Chapter 1

Knowledge Discovery and Information Extraction on the Open Internet Using MATLAB and Amazon Web Services (AWS)

Kelly Bennett and James Robertson

Additional information is available at the end of the chapter

http://dx.doi.org/10.5772/58895

1. Introduction

The popularity of social networking has allowed access to staggering amounts of unique data, which has created new possibilities for data analysis and exploitation. Such data has proven useful in marketing, decision making, destabilizing terrorist networks, behavior evolution, and determining future social trends [1].

Increased usage of social networking sites has also been observed during events related to natural disasters; significant political, sporting, and social events; and other crises. Twitter users provide status updates through tweets. Since tweets are typically short and always less than 140 characters, they may need to undergo additional analysis to provide contextual clues. Applying traditional natural language processing algorithms on such data is challenging. Before making use of this data, it must first be extracted, processed and analyzed appropriately using certain algorithms and theories.

According to a 2011 study from the International Data Corporation (IDC)–a marketing firm specializing in information technology and other consumer technologies–unstructured data, that is, data that does not have a pre-defined data model or is not organized in a pre-defined manner, is growing at a faster rate than structured data. Within the next decade, unstructured data will account for 90 percent of all data created.

A large driving factor in the increase in unstructured data is social networking data, such as tweets. It is estimated that more than 80 percent of all potentially useful data is unstructured [2].

The success of businesses in the coming decade will likely rely on their ability to successfully analyze data from social networks.

In addition to the increase in social networking, cloud computing technologies continue to increase in popularity and provide end users the ability to quickly spin up powerful servers with analytical capabilities, making analysis of large amounts of data more affordable and practical than in previous decades.

Main topics addressed in this chapter include:

- Explore viable approaches for extracting social networking data sets using tools compatible with MATLAB[1] and cloud technologies and infrastructures.
- Use existing data mining and statistical tools within MATLAB to conduct analysis on social networking site data.
- Discuss potential cost savings and document approaches for implementing social networking site data analysis via the cloud through vendors such as Amazon Web Services (AWS).
- Provide a tutorial on the use of MATLAB tools to analyze unstructured data with an emphasis on social networking data.
- Provide an overview and example on the use of MATLAB in a commercial cloud environment, such as AWS.

2. Previous work

Muhammad Mahbubur Rahman, author of "Mining Social Data to Extract Intellectual Knowledge," was able to extract data from Facebook using various data mining techniques. He first determined the most frequent terms that were being used, such as birthday, about me, gender, and music, and then created a database using these terms and the appropriate data types. [1] From the "about me" information, he was able to use an Euclidian distance formula to put each user into a certain "class", for example, aggressive, dishonest, romantic, eager to learn, or lazy. Grouping classes by age and gender creates a method of comparing various attributes by age and gender. This type of research demonstrates the ability to use data mining techniques to provide intellectual knowledge that can be used to possibility predict human behavior, provide insight into human decision making, or provide a method of determining anomalous events.

Data mining has proven useful in Twitter as well, with much research performed into detecting trends and bursty keywords. One has to pay attention to some abnormalities with Twitter, such as many people use smart phones to tweet and it is common for people to make mistakes while typing on smart phones. One would have to account for misspellings and merge them

1 Registered trademark of Mathworks Corporation.

into one variable so they are not separated and lost. In addition, some phrases are commonly abbreviated, such as "NYC" for "New York City," or reduced, such as "Vegas" for "Las Vegas." A study, using algorithms to account for these abnormalities, was performed that determined trendy and bursty keywords on Twitter with 92% accuracy when compared with Google trends [3].

Karandikar [4] described an approach to determine the most suitable topic model to cluster tweets by analyzing the effect of change in topic model parameters such as training data size and type and the number of topics on its clustering performance. The model was able to cluster tweets, using this approach, with an accuracy of greater than 64 percent for the two specific events.

Perera [5] developed a software architecture using a Twitter application program interface (API) to collect tweets sent to specific users. The extracted data were processed to characterize the inter-arrival times between tweets and the number of re-tweets. Analysis revealed that the arrival process of new tweets to a user can be modeled as a Poisson process while the number of retweets follows a geometric distribution.

Turner and Malleson [6] presented an exploratory geographical analysis of a sample of Twitter post data from June 2011 to March 2012 in the city of Leeds, England. Geographical cluster detection methods were used in combination with text mining techniques to identify patterns. Preliminary results suggested that the data could be used as a means of exploring daily spatial-temporal behavior.

The following presents insight in to the MATLAB toolboxes that can be used in detecting anomalies, that is, deviation from the normal pattern of life.

3. MATLAB toolboxes and classes for anomaly detection

MATLAB™ is a high-level language and interactive environment for numerical computation, visualization, and programming. MATLAB's product family includes numerous toolboxes for parallel computing, math, statistics and optimization, control system design and analysis, signal processing and communication, image processing and computer vision, test and measurement, and other areas [7]. A key feature of these products is the ability to research a number of algorithms quickly without having to design and code the algorithm.

MATLAB's interactive environment allows end users to quickly prototype their own algorithms if existing toolboxes do not provide the desired functionality. A well-established community of users supports the exchange of algorithms for those wishing to share their research and prototypes.

This research took advantage of the MATLAB community and the existing toolboxes. From the file exchange, Vladimir Bondarenko's contribution of the class to the Twitter REST API v1.1 was used to interface with Twitter to search for and extract tweets for specific topics and geographic regions [8]. Existing MATLAB Statistics and Neural Network Toolboxes were also

used in this research to provide statistical algorithms and unsupervised learning methods, such as cluster analysis, for exploring data to discover hidden patterns and groupings in the data.

3.1. TWITTY — Twitter REST API interface

Twitty is a useful interface that runs within MATLAB for communicating with Twitter. Methods used in Twitty are essentially wrapper functions that call the Twitter API [9]. The API caller function, callTwitterAPI(), does the main work.

Key steps to successfully using the Twitter API include obtaining Twitter credentials and using JavaScript Object Notation (JSON) parsers. The MATLAB file exchange provides a JSON parser developed by Joel Feenstra [10]. The JSON parser parses a JSON string and returns a MATLAB cell array with the parsed data. JSON objects are converted to structures and JSON arrays are converted to cell arrays. Twitter credentials are easily created by registering at the Twitter site, creating an application, and retrieving consumer and access keys. These keys are required for running Twitty and include specific values for:

- ConsumerKey
- ConsumerSecret
- AccessToken
- AccessTokenSecret

To use these keys in a MATLAB script, users assign the values to their credential structure. Then, a twitty instance can be created and methods, such as search, can be called as shown in Figure 1.

```
% Create credentials
credentials.ConsumerKey = 'YourConsumerKey'
credentials.ConsumerSecret = 'YourConsumerSecret'
credentials.AccessToken = 'YourAccessToken'
credentials.AccessTokenSecret = 'YourAccessTokenSecret'
% Create Twitty Instance
tw = twitty(credentials);
% Search for World Series Related Tweets
tw.search('World Series');
```

Figure 1. Twitty search method example.

Twitty provides a number of useful methods for interacting with the Twitter API. Example calls, along with a brief description, are shown in Table 1. Additional methods and detailed descriptions are found by typing *twitty.API* at the MATLAB prompt.

http://dx.doi.org/10.5772/58895

Method	Example Call	Description
search(text string)	tw.search('World Series')	Search all public tweets containing the words "World Series"
search(multiple parameters)	tw.search('NFL', 'count',20, 'include_entities','true', 'geocode','39.051300,-95.724660,1700mi','since_id',LastID)	Search public tweets having a identifier greater than LastID within 1700 miles of Topeka, Kansas containing the word "NFL" . Return entity data also (e.g., hashtags, URL mentions)
updateStatus()	tw.updateStatus('Watching the World Series tonight')	Twit the text "Watching the World Series tonight" from your account to the public
sampleStatuses()	tw.sampleStatuses()	Return a continuous stream of random public tweets

Table 1. Twitty method examples.

3.2. MATLAB — Statistics TOOLBOX™

The Statistics Toolbox™ provides statistical and machine learning algorithms and tools for organizing, analyzing, and modeling data. Key features include data organization and management via data set arrays and categorical arrays, exploratory data analysis with the use of interactive graphics and multivariate statistics, and regression or classification for predictive modeling [11]. The toolbox also includes functions that allow users to test hypotheses more effectively by checking for autocorrelation and randomness as well as other tests, for example, t-tests, one-sample tests, and distribution tests such as Chi-square and Kolmogorov-Smirnov.

Clustering algorithms available within MATLAB's Statistical Toolbox include *k*-means and hierarchical approaches. Clustering algorithms are particularly useful for analyzing social networking data as they help identify natural groupings that can then be further analyzed to determine similarities or differences and make business, marketing, or other decisions.

A *k*-means clustering algorithm forms *k* clusters by minimizing the mean between all cluster members. Clusters are defined by the centroid or center of each cluster. The algorithm works by moving data between clusters until the sum of the distances between each member and the centroid is minimized.

MATLAB allows different distance measures to be selected. Table 2 lists the available distance measures and a brief description and example call.

Distance Measure	Description	Example call
Squared Euclidean	Each centroid is the mean of the points in that cluster.	kmeans(meas,3,'dist','sqeuclidean');
City Block	Sum of absolute differences. Each centroid is the component-wise median of the points in that cluster.	kmeans(meas,3,'dist','cityblock');
Cosine	One minus the cosine of the included angle between. Each centroid is the mean of the points in that cluster.	kmeans(meas,3,'dist','cosine');
Correlation	One minus the sample correlation between points. Each centroid is the component-wise mean of the points in that cluster.	kmeans(meas,3,'dist','correlation');
Hamming	Percentage of bits that differ in binary data. Each centroid is the component-wise median of points in that cluster.	kmeans(meas,3,'dist','hamming');

Table 2. MATLAB *k*-means clustering distance measures.

Visually displaying the results for multidimensional data can be challenging. However; the silhouette plot, available within the MATLAB Statistics Toolbox, displays a measure between 0 and 1, representing how close each point in one cluster is to the points in the neighboring clusters. Values close to 1 indicate points are distant from neighboring clusters whereas values close to 0 indicate points are not distinctly different from one cluster or another. Negative values indicate points that are most likely assigned to the wrong cluster.

In the following example, cidx2 represents the cluster index for each given sample of data. Assuming cidx2 was returned from the call to the *k*-means function, an example call of the silhouette function is:

```
silhouette(meas,cidx2,'sqeuclidean');
```

Hierarchical clustering groups data to create a tree structure consisting of multiple levels. Users can prune parts of the tree depending upon the application and level of detail required. The links between data are represented as upside-down U-shaped lines with the height of each line indicating the distance between the data. This height is known as the cophenetic correlation distance between the two objects. MATLAB has a function that measures this distance.

http://dx.doi.org/10.5772/58895

The closer the value of the cophenetic correlation coefficient is to 1, the more accurately the clustering solution reflects your data.

Distance measures, similar to the *k*-means clustering, need to be selected to generate the linkages of the tree. Figure 2 shows the code to create a tree based on the Euclidean distance for a matrix named 'meas' and displays the results in a tree and then calculates the cophenetic correlation.

```
eucD = pdist(meas,'euclidean');
clustTreeEuc = linkage(eucD,'average');
[h,nodes] = dendrogram(clustTreeEuc,0);
measCophenet = cophenet(clustTreeEuc,eucD);
```

Figure 2. MATLAB code to calculate cophenetic correlation.

3.3. MATLAB – Neural Network TOOLBOX™

The Neural Network Toolbox™ provides functions and apps for modeling complex nonlinear systems that are not easily modeled with a closed-form equation. Primary features include data fitting, clustering, and pattern recognition to forecast future events, both supervised and unsupervised network architectures, and training algorithms, such as gradient descent and conjugate gradient methods, to help automatically adjust the network's weights and biases [12]. Also included are preprocessing and post-processing functions that improve the efficiency of neural network training and enable detailed analysis of network performance by reducing the dimensions of the input vectors using principal component analysis and normalizing the mean and standard deviation of the training set.

MATLAB's Neural Network Toolbox provides self-organizing maps for both unsupervised and supervised clustering. Self-organizing maps retain topological information for similar classes and provide reasonable classifiers. Self-organizing maps can be used for multidimensional data with complex features, making them attractive for analysis of social networking data. The self-organizing maps functionality within MATLAB can be used to cluster tweets based on the Twitter-extracted fields.

Self-organizing maps learn to classify input vectors according to how they are grouped in the input space. The neurons in a self-organizing map can be arranged based on specific topologies. Within MATLAB, topologies for gridtop, hextop and random are possible. Results are not affected by the choice of topology used; however, visual inspection of the data space is best achieved by use of a hexagonal grid (hextop) [13]. A gridtop topology has neurons evenly spaced in matrix of specific dimensions. For example, an 80-neuron self-organizing map may be arranged in an 8 × 10 gridtop topology. A hextop topology is similar, but the shape of the grid can best be described as a hexagon. A random topology has the neurons randomly located. Figure 3 illustrates gridtop, hextop and random topologies for a 3 × 4 set of neurons.

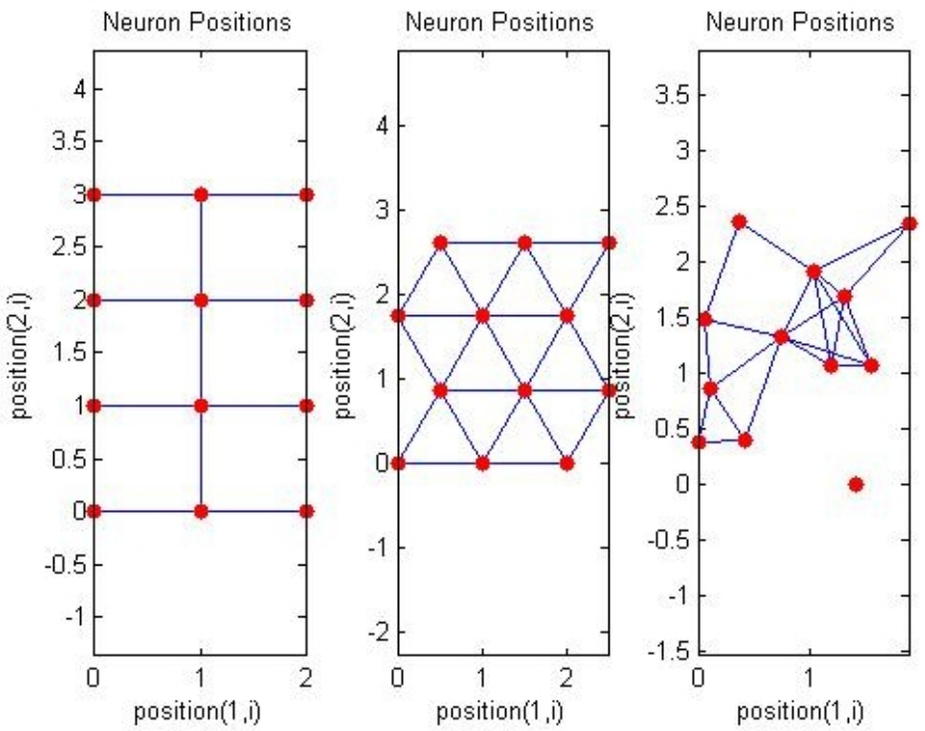

Figure 3. Gridtop, hextop, and randtop self-organizing map topologies.

The self-organization map routine can be run in MATLAB by calling the selforgmap function. Once a map is set up, it can be trained using specific input, and results can be displayed visually. Figure 4 shows a code example to generate a 3 × 4 neuron self-organizing map, conduct training, and display the results.

```
net = selforgmap([3 4]);
net = train(net,meas);
view(net);
y = net(meas);
classes = vec2ind(y);
```

Figure 4. MATLAB code example to generate a 3 × 4 neuron self-organizing map, conduct training, and display the results.

The Neural Network Toolbox also contains the nnstart tool, which provides an interactive tool for loading data and selecting algorithms and training iterations. As shown in Figure 5, the interactive tool allows the user to select run options and then train for a specific number of epochs.

http://dx.doi.org/10.5772/58895

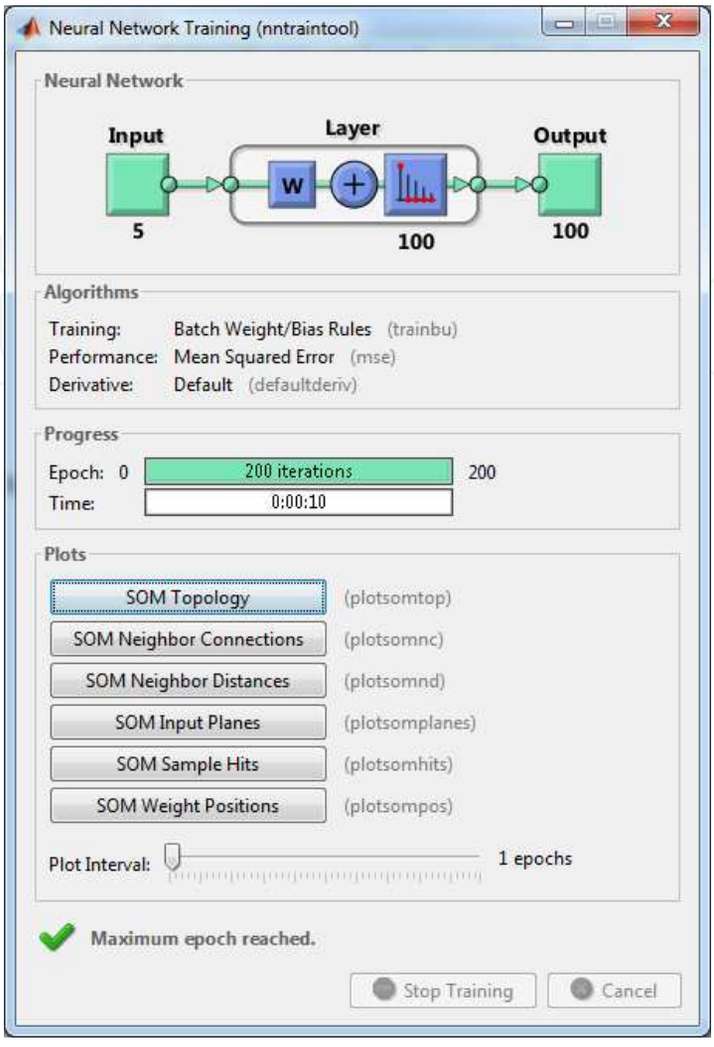

Figure 5. MATLAB's nnstart interactive interface.

The addition of the MATLAB's Parallel Computing Toolbox™ allows the user to speed up training and handle large data sets by distributing computations and data across multiple processors and graphic processing units (GPUs). The ability to spin up multiple central processing units (CPUs) and GPUs from cloud services such as AWS makes this toolbox attractive. The user may be limited by the availability of MATLAB licenses for this option. MATLAB and AWS are currently working out a process to make this more feasible with a pay-as-you-go license feature that could be quite attractive for businesses that do not have the resources to purchase hardware and software.

3.4. Deploying MATLAB on AWS

Cloud service providers such as AWS can significantly reduce the startup and maintenance costs for high-performance computers needed for efficient running of complex math, statistics, and optimization algorithms. AWS has free and pay tier options that will fit most budgets. Previous research has shown the cost of running multiple high-performance servers for many

hours was just a few dollars [14]. As long as users have a valid MATLAB license, they can install MATLAB on the AWS machine(s) and run the data and analysis.

Once the runs are complete, users can disable the MATLAB license on the AWS machine. If users need to run the experiment again, they can spin up the server and enable the license to continue the experiment.

4. Experiment

The experiment consisted of running MATLAB code on commercial cloud architecture (AWS) to collect publically available tweets on common keywords, such as NFL and World Series, during specific time intervals of a weekend over a large geographic region when both NFL games and World Series games were being played. Various MATLAB tools and functions were used to gather statistical information and analyze the data by use of unsupervised learning and clustering methods.

For this experiment, a server running Windows[2] 2008 operating system was spun up in the AWS free tier. A MATLAB license was installed that included the Statistics and Neural Network toolboxes. The twitty.m and parse_json.m files were also uploaded to allow MATLAB to call the Twitter API.

After successful installation, a MATLAB script file (m file) was created to provide the proper Twitter credentials, and then search for tweets related to two different sports events, the World Series and NFL games. The specific search strings used included "NFL" and "World Series".

Four different 3-hour data collections were made over a period of two days. Collection times included Saturday from 5 PM to 8 PM and 8 PM to 11 PM and Sunday from 1 PM to 4 PM and 5 PM to 8 PM during the weekend of the 2013 World Series. During this time frame, a World Series event took place on both Saturday and Sunday evenings, and a series of NFL games occurred on Sunday afternoon and evening.

A large geographic collection area, centered over a 1700 mile radius about Topeka, Kansas, was used for this experiment. This geographic radius included most of the continental United States and parts of Canada and Mexico. The search parameters included the Twitter entities for storing User-, URL-, and hashtag-mentions. Sampling occurred every 60 seconds over the 3-hour window with Twitter ID information being used to eliminate duplicate tweets in the time frame. A sample call for the search is as follows:

```
S=tw.search('NFL',        'count',20,        'include_entities','true','geoco-
de','39.051300,-95.724660,1700mi', 'since_id',LastID);
```

In this call, Twitter is being searched for up to 20 tweets that include the text "NFL" that originated within 1700 miles of Topeka, Kansas. The LastID variable was updated after each

2 Registered trademark of Microsoft Corporation.

http://dx.doi.org/10.5772/58895

iteration to gather only recent tweets beyond a specific ID. Appendix A shows the MATLAB m file that was run for the experiment.

Statistics and analysis were run at the end of the data collection using existing MATLAB tools and functions provided in the Statistics and Neural Network Toolboxes. Within the Statistics Toolbox, the *k*-means and hierarchical clustering approaches and associated visual displays were used.

The distance measure used for all *k*-means calculations was the default selection of Squared Euclidian. An example call used for three clusters with Squared Euclidian distance is as follows:

```
kmeans(meas,3,'dist','sqeuclidean');
```

The self-organizing map functions within the Neural Network Toolbox were used to cluster the Twitter collected data.

In addition to the existing Statistical and Neural Network functions, some simple data manipulation algorithms were used to extract the time between tweets. As shown in Figure 6, the following code example determines and stores the time between tweets found in MATLAB cell array named DatesQ2.

```
for i=1:length(DatesQ2)-1

  time1 = [str2num(DatesQ2{i}(26:30)) 10 str2num(DatesQ2{i}(9:10)) str2num(DatesQ2{i}(12:13))
str2num(DatesQ2{i}(15:16)) str2num(DatesQ2{i}(18:19))];

  time2 = [str2num(DatesQ2{i+1}(26:30)) 10 str2num(DatesQ2{i+1}(9:10)) str2num(DatesQ2{i+1}(12:13))
str2num(DatesQ2{i+1}(15:16)) str2num(DatesQ2{i+1}(18:19))];

  deltaWorldSeries(i) = abs(etime(time1,time2));

end
```

Figure 6. MATLAB code to determine time between tweets.

In this code, time {i} and time {i+1} values are constructed using the default time structure of the Twitter collection and converted to a MATLAB time structure. The dates are then differenced using MATLAB's built-in elapsed time function (etime). This process continues to calculate the time delta for all consecutive tweets for all search queries.

Appendix B shows the Data analysis m file used for this experiment.

5. Initial results and discussion

Results were provided using MATLAB's visual and plotting functions. To better understand the frequency, or popularity, of the two search terms for each time period during the experiment, some baseline descriptive statistics were calculated.

5.1. Boxplots

Boxplots are a convenient way for visualizing the interquartile range, average and outlier data. Figures 7–10 represent boxplots for the 3-hour experiment windows from Saturday 5 PM to 8 PM, Saturday 8 PM to 11 PM, Sunday 1 PM to 4 PM, and Sunday 5 PM to 8 PM, respectively.

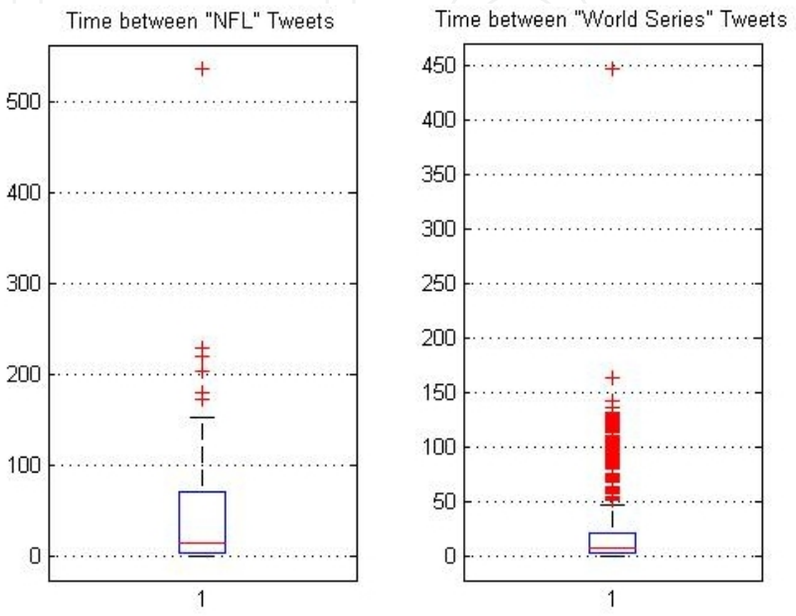

Figure 7. Boxplot for NFL versus World Series time between tweets, Saturday 5 PM to 8 PM.

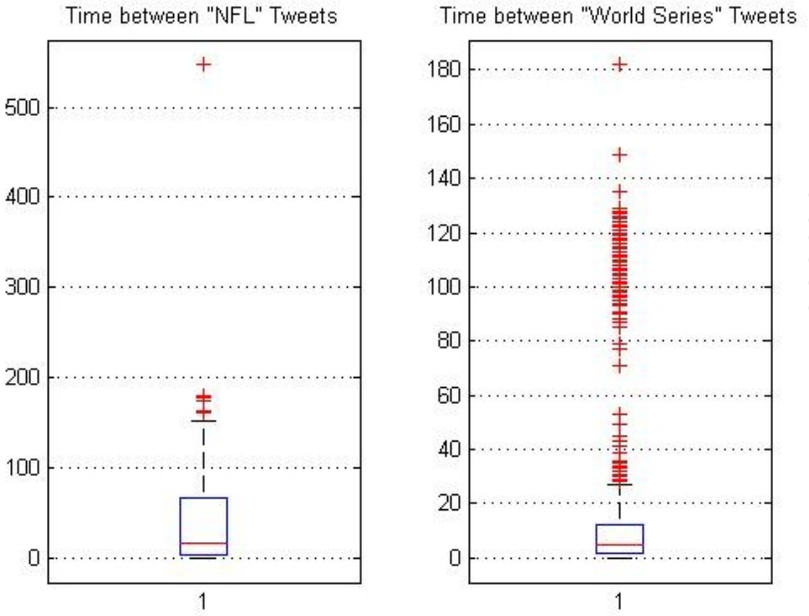

Figure 8. Boxplot for NFL versus World Series time between tweets, Saturday 8 PM to 11 PM.

http://dx.doi.org/10.5772/58895

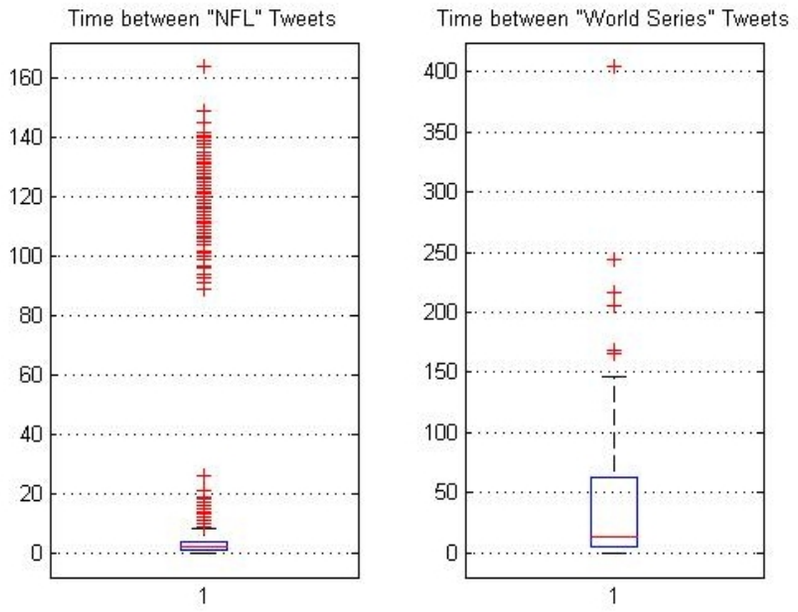

Figure 9. Boxplot for NFL versus World Series time between tweets, Sunday 1 PM to 4 PM.

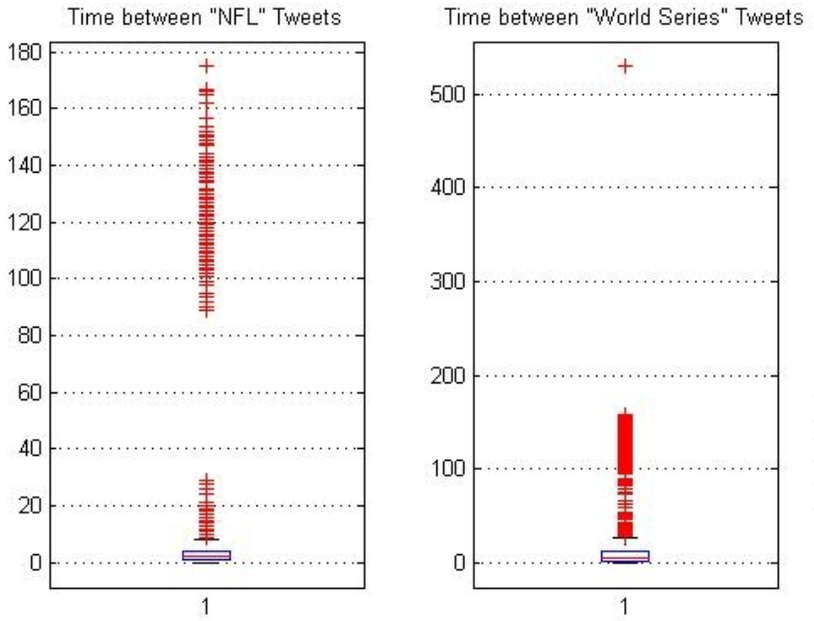

Figure 10. Boxplot for NFL versus World Series time between tweets, Sunday, 5 PM to 8 PM.

Reviewing the boxplots in Figures 7–10 reveals a couple of interesting trends. In most cases, outliers outside of the interquartile ranges exist; however, as a sporting event gets closer to start time, the frequency of tweets increase. Note the narrowing of the boxes for both the NFL

and World Series plots during or approaching the actual game events. By Sunday late afternoon, there was significant activity in both the NFL and World Series tweets in terms of frequency. The descriptive statistics were very similar, making it difficult to see any difference, in terms of tweet frequency between NFL and World Series fans.

5.2. XY plots

MATLAB was also used to plot the time between tweets for each collection period and search query. This view, although somewhat cluttered, can be used to quickly compare the number of tweets and any other patterns obvious in an xy plot. The time between tweets is plotted in Figures 11–14, for each of the four collection periods. The most revealing information from these plots is the noticeable increase in the number of tweets as the events draw closer. On Saturday, the number of tweets related to the World Series was higher than NFL tweets. This seems reasonable since a World Series game took place on Saturday but no NFL games occurred until Sunday.

On Sunday, with multiple NFL games taking place, the number of NFL tweets was larger than World Series-related tweets. However, the number and frequency of World Series tweets was also high due to the Sunday evening World Series event, but not as significant as the NFL-related tweet activity.

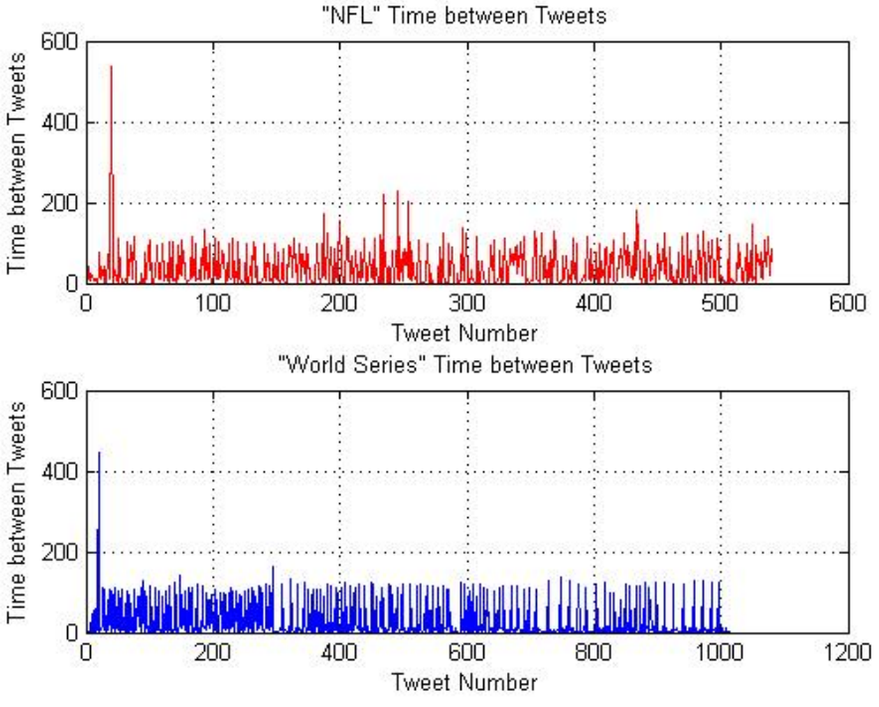

Figure 11. XY plot for NFL versus World Series time between tweets, Saturday 5 PM to 8 PM.

http://dx.doi.org/10.5772/58895

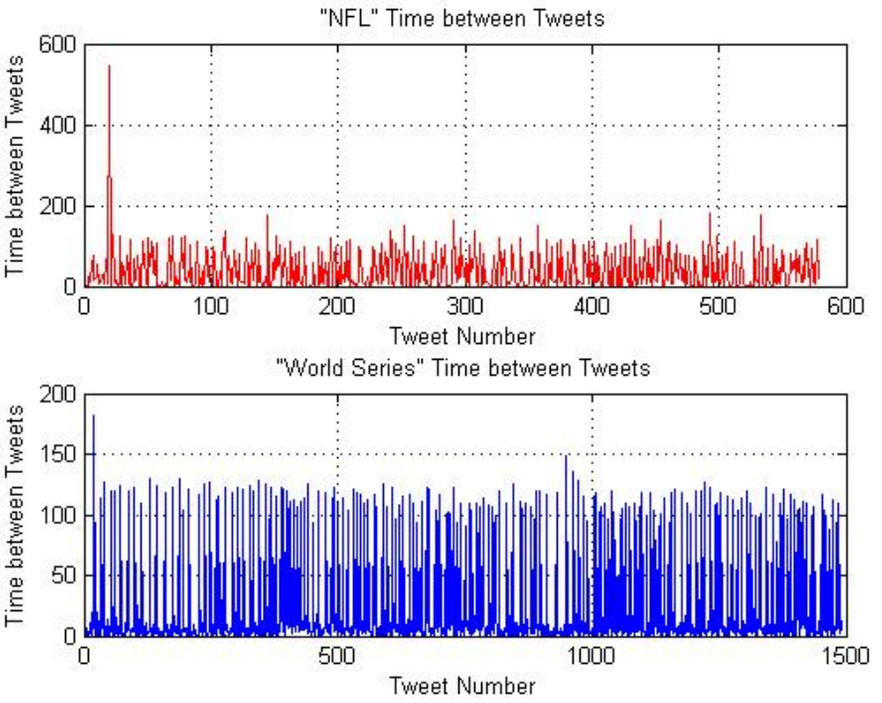

Figure 12. XY plot for NFL versus World Series time between tweets, Saturday 8 PM to 11 PM.

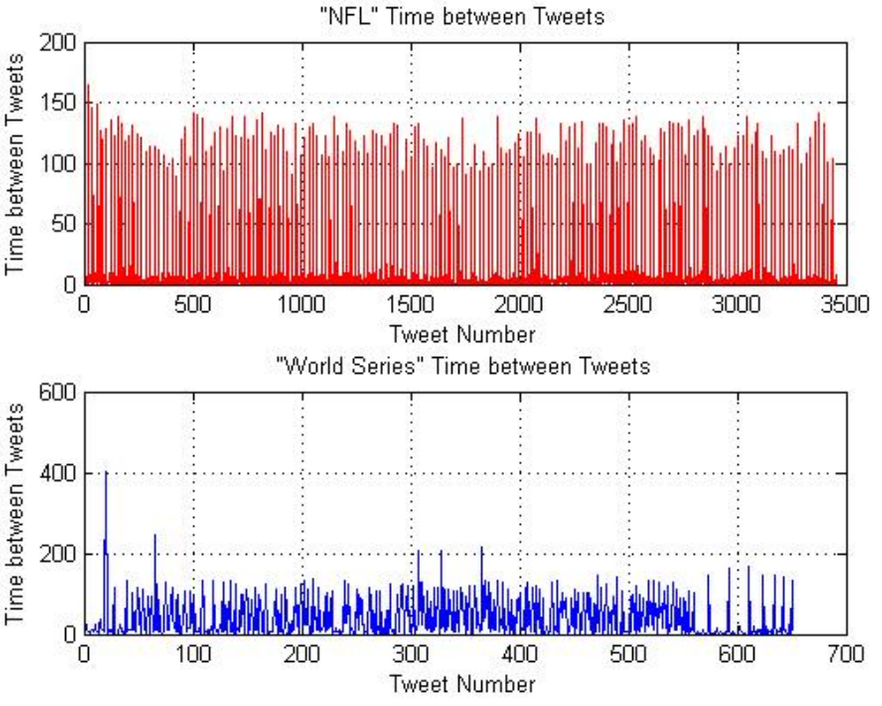

Figure 13. XY plot for NFL versus World Series time between tweets, Sunday 1 PM to 4PM.

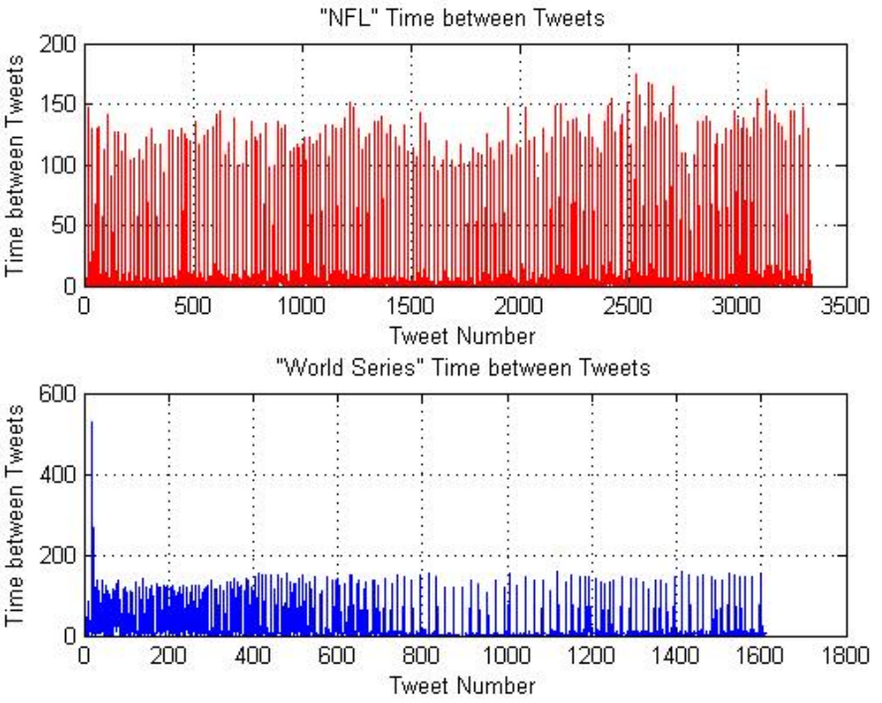

Figure 14. XY plot for NFL versus World Series Time between tweets, Sunday 5 PM to 8 PM.

5.3. Histograms

The histogram functionality within MATLAB was used to get a better idea of the quantity and length of the time between tweets for each of the collection periods. As shown in Figures 15 and 16, on Saturday, the number of tweets with time difference values less than 10 seconds was much greater for World Series than NFL tweets. In all histograms, a handful of tweets had a time between tweets of greater than 100 seconds. The shift to NFL interest is very visible on Sunday in the histograms shown in Figures 17 and 18. Note the significant increase in the both the number and quantity of tweets occurring in less than 10 second intervals for NFL tweets. However, in Figure 18, the histograms for Sunday late afternoon and evening reveal similar results for the time between tweets for both NFL and World Series texts.

http://dx.doi.org/10.5772/58895

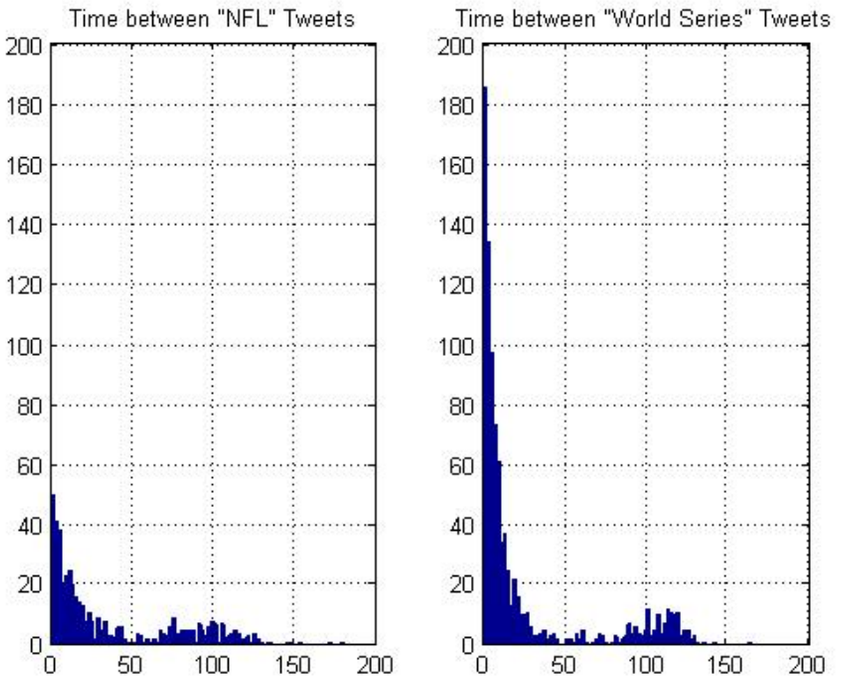

Figure 15. Histogram plot for NFL versus World Series time between tweets, Saturday 5 PM to 8 PM.

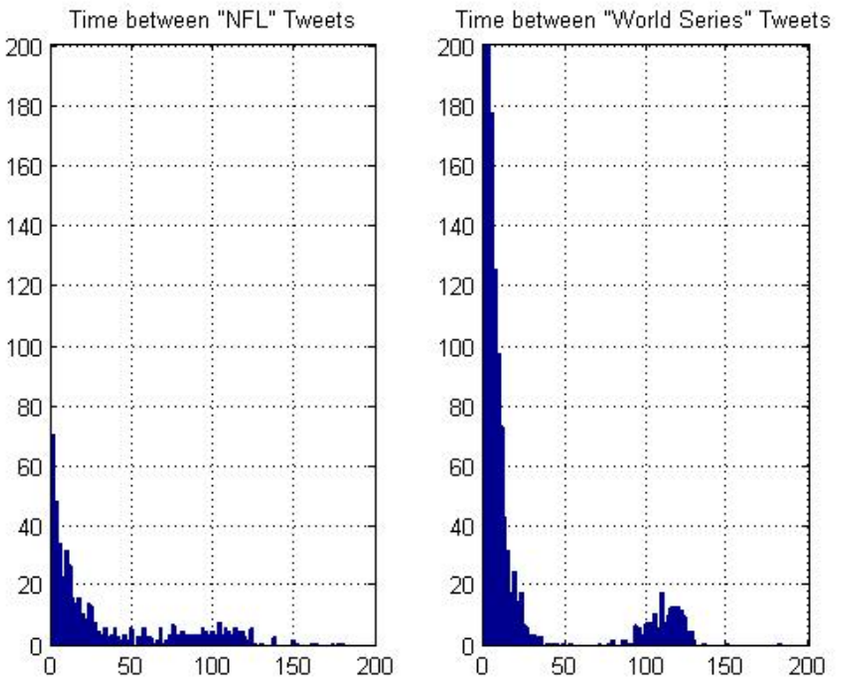

Figure 16. Histogram plot for NFL versus World Series time between tweets, Saturday 8 PM to 11 PM.

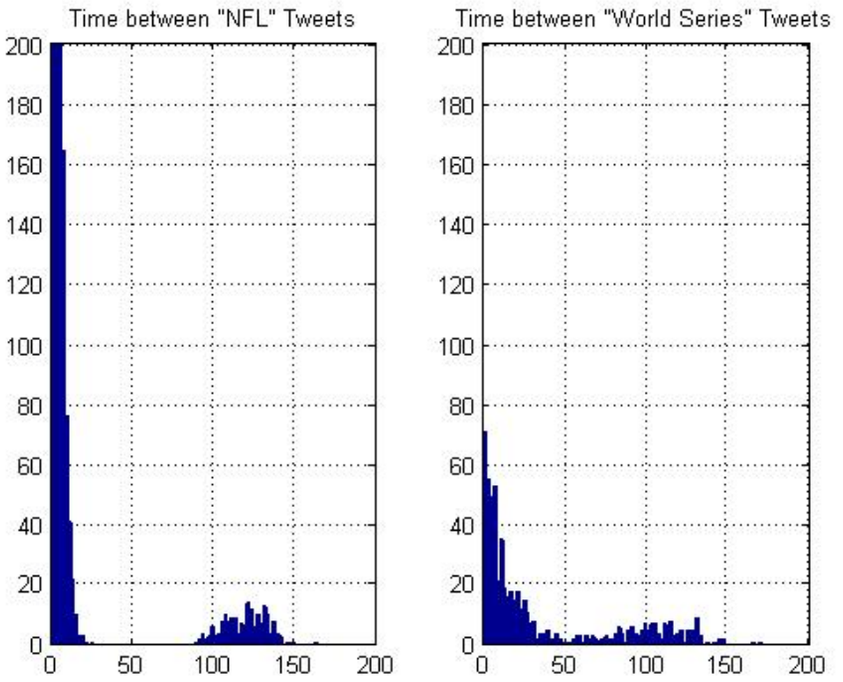

Figure 17. Histogram plot for NFL versus World Series time between tweets, Sunday 1 PM to 4 PM.

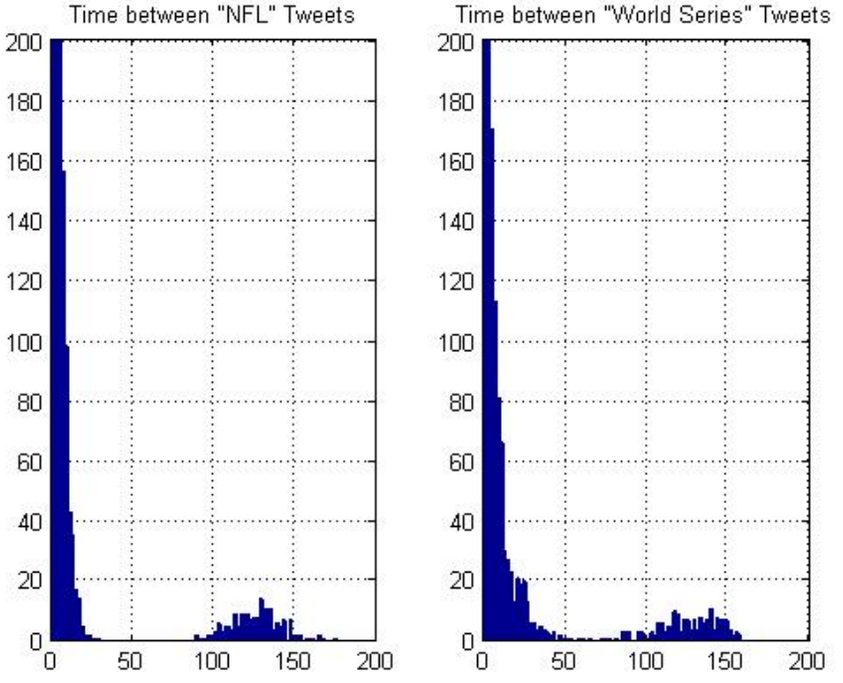

Figure 18. Histogram plot for NFL versus World Series time between tweets, Sunday 5 PM to 8 PM.

http://dx.doi.org/10.5772/58895

5.4. Scatter plots

Twitter queries provide additional data beyond the time and text of the tweet. For example, the number of friends, number of followers, user screen name, date of account creation, and time zone of user are also available and easily extract from the queries. This information can be used to determine if any significant differences or similarities exist among users of Twitter.

Within MATLAB, scatter plots can be used to visually identify clusters of data. Figures 19–22 illustrate the use of scatter plots to compare the number of friends and number of followers for NFL (blue Xs) and World Series (red circles) tweeters during each of the four data collection periods. Although a significant amount of overlap exists in the groups, particularly for small numbers of friend and followers, outliers can be identified in all four plots. Additional variables and analysis may be needed to further isolate these groups.

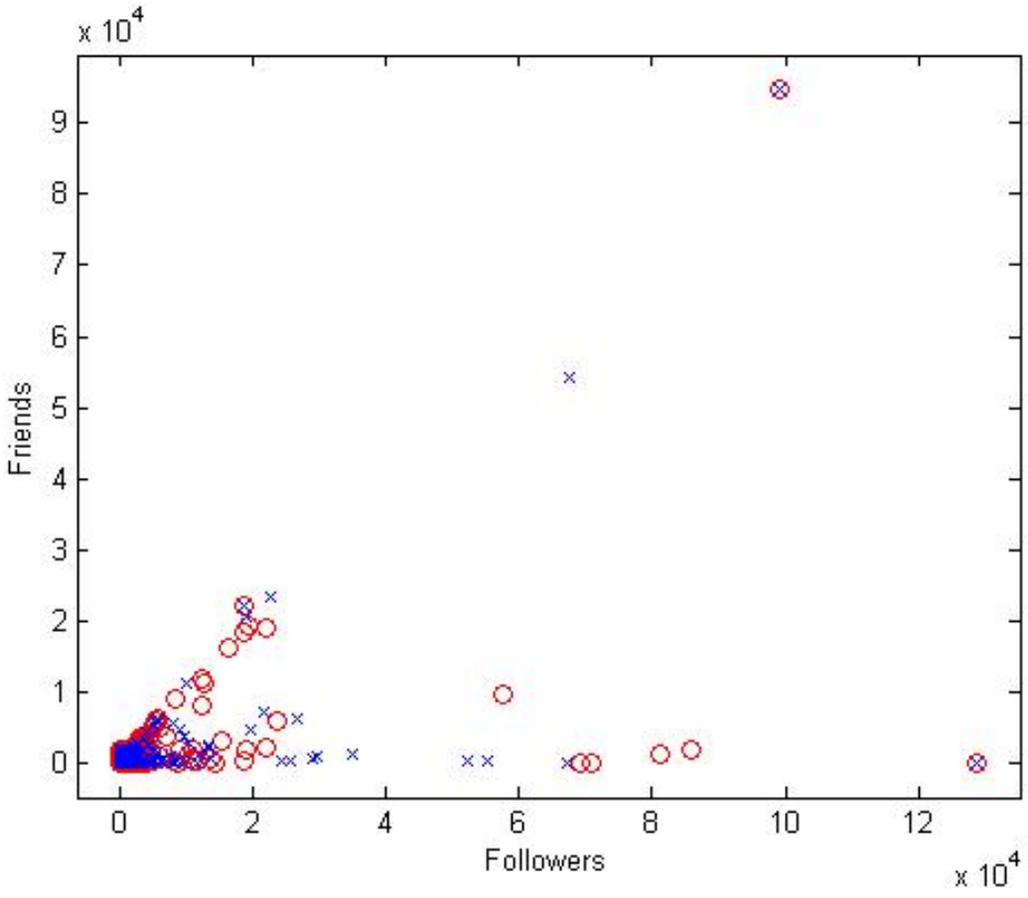

Figure 19. Scatter plot for friends versus followers, Saturday 5 PM to 8 PM.

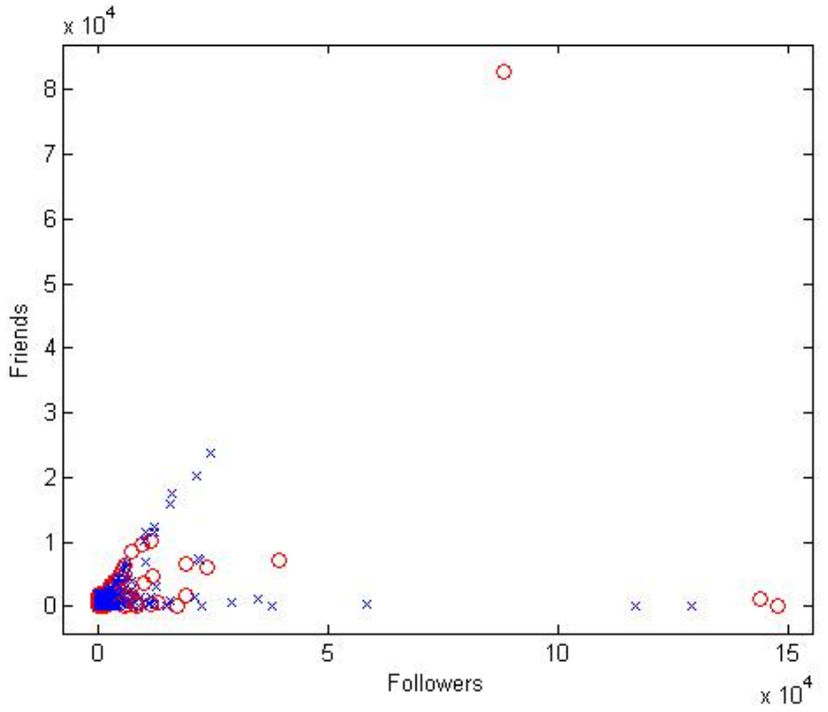

Figure 20. Scatter plot for friends versus followers, Saturday 8 PM to 11 PM.

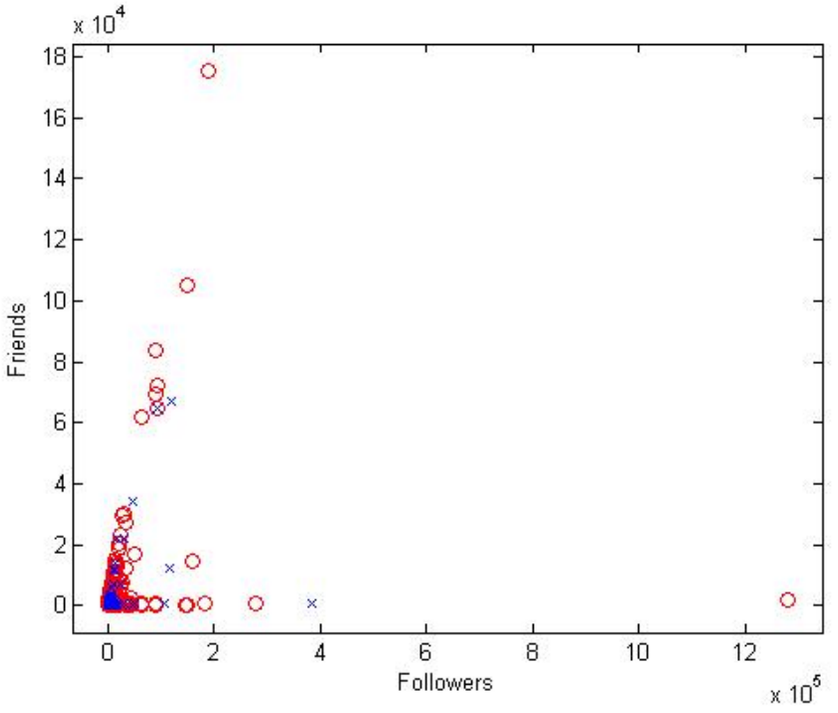

Figure 21. Scatter plot for friends versus followers, Sunday 1 PM to 4 PM.

http://dx.doi.org/10.5772/58895

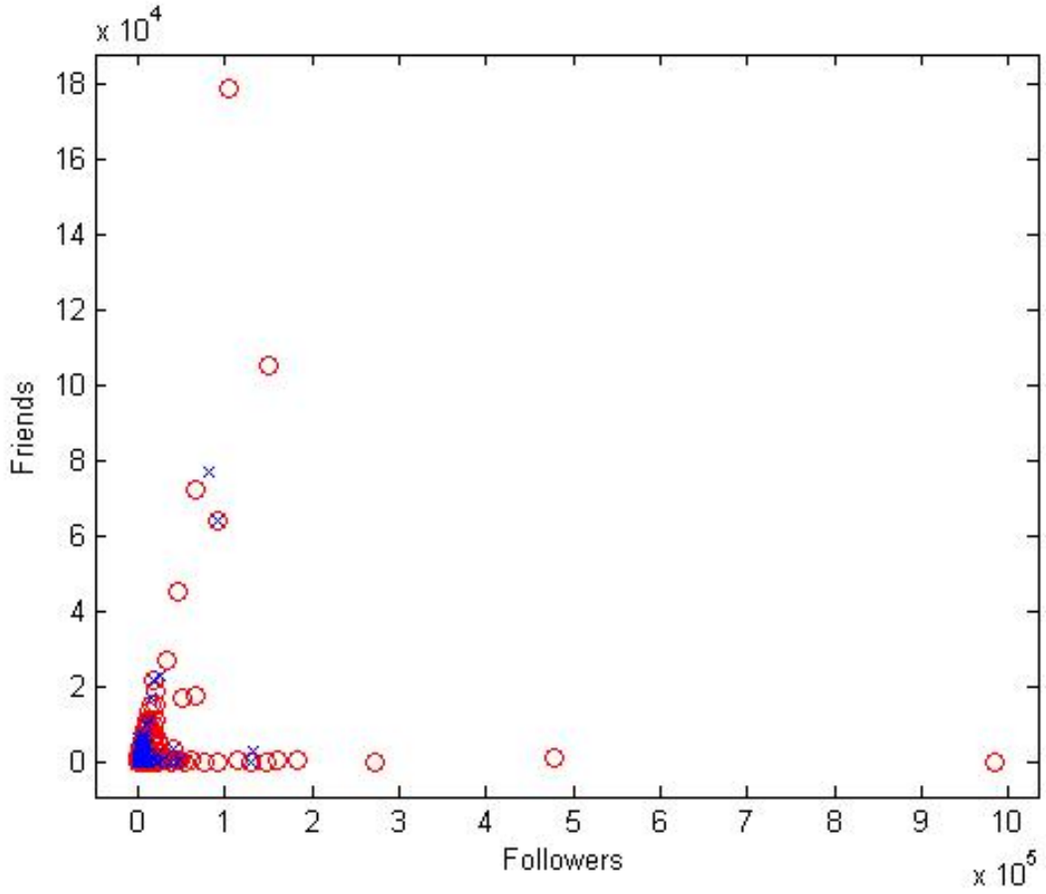

Figure 22. Scatter plot for friends versus followers, Sunday 5 PM to 8 PM.

5.5. Silhouette plots

Running the *k*-means algorithm in MATLAB for the Twitter feature sets further reveals the similarity in both NFL and World Series sets of tweets for all four time collection periods. Figures 23–26 illustrate the results of the applying the *k*-means algorithm for three clusters.

In all cases, one very large cluster with silhouette values very close to 1 is observed. However, the other two clusters are very small with both negative and lower positive silhouette values. This indicates that the separation into unique clusters is difficult for this set of data and features. Additional selection of features and analysis would be needed to better identify clusters of similar Twitter users.

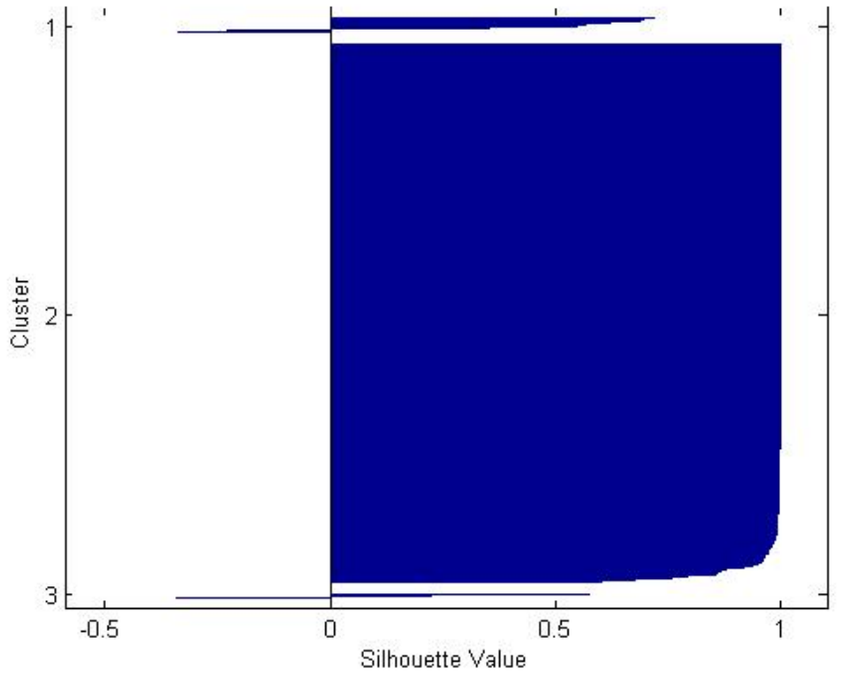

Figure 23. Silhouette plot of NFL versus World Series tweet features, Saturday 5 PM to 8PM.

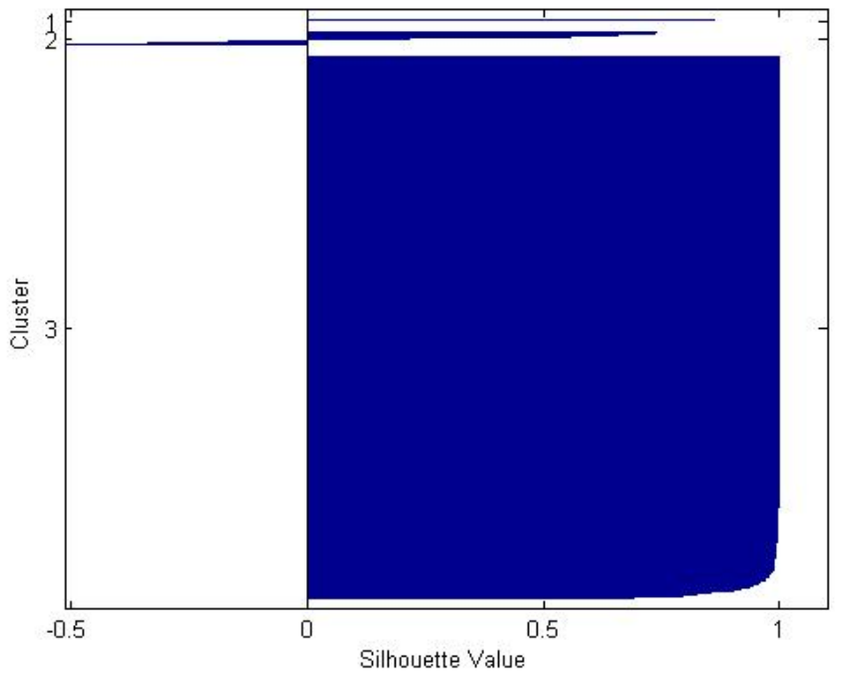

Figure 24. Silhouette plot of NFL versus World Series tweet features, Saturday 8 PM to 11 PM.

http://dx.doi.org/10.5772/58895

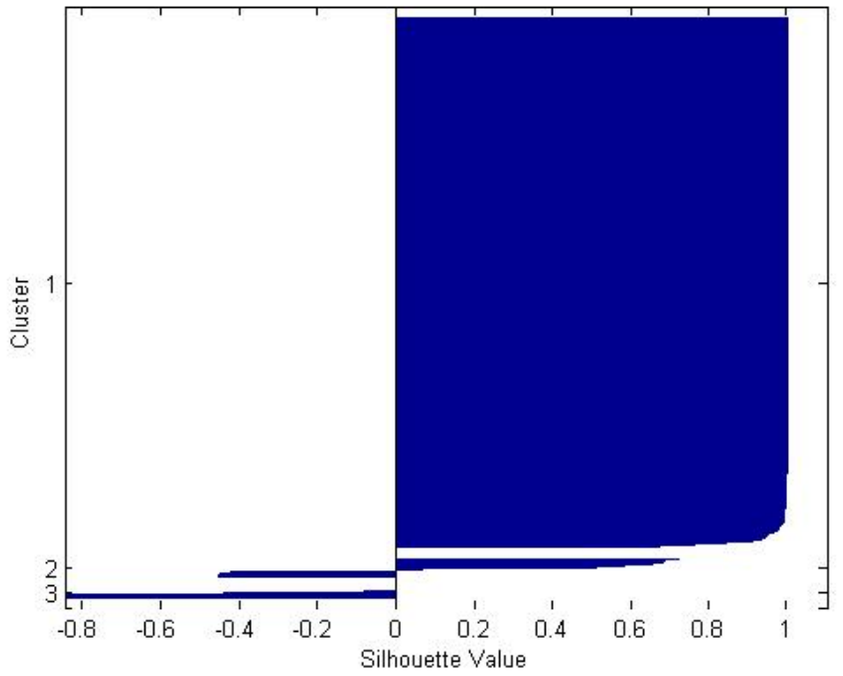

Figure 25. Silhouette plot of NFL versus World Series tweet features, Sunday 1 PM to 4 PM.

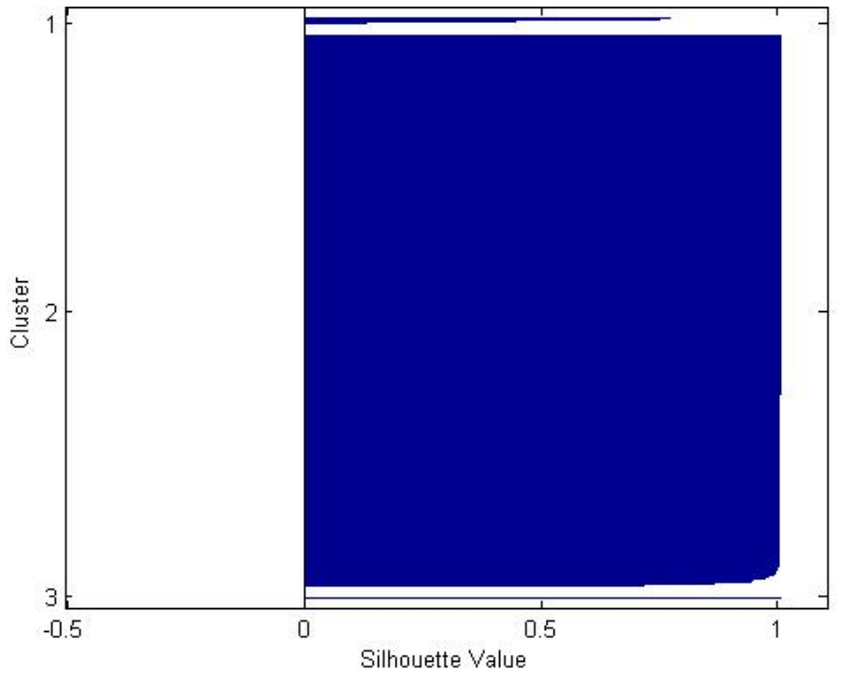

Figure 26. Silhouette plot of NFL versus World Series tweet features, Sunday 5 PM to 8 PM.

5.6. Hierarchical plots

Hierarchical clustering provides an additional visualization of possible cluster separation for the tweet data. In all cases, the number of cluster and representatives within that cluster matches the *k*-means clustering results. Specifically, one large cluster was identified along with at most one or two very small clusters as illustrated in Figures 27–30 where the left portion of the tree has many nodes whereas the right portion just has one or two. The size of the cluster grows over the collection periods with the final collection period on Sunday resulting in the largest cluster with the least amount of additional clusters. For example, on Sunday, the tree's one small branch is evident on the right side of the graph with the larger branch on the left side containing most observations.

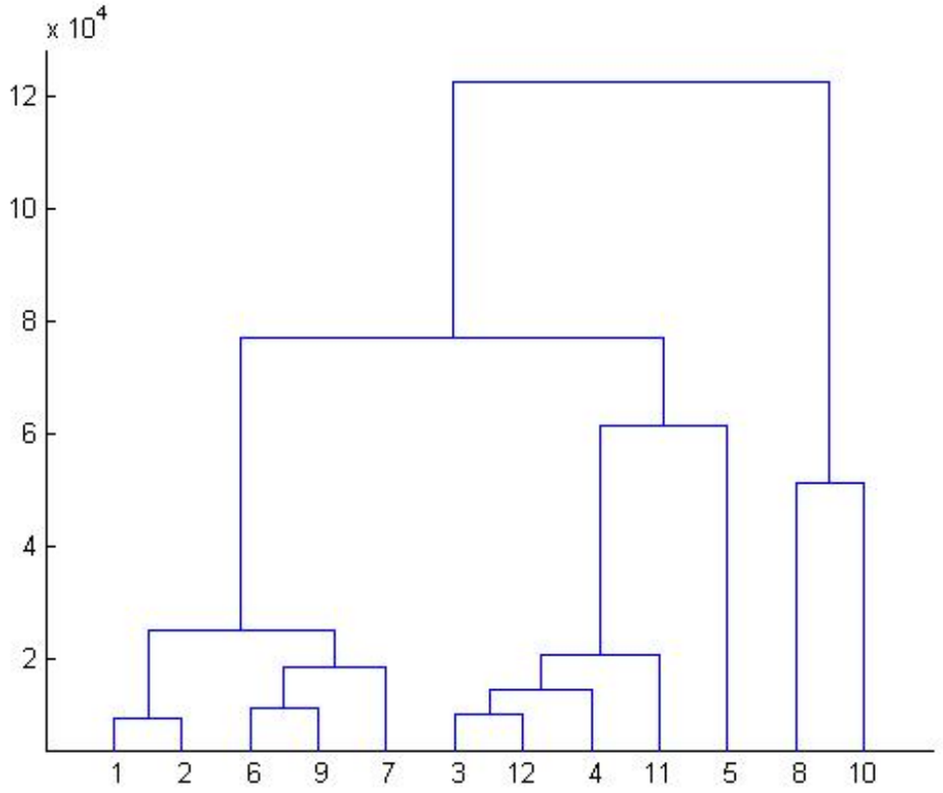

Figure 27. Hierarchical clustering of NFL versus World Series tweet features, Saturday 5 PM to 8 PM.

http://dx.doi.org/10.5772/58895

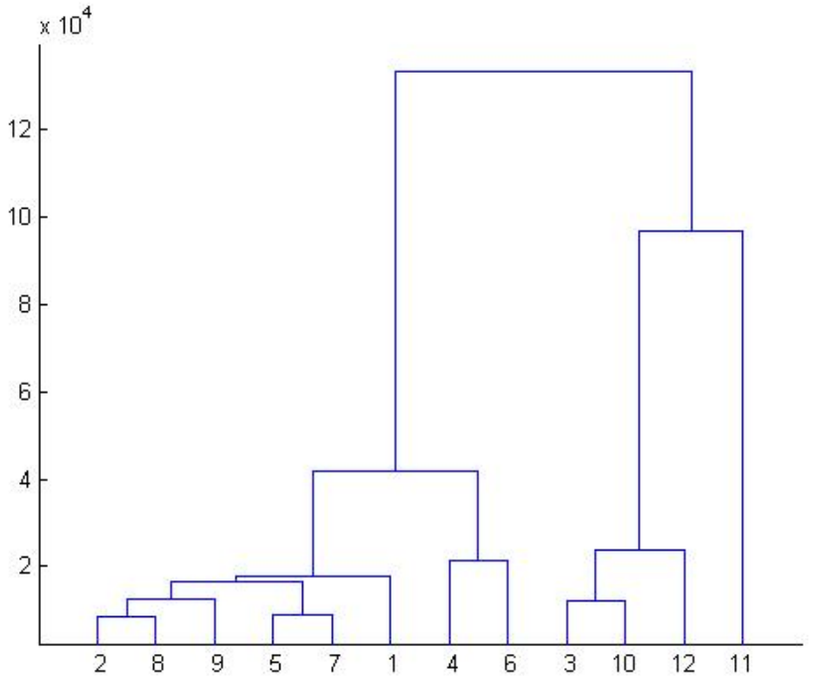

Figure 28. Hierarchical clustering of NFL versus World Series tweet features, Saturday 8 PM to 11 PM.

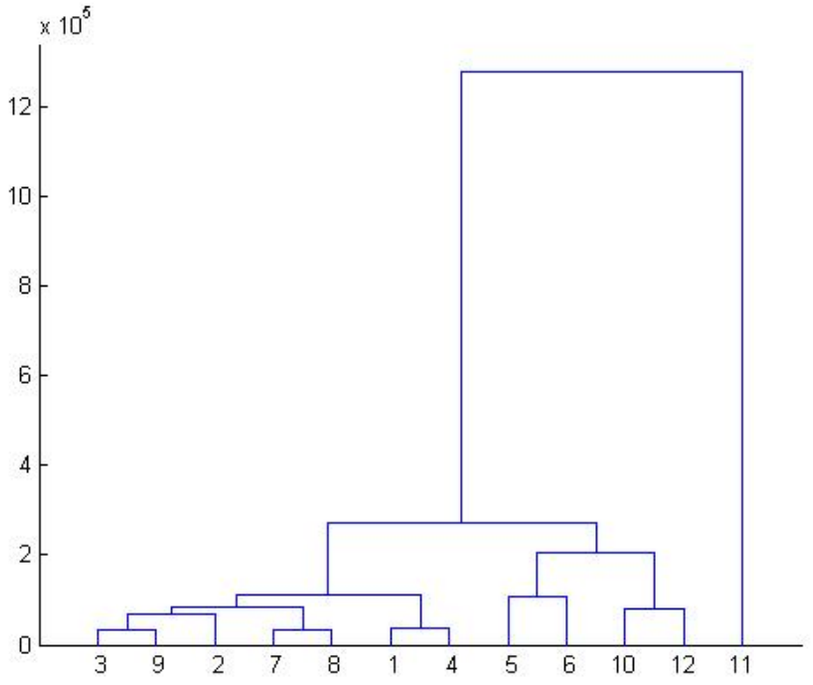

Figure 29. Hierarchical clustering of NFL versus World Series tweet features, Sunday 1 PM to 4 PM.

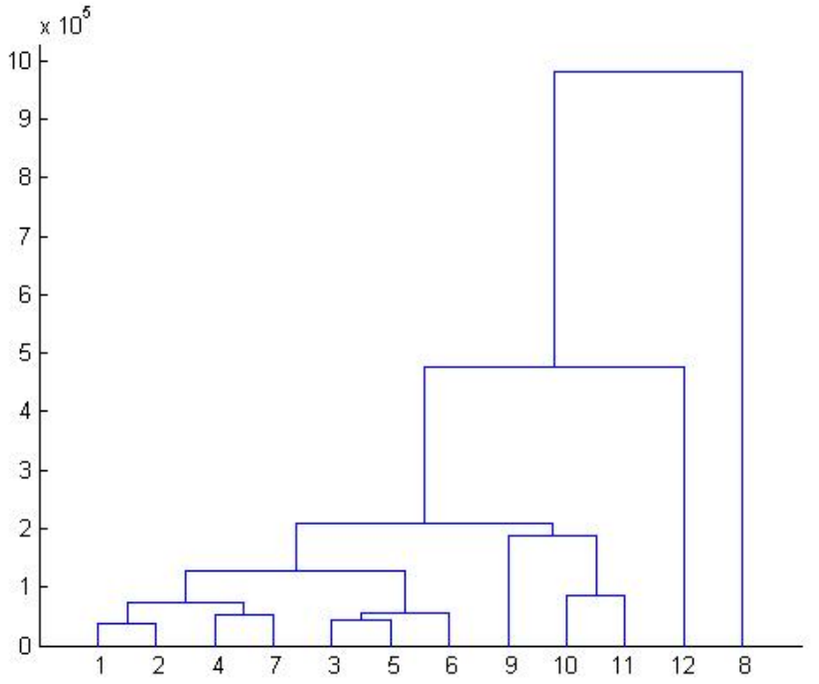

Figure 30. Hierarchical clustering of NFL versus World Series tweet features, Sunday 5 PM to 8 PM.

5.7. Self-Organizing Maps (SOM) plots

MATLAB's Neural Network Toolbox was also used to cluster the tweets through the self-organizing maps function. A hextop topology of 10 ×10 nodes was selected with 200 training epochs. After training, several visualizations are available to help determine how the input is distributed across the nodes. The locations of the data points and the weight vectors are shown by selecting the weight positions plot. With this display, only two weights can be shown at one time. Figure 31 shows that most of the data points cluster in one area and they are not very well distributed. This type of clustering was observed in both the hierarchical and *k*-means results.

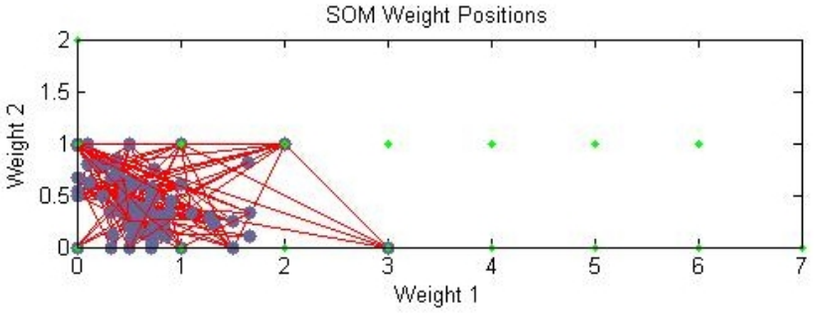

Figure 31. SOM weight positions, Saturday 5 PM to 8 PM.

http://dx.doi.org/10.5772/58895

5.7.1. SOM weight distance plots

The SOM neighbor weight distances plot, using multiple dimensions, provides more information. In Figures 32–35 SOM neighbor weight distance plots are displayed for each of the four time collections. The following diagram colors and description should be used when interpreting these plots:

1. Neurons are represented by blue hexagons
2. Red lines connect neighboring neurons
3. Dark-colored regions represent larger distances between neurons
4. Light-colored regions represent smaller distances between neurons

Figures 32–35 show one large cluster, with small distances between the member records, present in all four collection periods. One or two very small clusters are also present, with relatively large distances between the member records.

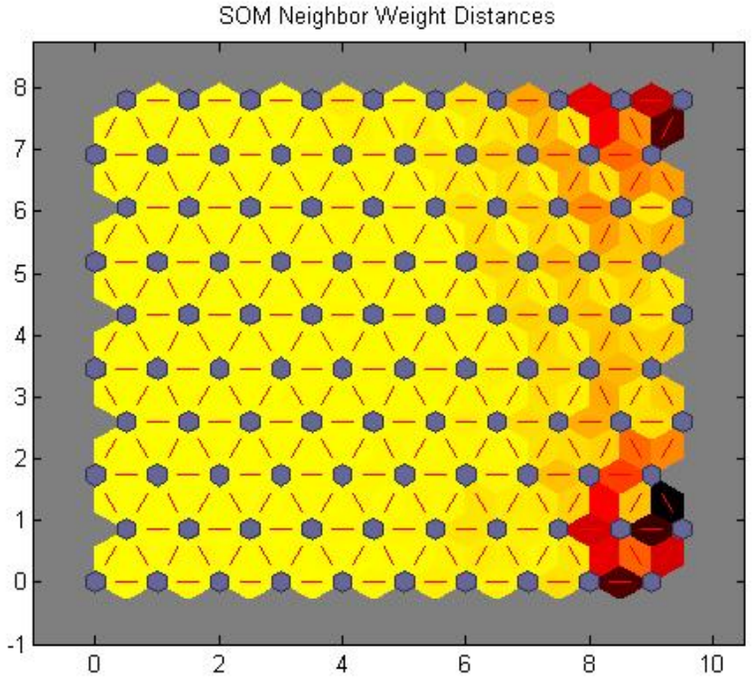

Figure 32. SOM weight distances, Saturday 5 PM to 8 PM.

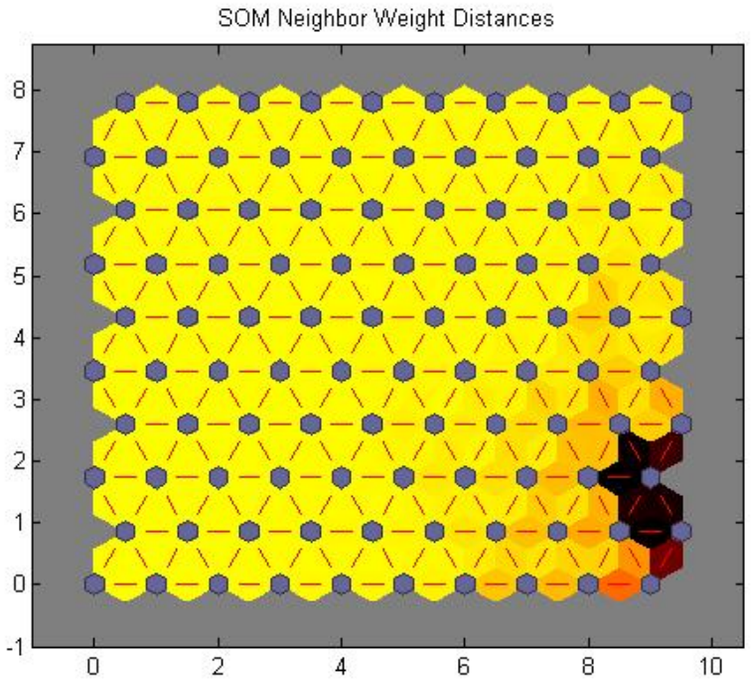

Figure 33. SOM weight distances, Saturday 8 PM to 11 PM.

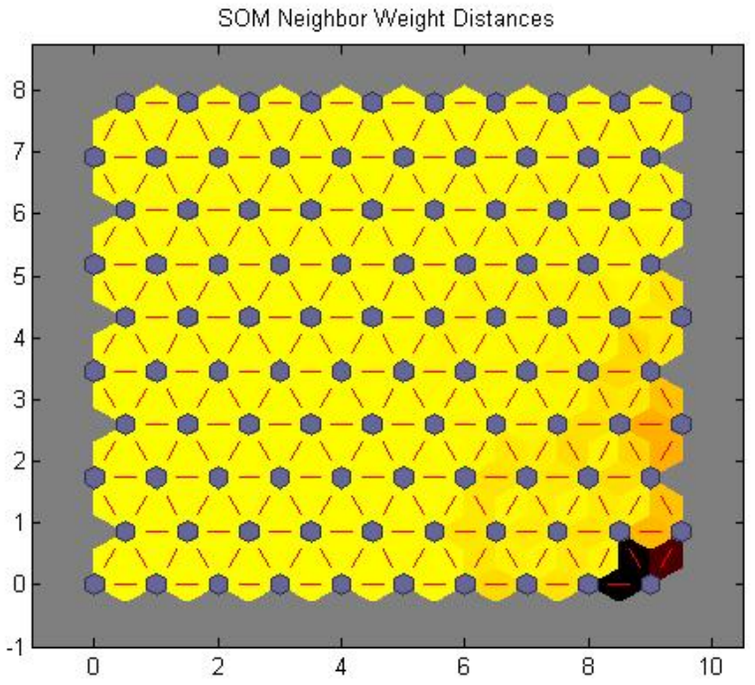

Figure 34. SOM weight distances, Sunday 1 PM to 4 PM.

http://dx.doi.org/10.5772/58895

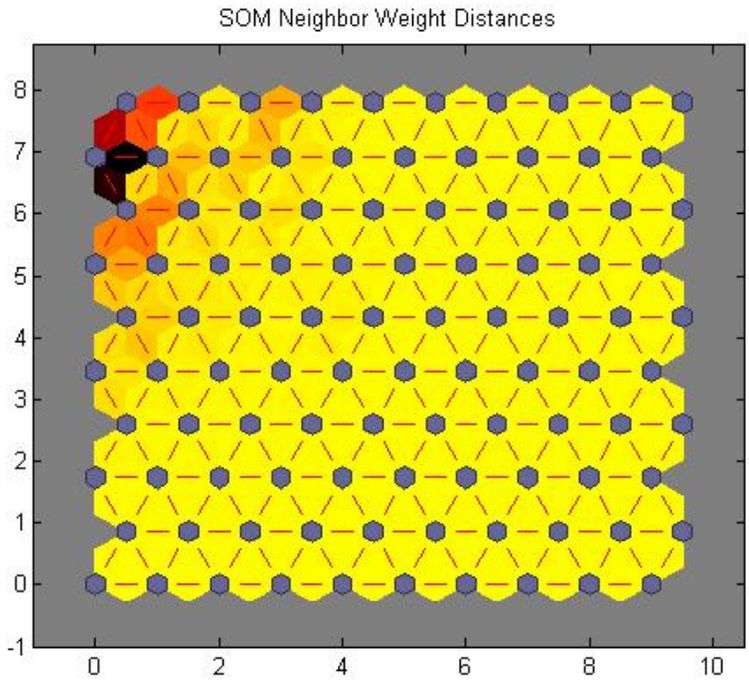

Figure 35. SOM weight distances, Sunday 5 PM to 8 PM.

5.7.2. SOM weight plane plots

The SOM weight plane plots are used to visualize the strength of weights that connect each input to each of the neurons. For our experiment, five inputs were used; therefore, five subplots were generated for each experiment. The five input features included, for each tweet, the number of user mentions, URL mentions, hashtag mentions, followers, and friends. Lighter colors in the plots represent larger weights whereas darker colors represent smaller weights. Similar connection patterns of the inputs indicate a high correlation.

Weight plane plots are shown for each of the collection times in Figures 36–39. Inputs 4 and 5 appeared to be similar in all collection times and were interpreted as highly correlated. Input from variables 1, 2, and 3 seemed to contribute the smallest amount of cluster separation in the data sets as they appear to be the least similar and less correlated. This seems reasonable because the number of friends and followers do seem to be correlated with Twitter users as a large number of friends also have a large number of followers. The information from the user-, URL-and hashtag-mentions shows some promise as the maps show these as not being highly correlated. Additional features and analysis are recommended to enhance the differences in these maps and to perform better clustering.

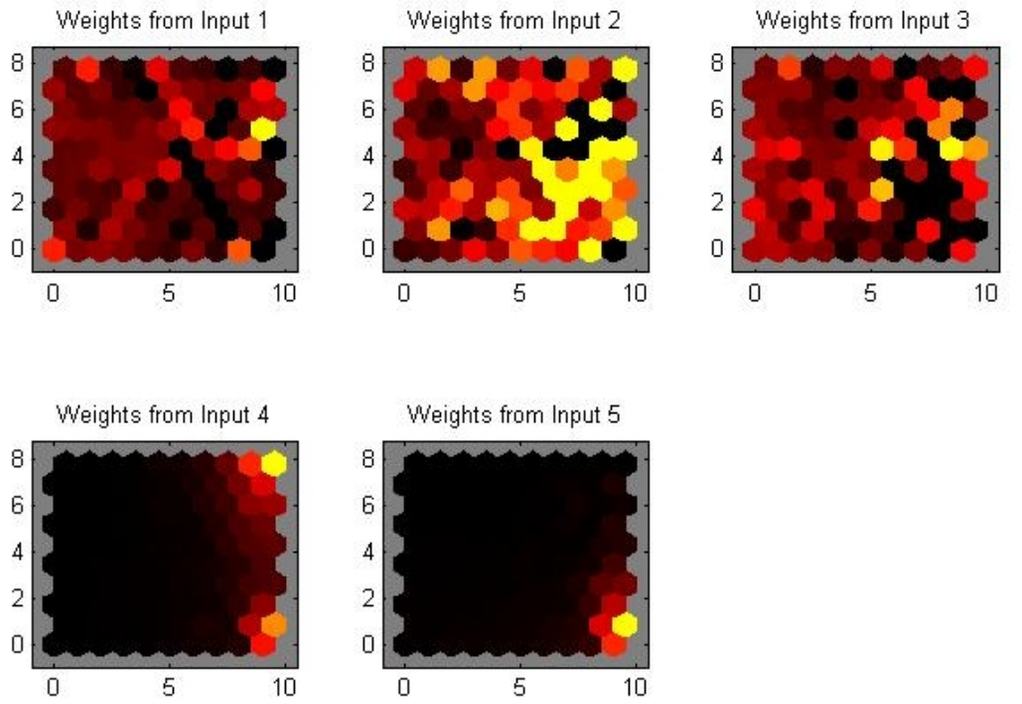

Figure 36. SOM weight planes, Saturday 5 PM to 8 PM.

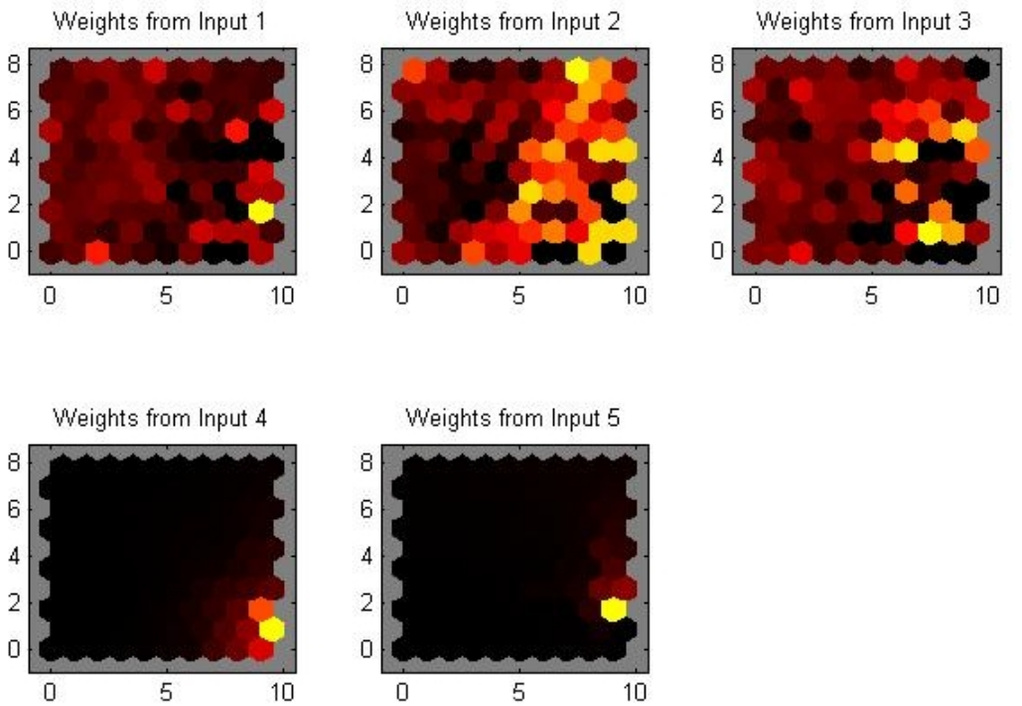

Figure 37. SOM weight planes, Saturday 8 PM to 11 PM.

http://dx.doi.org/10.5772/58895

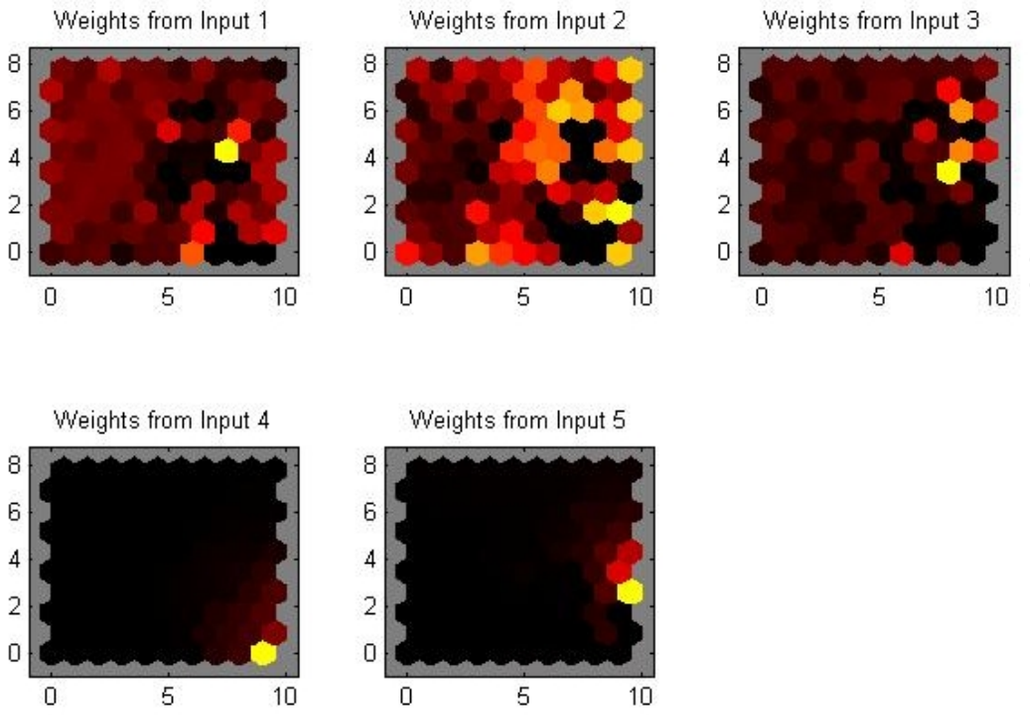

Figure 38. SOM weight planes, Sunday 1 PM to 4 PM.

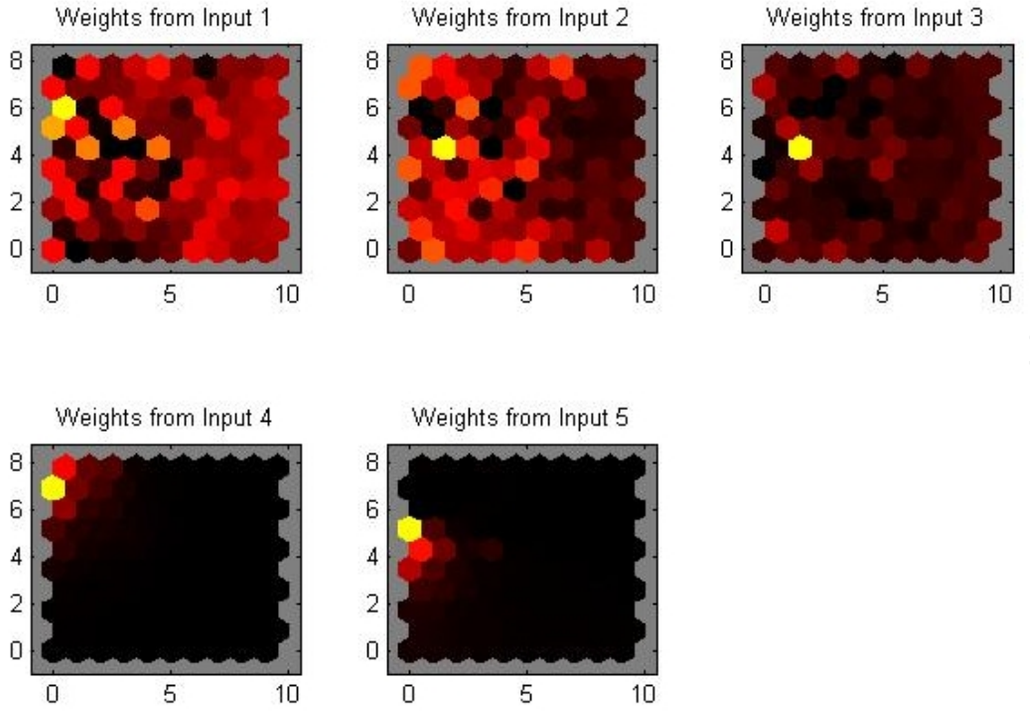

Figure 39. SOM weight planes, Sunday 5 PM to 8 PM.

5.7.3. SOM sample hits plots

One additional and useful visual plot provided by the MATLAB SOM function is the SOM sample hits plot. The sample hits plot counts the number of data records associated with each neuron. In an ideal situation, a relatively even distribution across the neurons is desired.

Figures 40–43 show the distribution across the neurons was not even in our experiment. In most cases, the distribution was clustered in one area, which indicates very similar data without much separation. Even though the location of concentration seems to shift from one area of the map to another for each time frame, the results are essentially the same for each collection time. The exception is the increase in concentration of data around the "heavy-hitting" neurons on Sunday as both NFL and World Series events were scheduled.

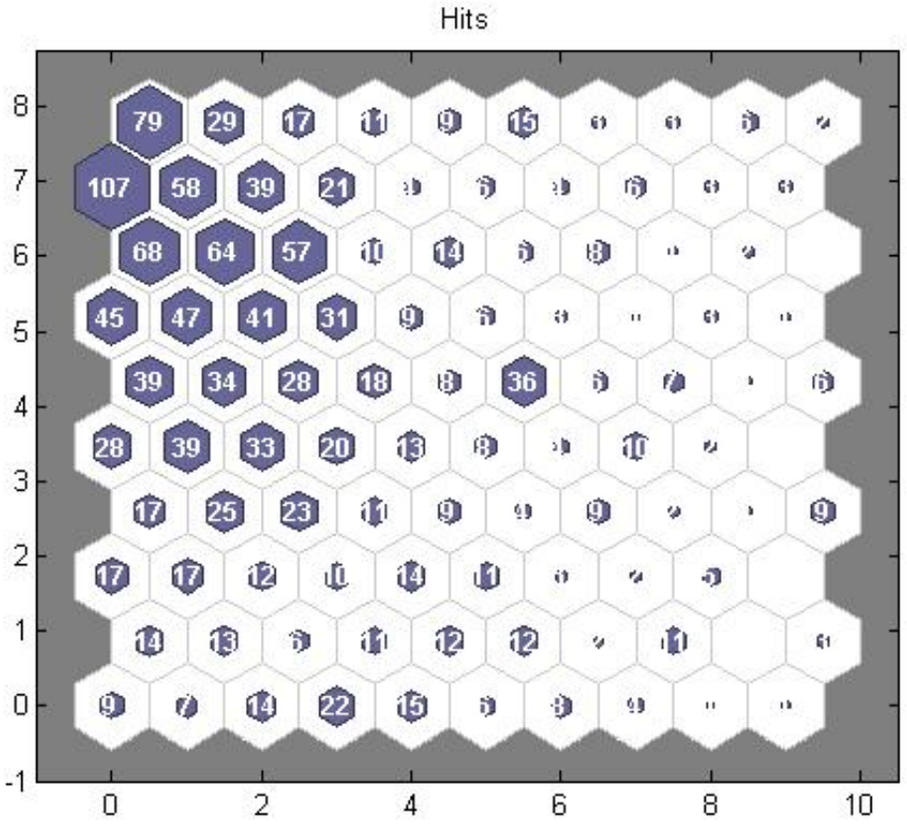

Figure 40. SOM sample hits, Saturday 5 PM to 8 PM.

http://dx.doi.org/10.5772/58895

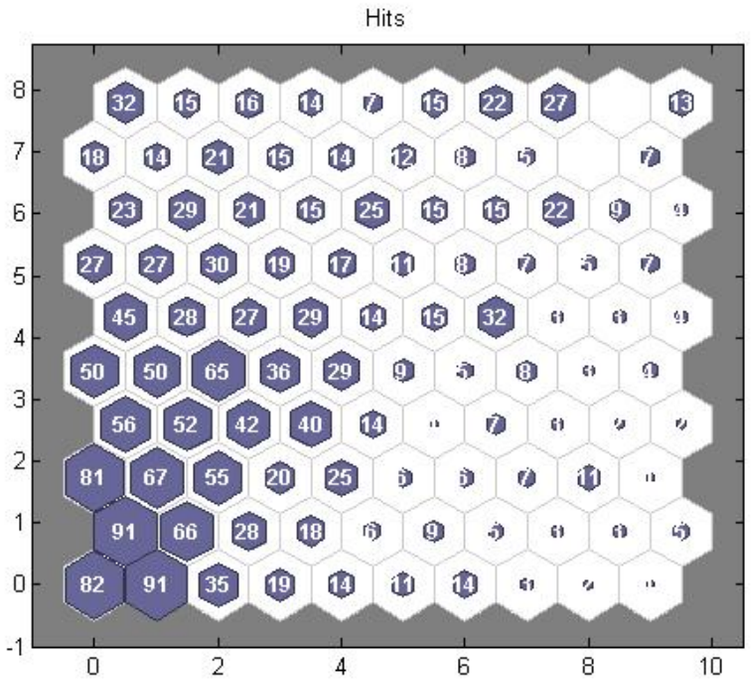

Figure 41. SOM sample hits, Saturday 8 PM to 11 PM.

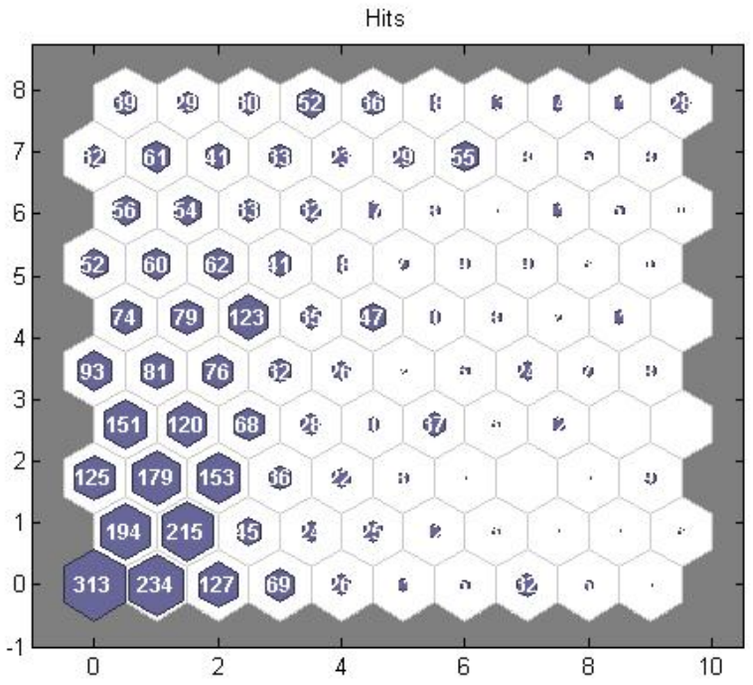

Figure 42. SOM sample hits, Sunday 1 PM to 4 PM.

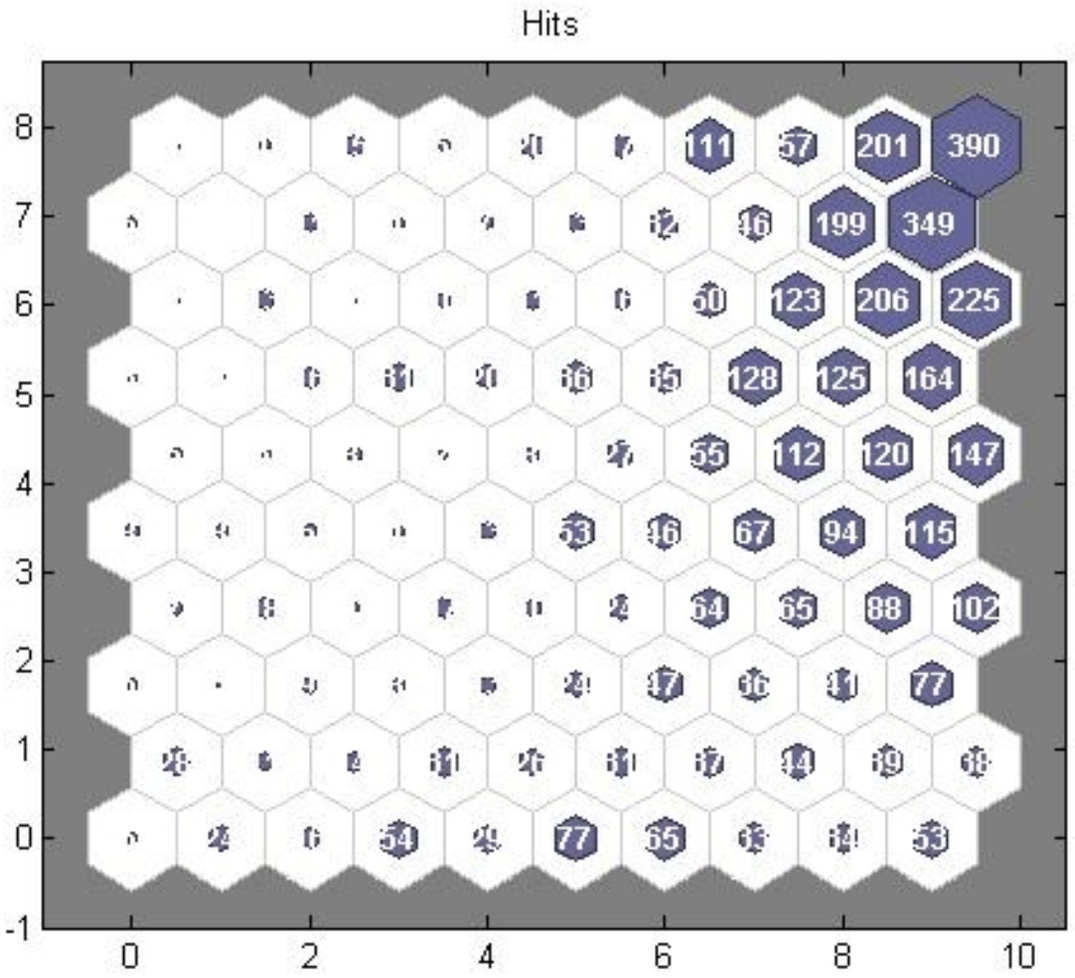

Figure 43. SOM sample hits, Sunday 5 PM to 8 PM.

6. Conclusion

This research used MATLAB tools to extract and analyze social networking data sets and leverage cloud technologies and infrastructures. AWS was used to spin up a Windows 2008 server and install MATLAB and its associated toolboxes for data mining and statistics. In addition, a community MATLAB m file interfaced with the Twitter API to search specific text queries and retrieve associated data. The setup process on AWS was straightforward and provided a cost-effective and free solution for the hardware and operating system. The MATLAB license was still needed in this implementation to be able to use the toolboxes and associated m files.

http://dx.doi.org/10.5772/58895

AWS provided cost savings, including all server-related hardware and software. Since MATLAB was used, the development costs for some very specialized analysis visualization tools were also reduced. Further cost savings are envisioned as MATLAB provides cloud options for running its software for short durations using a pay-as-you-go cost model.

At the time of publishing, access to MATLAB Distributed Computing Server on the cloud is available as part of an Early Adopter Program for MATLAB Distributed Computing Server on Elastic Compute Cloud (EC2). Future efforts could take advantage of this option to speed up implementations, run algorithms in parallel, and possibly further reduce licensing costs.

The twitty.m and parse_json.m community files were successfully used to interface with the Twitter API to search for tweets related to specific queries including "NFL" and "World Series" sporting events. In addition to the tweet texts, other information used in the experiment included user data such as number of friends, number of followers, time between tweets, number of URL mentions, number of Hashtag mentions, and number of user mentions in each text.

The MATLAB Statistics and Neural Network Toolboxes were then used to extract and display descriptive statistics and perform unsupervised clustering using *k*-means, hierarchical, and self-organized maps. Initial analysis of the visualization and data output revealed that sports events and the associated popularity and frequency of tweets increases as the time of the event gets closer. The frequency of tweets also persisted throughout the event.

Tweet users were also shown to have similar characteristics and profiles and would be difficult to separate based on just a few features. Referencing the SOM weight distance plots, smaller clusters were observed in all the plots, however, the distance among the members of the cluster were large compared to the one large cluster observed in each plot. Additional research with larger data sets and more robust features are needed to form predicative models.

7. Recommendations

This research demonstrated the use of MATLAB for social networking site analysis using AWS servers to reduce costs is feasible, cost-effective, and efficient. Further research is recommended to Investigate MATLAB's AWS Parallel computing license options and costs. It is believed even more cost-savings could be realized with a pay-as-you-go model. This is particularly attractive for smaller companies and start-ups that might have limited financial resources, yet have the personnel skills to conduct some excellent research.

This initial experiment collected data from four different, but relatively short, time periods for two sports-related tweets. Only a handful of features were used to identify clusters and perform statistical analysis. We recommend expanding the time frame, number of tweet queries, and features to be able to provide further insight and understanding from the information available from social networking sites, such as Twitter.

Appendix A: Example MATLAB Code

```
% Test Twitty call account
% Clear variables
clear;
% Set up the credentials based on Twitter account
credentials.ConsumerKey = 'YourKey';
credentials.ConsumerSecret = 'Yoursecret';
credentials.AccessToken = yourAccess;
credentials.AccessTokenSecret = 'youraccesssecret';

% Create Tweet instance based on credentials
% tw = twitty(credentials);
% Create a Tweet example
% Pick Topeka Kansas for Center of Continental U.S.
% Set the last ID to Any value to start
LastID = 123456;
LastIDQ2 = 123456;
NumIts = 180;
TweetCnt=0; % Initialize to 0 counts
TweetCntQ2=0;
PauseTime = 60;
% Start a timer
Tic;
% Loop several times gathering data
for j=1:NumIts
j
% clear old data
clear S;
clear Q2;
% Time out issues establish new tw each iteration
tw = twitty(credentials);
% Run the First Query
S    =    tw.search('NFL',    'count',20,    'include_entities','true','geoco-
de','39.051300,-95.724660,1700mi','since_id',LastID);
% Let us see how many we got
StatCnt = length(cellfun('ndims',S{1,1}.statuses));
for i=1:StatCnt
 TweetCnt = TweetCnt+1;
 MyTweets=S{1,1}.statuses{1,i}.text ;
 % Put the data in
 Tweets{TweetCnt}=MyTweets;
 % Put the dates/time in
 Dates{TweetCnt} = S{1,1}.statuses{1,i}.created_at;
 % Gather other stuff
 FollowerCount(TweetCnt) = S{1,1}.statuses{1,i}.user.followers_count;
 FriendsCount(TweetCnt) = S{1,1}.statuses{1,i}.user.friends_count;
 TZones = S{1,1}.statuses{1,i}.user.time_zone;
 TimeZones{TweetCnt} = TZones;
 % Get screen name
 SName = S{1,1}.statuses{1,i}.user.screen_name;
```

http://dx.doi.org/10.5772/58895

```
 ScreenName{TweetCnt} = SName;
 % Get the User Mentions Count
  UserMentionCnt{TweetCnt}  =  length(cellfun('ndims',S{1,1}.statuses{1,i}.enti-
ties.user_mentions));
  URLMentionCnt{TweetCnt}  =  length(cellfun('ndims',S{1,1}.statuses{1,i}.enti-
ties.urls));
 HashTagsMentionCnt{TweetCnt} = length(cellfun('ndims',S{1,1}.statuses{1,i}.en-
tities.hashtags));
 end

 % Get the lastID to eliminate Dups
 if (StatCnt > 0)
 LastID = S{1,1}.statuses{1,1}.id_str;
 end
 % Run the World Series Query
Q2  =  tw.search('World  Series',  'count',20,  'include_entities','true','geoco-
de','39.051300,-95.724660,1700mi','since_id',LastIDQ2);
% Let us see how many we got
StatCntQ2 = length(cellfun('ndims',Q2{1,1}.statuses));

for i=1:StatCntQ2
 TweetCntQ2 = TweetCntQ2+1;
 MyTweetsQ2=Q2{1,1}.statuses{1,i}.text ;
 % Put the data in
 TweetsQ2{TweetCntQ2}=MyTweetsQ2;
 % Put the dates/time in
 DatesQ2{TweetCntQ2} = Q2{1,1}.statuses{1,i}.created_at;
 % Gather other stuff
 FollowerCountQ2(TweetCntQ2) = Q2{1,1}.statuses{1,i}.user.followers_count;
 FriendsCountQ2(TweetCntQ2) = Q2{1,1}.statuses{1,i}.user.friends_count;
 TZonesQ2 = Q2{1,1}.statuses{1,i}.user.time_zone;
 TimeZonesQ2{TweetCntQ2} = TZonesQ2;
 % Get screen name
 SNameQ2 = Q2{1,1}.statuses{1,i}.user.screen_name;
 ScreenNameQ2{TweetCntQ2} = SNameQ2;
 % Get the User Mentions Count
     UserMentionCntQ2{TweetCntQ2}     =     length(cellfun('ndims',Q2{1,1}.sta-
tuses{1,i}.entities.user_mentions));
 URLMentionCntQ2{TweetCntQ2} = length(cellfun('ndims',Q2{1,1}.statuses{1,i}.en-
tities.urls));
    HashTagsMentionCntQ2{TweetCntQ2}    =    length(cellfun('ndims',Q2{1,1}.sta-
tuses{1,i}.entities.hashtags));
 end
 % Get the lastID to eliminate Dups
 if (StatCntQ2 > 0)
 LastIDQ2 = Q2{1,1}.statuses{1,1}.id_str;
 end
 % Pause a few seconds
 pause(PauseTime);
end
```

Appendix B: Example Data Analysis MATLAB M-File

```
% Combine the Data into a vector
measWS = [cell2mat(UserMentionCntQ2)' cell2mat(URLMentionCntQ2)' cell2mat(Hash-
TagsMentionCntQ2)' FollowerCountQ2' FriendsCountQ2'];
measNFL= [cell2mat(UserMentionCnt)' cell2mat(URLMentionCnt)' cell2mat(HashTags-
MentionCnt)' FollowerCount' FriendsCount'];
% Combine
meas = [measWS;measNFL];
% Now let's do some Stats and Clustering
[cidx2,cmeans2] = kmeans(meas,2,'dist','sqeuclidean');
figure;
[silh2,h] = silhouette(meas,cidx2,'sqeuclidean');
% Look for maximum number of clusters
[cidx3,cmeans3] = kmeans(meas,3,'display','iter');
% Now do some clustering an
figure;
[silh3,h] = silhouette(meas,cidx3,'sqeuclidean');
% Some plots Might want to plot with NFL versus World Series
% Hierarchical Clustering
eucD = pdist(meas,'euclidean');
clustTreeEuc = linkage(eucD,'average');
myCop = cophenet(clustTreeEuc,eucD)
figure;
[h,nodes] = dendrogram(clustTreeEuc,0);
set(gca,'TickDir','out','TickLength',[.002 0],'XTickLabel',[]);
% Reduce nodes
figure;
[h,nodes] = dendrogram(clustTreeEuc,12);
toc
% World Series tweet Gaps
for i=1:length(Dates)-1
time1 = [str2num(Dates{i}(26:30)) 10 str2num(Dates{i}(9:10)) str2num(Dates{i}
(12:13)) str2num(Dates{i}(15:16)) str2num(Dates{i}(18:19))];
time2 = [str2num(Dates{i+1}(26:30)) 10 str2num(Dates{i+1}(9:10))
str2num(Dates{i+1}(12:13)) str2num(Dates{i+1}(15:16)) str2num(Dates{i+1}
(18:19))];
deltaWorldSeries(i) = abs(etime(time1,time2));
end
% NFLtweet gaps
for i=1:length(DatesQ2)-1
time1 = [str2num(DatesQ2{i}(26:30)) 10 str2num(DatesQ2{i}(9:10)) str2num(Da-
tesQ2{i}(12:13)) str2num(DatesQ2{i}(15:16)) str2num(DatesQ2{i}(18:19))];
time2 = [str2num(DatesQ2{i+1}(26:30)) 10 str2num(DatesQ2{i+1}(9:10))
str2num(DatesQ2{i+1}(12:13)) str2num(DatesQ2{i+1}(15:16)) str2num(DatesQ2{i+1}
(18:19))];
deltaNFL(i) = abs(etime(time1,time2));
end
figure (1)
subplot(2,1,1)
plot (deltaNFL,'r');
```

http://dx.doi.org/10.5772/58895

```
ylabel('Time between Tweets')
xlabel ('Tweet Number')
title('"NFL" Time between Tweets');
grid;
subplot(2,1,2)
plot(deltaWorldSeries, 'b');
ylabel('Time between Tweets')
xlabel ('Tweet Number')
title('"World Series" Time between Tweets');
grid;
% Stats
% Quick histograms for the 2 delta times
xvalues = -1:2:800;
figure (2)
subplot (1,2,1)
hist(deltaNFL,xvalues);
grid;
title ('Time between "NFL" Tweets')
axis ([-1 200 0 200]);
subplot (1,2,2)
hist(deltaWorldSeries,xvalues);
grid;
title ('Time between "World Series" Tweets')
axis ([-1 200 0 200]);
% Box plots for Descriptive visualization
figure(3)
subplot(1,2,1)
boxplot(deltaNFL)
title ('Time between "NFL" Tweets')
grid;
subplot(1,2,2)
boxplot(deltaWorldSeries)
title ('Time between "World Series" Tweets')
grid;
% Group Scatter
% Put everything together
MyGroup1 = zeros(1,length(HashTagsMentionCnt));
MyGroup2 = ones(1,length(HashTagsMentionCntQ2));
MyGroups = [MyGroup1 MyGroup2];
A1 = cell2mat(HashTagsMentionCnt);
B1 = FollowerCount;
C1 = FriendsCount;
D1 = cell2mat(UserMentionCnt);
E1 = cell2mat(URLMentionCnt);
A2 = cell2mat(HashTagsMentionCntQ2);
B2 = FollowerCountQ2;
C2 = FriendsCountQ2;
D2 = cell2mat(UserMentionCntQ2);
E2 = cell2mat(URLMentionCntQ2);
MyVars = [A1' B1' C1' D1' E1'; A2' B2' C2' D2' E2'];
varNames = {'HashTagMention' 'FollowerCount' 'FriendCount' 'URLMention' 'Hash-
TagMention'};
```

```
figure(4)
gplotmatrix(MyVars(:,2),     MyVars(:,3),MyGroups,['r'     'b'],['O'     'X'],
[],'off','hist',['Followers'],['Friends']);
grid;
```

Author details

Kelly Bennett[1] and James Robertson[2]

1 U.S. Army Research Laboratory, Sensors and Electron Devices Directorate, Adelphi, MD, USA

2 Clearhaven Technologies LLC, Severna Park, MD, USA

References

[1] Rahman, M. Mining social data to extract intellectual knowledge. doi:10.5815/ijisa.2012.10.02

[2] Das TK., Kumar P. BIG Data Analytics: A Framework for Unstructured Data Analysis. International Journal of Engineering Science and Technology 2013;5(2) 153-156.

[3] Daehoon K., Daeyong K., Seungmin R., Eenjun H. Detecting Trend and Bursty Keywords Using Characteristics of Twitter Stream Data. International Journal of Smart Home 2013;7(1) 209-220.

[4] Karandikar A. Clustering short status messages: A topic model based approach. Master's thesis. University of Maryland; 2010. http://ebiquity.umbc.edu/_file_directory_/papers/518.pdf (accessed 29 October 2013).

[5] Perera R. Twitter Analytics: Architecture, Tools and Analysis, Proc. The 2010 Military Communications Conference, MILCOM2010, Oct. 31, 2010-Nov. 3, 2010, San Jose Convention Center, CA; 2010.

[6] Turner and Malleson (2011). Applying geographical clustering methods to analyze geo-located open micro-blog posts: A case study of Tweets around Leeds since June 2011, https://docs.google.com/document/d/1yaRBUwJy8Cb3JWlN-fJOZZRN1JnXvpdWzhRDstqwsoI/edit?pli=1#heading=h.rx5m14o2e6yw (accessed 30 September 2013).

[7] MathWorks. MATLAB Products and Services. Overview. http://www.mathworks.com/products/MATLAB/ (accessed 8 October 2013).

[8] MathWorks. MATLAB Central. File Exchange. twitty by Vladimir Bondarenko. 30 January 2012 (Updated 12 July 2013). Interface-class to access the Twitter REST API

v1.1. http://www.mathworks.com/MATLABcentral/fileexchange/34837-twitty/content/twitty.m. (accessed 21 September 2013).

[9] Twitter API. https://dev.twitter.com/docs/api/1.1 (accessed 12 July 2013).

[10] MathWorks. MATLAB Central. File Exchange. JSON Parser. JSON Parser by Joel Feenstra. 3 July 2008 (Updated 18 June 2009). http://www.mathworks.com/MATLAB-central/fileexchange/20565-json-parser. (accessed 23 September 2013).

[11] MathWorks. Statistics Toolbox. Overview. http://www.mathworks.com/products/statistics/ (accessed 1 October 2013).

[12] MathWorks. Neural Network Toolbox Overview. http://www.mathworks.com/products/neural-network/ (accessed 18 September 2013).

[13] Kohonen T. Self-Organizing Maps. 3rd ed. Springer Series in Information Sciences. Springer-Verlag Heidelberg: New York; 2001.

[14] Bennett K, Robertson J. Signal and image processing algorithm performance in a virtual and elastic computing environment, Proc. SPIE Vol. 8734, Active and Passive Signatures IV, 87340B (2013).

IntechOpen