in text, location, and posted around the same times, other advanced methods need to be discovered to separate these posts. Alternatively, it may be acceptable to include this noise and leave it to the user to decipher by exploring the event, but the more accuracy that can be achieved the better characterization we can display overall.

5.3 Clustering Parameter Trials

![Figure 15: Parameter Trials](image)

In order to fully evaluate the system we must consider how the clustering algorithm is performing. As mentioned previously, we built in some advanced settings to tweak the parameters and weights used in clustering. We have experimented with these values and Figure 15 shows a table of our results. All of these results were run over the same data that originally found the Montreal Riots. The posts clustered in this table ranged from March 15th
2012 at 2pm until the next morning at 6am. Gray shading indicated changes from the default values. The goal in these trials was to capture the one night of rioting. The total number of posts we have from that time period is around 56,100.

The trial run with the default values had 58% relevant posts in the Montreal cluster. This is not a bad starting point; many of the trials performed much worse.

The next three trials each focus on one dimension: time, terms, or geo. Each trial puts more weight in one of those dimensions. Notice that the number of clusters produced by each of these trials varies greatly.

High Time Weights produced a very small number of clusters. This is likely because the time range we clustered over was small, so by weighing time higher, more posts were clustered that potentially should not have been. We can tell something like this happened because the Montreal cluster completely disappeared. These posts were most likely clustered with posts more similar in time, than in location or terms.

High Terms Weights produced a very large number of clusters. This indicates there were a wide variety of terms, and when not considering location or time as much, the system just created new clusters when the terms did not match strongly. There are two numbers in the last three columns because there existed two clusters near Montreal that contained related posts. The average percent relevancy for this trial was 39.5%, which puts it in third place overall. It is problematic that these two clusters are not combined as one.

High Location Weights also produced two separate clusters, but their average relevancy was only 25%. It seems that if you put too much weight in one dimension, then the results suffer. This makes sense, because one
dimension itself does not determine that two posts are similar. Just because they both are in the same location does not mean they are related.

The last two trials were quite interesting, but still follow intuition. When the ‘minSimilarity’ was set low, fewer clusters were produced and the quality diminished. This is because a low value here allows for posts to be clustered even when they aren’t that similar. A very interesting and potentially useful trial was the high ‘minSimilarity’ trial. In this trial abnormally high relevancy was obtained. Once again there were multiple clusters near Montreal, but on average they contained 77% relevant posts. This is the highest relevancy we have seen in any of the trials. It appears that improvements can be made to the clustering algorithm, and one of those may be to increase the ‘minSimilarity’ value. The reasoning behind this is that a higher value only allows posts to be clustered if we are confident they are similar. So by nature we would expect more similar posts in these clusters. Though the optimal parameters would provide high relevancy, but not three different clusters for the Montreal Riots.
Chapter 6

Conclusion

Our research began with hopes of characterizing and visualizing emerging event on Twitter. In the process of studying events on Twitter we have built a system to cluster, analyze and store posts, relationships, and events. Our research has provided a usable tool to explore events that have appeared online, and to learn about these events by displaying key information about each. The system we have designed is flexible and customizable so that future work can modify and improve upon it. Our evaluation shows that the system is working as expected and can detect events that the user would otherwise be unaware of. It also showed that there is room for improvement with future work.

6.1 Future Work

Our work has laid the framework to solve a much larger problem. It serves as the first step towards a better understanding of events that emerge online. Future work can be done to predict events before they occur. Our system can be used to analyze and study events that emerge online, and even as a platform for visualizing event predictions. The prediction component is not trivial, it would require in depth analysis, and most likely machine-learning
techniques to determine when an event is imminent. As discussed earlier, this is an area of focus for the government, and would be valuable to demonstrate.

A second area that future work could focus on is perfecting the clustering thresholds of our system. This task would also likely require machine-learning techniques. Our system could be easily tweaked to use different threshold parameters, and the visuals would be a great indicator as to the quality of those parameters. Finding the optimal thresholds would greatly improve the effectiveness and accuracy of the visuals we have created. Research could be done to determine whether very tightly bound clusters provide more accurate visuals, or whether broader clusters allow for more subtle changes to be seen. Alternatively, completely different clustering algorithms can be tested with this system. The entire framework is present to visualize and test the functionality of other algorithms compared to the one we have implemented.

One such modification could be clustering the clusters that are determined in the first pass. This would be an iterative approach that would aim to solve the problem of multiple small clusters that really should be one cluster. In our evaluation when we found the highest relevancies, we also had our clusters broken up into smaller ones. So this approach may be able to take advantage of the relevancy and still only obtain one cluster per event. In any case it is important to determine what the user prefers to see, and what captures the best representation of what is truly happening.
References

   http://www.iarpa.gov/solicitations_osi.html


