A SYSTEM FOR CROWD ORIENTED EVENT DETECTION, TRACKING, AND SUMMARIZATION IN SOCIAL MEDIA

An Undergraduate Research Scholars Thesis

by

JASON M. BOLDEN

Submitted to Honors and Undergraduate Research
Texas A&M University
in partial fulfillment of the requirements for the designation as

UNDERGRADUATE RESEARCH SCHOLAR

Approved by
Research Advisor: Dr. James Caverlee

May 2013

Major: Computer Engineering

TABLE OF CONTENTS

ABSTRACT

A System for Crowd Oriented Event Detection, Tracking, and Summarization in Social Media. (May 2013)

Jason M. Bolden
Department of Computer Science and Engineering
Texas A&M University

Research Advisor: Dr. James Caverlee
Department of Computer Science and Engineering

In this research thesis, we investigate new methods for crowd-oriented event detection in social media (specifically Twitter). Specifically, we describe, evaluate, and suggest content based methods of extracting features that define events occurring in social media streams. Content-based methods examine the appearance of event-describing keywords, topical words, in a stream of tweets. With these aggregated features, tweets are then clustered using a parallelized version of canopy and k-means clustering in order to find groups of similar tweets which represent events. Tracking of events through time is done by evaluating the similarity of events in consecutive time periods. The effectiveness of the feature extraction stage is determined by the relevance of tweets to one another in event summaries. Our experiments aim toward finding the optimal parameters for feature extraction and event clustering.

DEDICATION

There are a select few people that I would like to dedicate the writing and completion of this thesis to, for without them I feel I would have never had the opportunity nor the knowledge to complete this document. Firstly, thank you Mother and Father for raising me to be the curious challenge seeking person that I am. If it weren't for your support I feel my hunger for knowledge would have been snuffed out long ago. Also I would like to thank Matthew Cestarte. Thank you Matt for letting me borrow that book on C++ when I was very young and mentoring me throughout my years of middle and high school. You introduced me to programming and computer science and our many coding projects together helped me learn quite a lot about things that I had never heard of in the realm of computing. As well, I thank the many teachers that have served as beacons throughout my career as a student. Thank you Mr. Steve Smiley and Mr. John Pinkerton. You both provided, to the best of your ability, learning environments that I could continue to grow in. You also taught me how to be an independent learner and that I don't necessarily need someone else to teach me what I want to learn. To Dr. Jyh-Charn Liu, thank you for taking a curious freshman into your lab. Through the early years of my undergraduate studies you taught me the importance of striving for excellence rather than just getting by. You gave me focus and if it weren't for you I feel I would have wasted away. Lastly, I would like to thank my current faculty advisor Dr. James Caverlee. You introduced me to this world of data analytics and mining in your Information Retrieval class. Since then you've invited me to your lab, introduced me to many great students and friends, and given me this great oppurtunity to write an undergraduate thesis under your guidance. To you all of you, with great love and sincerity, thank you.

Jason M. Bolden

NOMENCLATURE

IDF	Inverse Document Frequency, the inverse probability of a keyword appearing in a document in the time frame.
TF	Term Frequency, how many times this keyword appears in a tweet out of all other keywords the tweet.
PoH	Probability of being a Hashtag, this is a binary value being 1 if the keyword is a hastag, 0 otherwise.
Entropy	A measure of the uncertainty in a random variable, in our case a keyword.
RSS	Residual Sum of Squares, for measuring intra-cluster distance.
AWS	Amazon Web Services, suite of services which were used for our computations.
JSON	JavaScript Object Notation.
Job	a MapReduce job run on an Hadoop Cluster.

CHAPTER I

INTRODUCTION

Over the past decade, society has integrated their everyday life more tightly and deeply into the fabric of the Internet. With the growing popularity of web-services such as Facebook, Twitter, and Foursquare, this melding has been made increasingly easier. Just as one could observe the social interactions of people communicating face-to-face one could also analyze a stream of Twitter data to identify groups of interaction. Due to the large volumes of data generated by the previously mentioned services, this would equate to vast numbers of crowds of many varying topics. From a researchers standpoint this social-computing way of life could prove useful in crowd-oriented event detection. One could imagine a fire occurring in a office building, or the release of a new product, or even a politician speaking at a public venue. A party of interest could benefit greatly from the crowds of information that form around these events. Emergency responders could better prepare for the office fire if they could read or view the pictures taken at the scene before they ever arrived, businesses could target their advertisements more effectively based on where their product is best received by the public, and a politician can easily see what he or she should do improve campaigning efforts.

However, it is easy to see that this task is not without major challenges. For one, we are faced with the problem of how do we perform event detection on large-scale social media. We need some way of quantizing each social media update in a meaningful manner such that we can perform comparisons. There is no formal way of doing this so we must experiment with various techniques such as term frequency analysis, term entropy, and hashtag occurrence to try and best predict the topic a document belongs to. We are also taxed with the challenge of evaluating models. It is difficult to generate a ground truth set of documents that can be used for testing the effectiveness of a system. On a more practical matter, the amount of data generated by social media is absurdly large. Performing statistical analytics on data of this volume is not feasible nor possible using traditional means. In our application we address each of these problems and suggest alternatives to overcoming them.

In the rest of this thesis we describe our initial efforts investigating the viability of an end-to-end crowd-oriented system capable of event detection, tracking, and summarization. As an initial investigation, we focus primarily on the systems-oriented development aspect, the design of our preliminary algorithms, and some initial experimental evidence supporting our design choices.

At a high-level, there are four distinct steps performed at any given time period in our system. The first, feature aggregation, attempts to extract event describing attributes from the tweets within a given fifteen minute time frame. Once the sets of event describing features are collected, the tweets are quantized using their respective features and parallelized clustering is performed to identify crowds of similar tweets. The next step is to then create summaries of each detected event, and lastly tracking is done on the event through the span of time by comparing the similarity of events in consecutive time periods.

The following sections will describe these steps in greater detail along with methods used to improve the performance and accuracy of each procedure. We conclude with some final thoughts on the challenges inherent in such a crowd-oriented system, a discussion of the limitations of our approach, and we identify several opportunities for building on this thesis as future work.

CHAPTER II

METHODS

In this chapter we discuss the incremental steps of event detection in our system. Each step is very intricate and tightly dependent on the previous. Selecting the appropriate features determines cluster quality which is reflected in the summarization step and will lead to nonsensical result in the tracking step if done poorly. In the conclusion of this paper we will examine the results of each step and show how much each affects the other. For now we will overview each portion of the system. Figure II.1 will serve as a road map of our system.

Data Collection and Cleaning

As stated before all of our data is collected from Twitter using their Streaming API. This allows us to take a sample of all tweets generated across the world as they're created. This sampling is done every 15 minutes and is done continuously. In raw form, Twitter gives us each tweet in JSON format. This includes a lot of information not used in our calculation.

Tweet Tokenization and Word Filtering

For our analysis we consider geo-located tweets that were posted from within the United States throughout the month of May, 2012. To ease computation in our processing and analysis stage we do as many calculations with the raw Twitter data offline and store it in a MongoDB database. The records we store are individual tweet statistics, time period statistics, and keyword statistics. The steps taken to extract individual keywords from each tweet include:

1. Tokenize the contents of the tweet on whitespace characters and convert to lowercase,

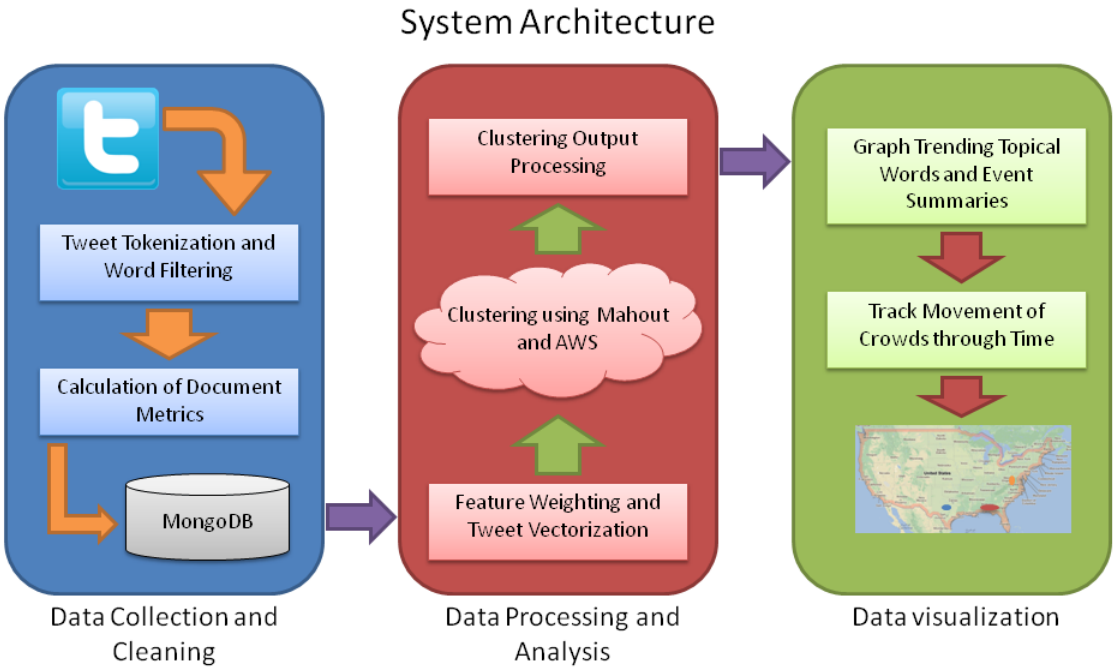

System Architecture
Tweet Tokenization and Word Filtering
Calculation of Document Metrics
MongoDB
Data Collection and Cleaning
Clustering Output Processing
Clustering using Mahout and AWS
Feature Weighting and Tweet Vectorization
Data Processing and Analysis
Graph Trending Topical Words and Event Summaries
Track Movement of Crowds through Time
Data visualization

Fig. II.1. Overall system architecture.

2. Disregard all tokens containing characters not found in the English 26 letter character set,
3. Disregard all tokens that are stop words; this also includes some Spanish stop words.

In our new data object for tweet statistics we couple together the tweet id, the contents of the tweet as it appears, the set of keywords in that tweet, the set of hashtags, geo-location, exact time post, the 15 minute time period in which it was posted, and keyword metrics.

Calculation of Document Metrics

Metrics such as TF, IDF, and Entropy can be stored as we receive tweets from Twitter. In our tokenization step we also take a count of how many times the keyword appears in the tweet and store a list of tuples, (keyword, TF), in every tweet statistics object. For the collection of keyword statistics we attach the keyword, PoH [1] value, time period for these statistics, document frequency, and an array of 7 document frequencies (for calculation of entropy). The array of document frequency is created in a way such that the center of the array is the document frequency of the word for the current 15 minute time period and on either side is document frequency of the previous and future time periods. This yields an array with the document frequency of a keyword for 105 consecutive minutes. Our time period statistics object contains the time period of the statistics, the total number of keywords, total number of hashtags, and total number of tweets. All of these objects are then stored in respective collections on a MongoDB database with appropriate indexing for fast querying.

Data Processing and Analysis

In this stage we process tweets and attempt to discover crowds from any 15 minute time period in our database. Before we can run any sort of analysis on the tweet we must define some method for quantizing them to a numerical and comparable value. For this we will define each tweet as a vector in a feature space where every axis in this space corresponds to the TF of an event describing feature. In the following we describe how we determine the importance of each feature in determining the existence of an event.

Feature Weighting and Tweet Vectorization

While it is true that we will extract words that imply the existence of an event we are still going to consider all keywords in the time period during the clustering step. The only words we do not consider are stop words. Stop words are common in the English and Spanish language that do little to add to the over all topic of the document. These are words such as “the”, “a”, “y”, “el”, etc. This is done to reduce the dimensionality of our feature space. Though we consider all keywords we also assign each word a certain weight which is calculated using the following three quantities. The goal is to emphasize the small number of tokens that are the most likely to describe an event and diminish the effect of other more common keywords.

IDF: This is a measure of the rareness of a token within a set of documents. Equation (II.1) shows the method we used to calculate this measure.

$$\mathrm{idf}_{t,d} = \log\left(\frac{N}{df_t}\right) \tag{II.1}$$

Where t is the term of interest in document d, N is the number of documents in the time period, and df is the document frequency of token t. If the token t occurs in a large number of documents then the logarithmic term approaches 0, thus lowering the token’s significance.

Entropy: The “burstiness” of a token within a given time frame. The motivation for this is the intuition that a word that is evenly distributed through time is less likely to describe an event over a word that has high frequency at one point but very lower frequency before and after that point in time. There are more formal ways of calculating entropy but our method simply observes the term frequency of a token appearing in the desired time period, three time periods before, and three time periods after. Equation (II.2) is then applied:

$$\mathrm{entropy}_t = \sum_{i=1}^{7} P_{t,i} * \log\left(P_{t,i} + 1\right) \tag{II.2}$$

$P_{t,i}$ is the probability of the token t appearing in a document on time period i. This value is maximized when the probability is not evenly distributed.

Probability of being a Hashtag (PoH): In the case of tweets, hashtags shed a great deal of information pertaining to the meaning of a tweet. With this fact hashtags are of more importance than plain text that appears within the tweet.

These three measures are used in equation (II.3) to give the token a weight. α, β, and γ are chosen experimentally to achieve the best set of weights. Metrics that are more important in defining an event can be discerned by the nature of the documents being clustered.

$$\text{weight}_t = \alpha * idf_t + \beta * \text{entropy} + \gamma * \text{poh} \tag{II.3}$$

We then create a vector, W_τ, which contains the weights for all keywords in the set K_τ where τ is the time period of we are performing event detection on. With this vector we can augment the TF vectors of the tweets in time period τ.

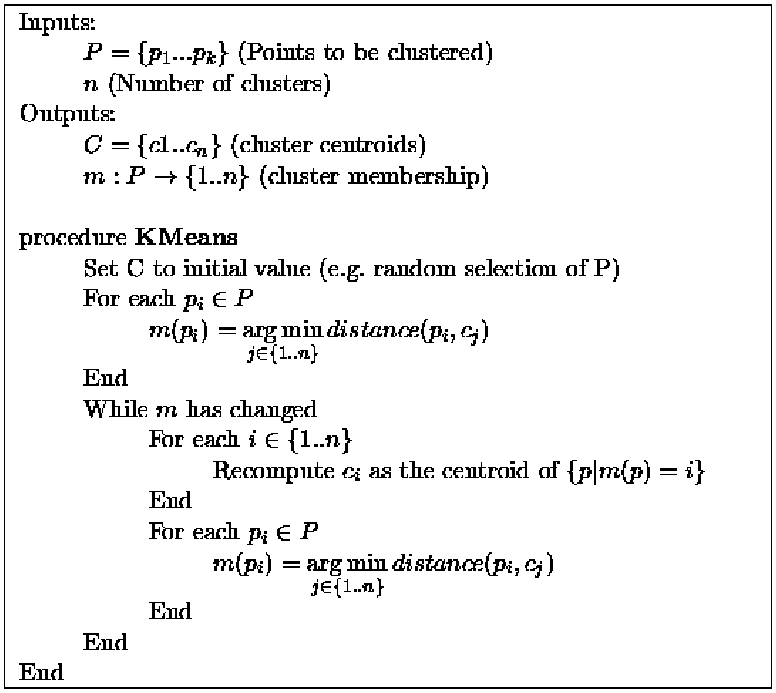
Clustering using Mahout and AWS

In order to perform a clustering algorithm we must first quantize the tweets in a vector form. To do this we define a vector space whose axes represent the term frequencies of the keywords in the set K_τ. This means that if there are 10,000 keywords in a given time period then our vector space has just as many dimensions. As stated before some word are of greater importance than others so we augment the document vectors with the weight vector as such:

$$V_d' = V_d * W_\tau^\intercal \tag{II.4}$$

Once the tweets have been transformed into a weighted vectorized form we begin clustering. Depending on the scale of the application one of two methods can be used. The first, sequential clustering methods, are best suited for a small datasets with small dimensionality, (eg., few keywords). These implementations limit the number of tweets that can be clustering in a reasonable amount of time, and the number of features that can be used for clustering is also limited resulting in hindered cluster quality. Though this is a cheaper method because it requires fewer resources, computationally it is very slow and not practical for our application. For this case, a parallelized clustering algorithm that utilizes cloud computing techniques is needed.

- Sequential K-Means: This clustering algorithm attempts to cluster the data set into *k* groups. Figure II.2 outlines the process. The initialization step involves generating *k* random vectors, seeds, as initial guesses or choosing *k* random vectors from the set for the cluster centroids. Then the iterative process begins by grouping each document with its closest centroid and recalculating the new centroid for every group by averaging. This process is repeated using the newly calculated set of centroids until convergence, when the centroids stop changing position. The closeness of the documents can be calculated using various distance measures; we examine two.



```
Inputs:
        $P = \{p_1...p_k\}$ (Points to be clustered)
        $n$ (Number of clusters)
Outputs:
        $C = \{c1..c_n\}$ (cluster centroids)
        $m : P \to \{1..n\}$ (cluster membership)

procedure KMeans
        Set C to initial value (e.g. random selection of P)
        For each $p_i \in P$
                $m(p_i) = \arg\min_{j \in \{1..n\}} distance(p_i, c_j)$
        End
        While $m$ has changed
                For each $i \in \{1..n\}$
                        Recompute $c_i$ as the centroid of $\{p | m(p) = i\}$
                End
                For each $p_i \in P$
                        $m(p_i) = \arg\min_{j \in \{1..n\}} distance(p_i, c_j)$
                End
        End
End
```

Fig. II.2. Sequential K-means Algorithm.

Euclidean distance: This is a measure of the vector difference between two documents. This method suffers from a naïve drawback if non-normalized vectors are used. Say vector a and b are being compared to c. a is twice the magnitude of b and is thus closer or further away from c than b is, depending on c's coordinates.

Cosine similarity: This is a measure of the angle between to vectors. Obviously the smaller the angle between the vectors, the more similar they are to each other in content.

The challenge in this algorithm is determining the correct value for k. There are technique that determine k by examining it's effect on the RSS of the clustering output.

$$\text{RSS} = \sum_{c \in C_\tau} \sum_{V' \in c} \left(V' - \mu_c \right)^2 \tag{II.5}$$

The value of k is not the only parameter that affects clustering quality. The initial seeds also have a large effect on the final output. To remedy this we use Canopy clustering as a precursor to K-means; more on this in the following.

- Parallelized Clustering: Mahout is a machine learning library containing various parallelized implementations of clustering algorithms that can be run on a Hadoop cluster. Hadoop is a cloud computing framework used for distributing the load of large job across multiple machines. It utilizes a computing framework called MapReduce which consists of mappers and reducers. Hadoop and MapReduce are integral in performing large scale analytics on the data we are processing. We utilize AWS's Elastic MapReduce services in order to run our jobs. The clustering algorithms we use are Canopy and K-means, provided by the Mahout library.

  As mentioned before, Canopy clustering [2] is used as a way of speeding up K-means clustering and also calculated a value for K. This algorithm take two parameters t_1 and t_2 where $t_1 > t_2$. The tweets in our feature space are partitioned into canopies in the following way:

1. Selecting a random tweet from the set of tweets in the feature space as a candidate for a canopy center,
2. Measuring its distance to all others tweets in the feature space,
3. Disregard all tweets within t_2 of the candidate as potential centers,
4. Repeat until there are no more candidates to consider.

The set of centers are then used as input to K-means as the seeding centroids, and k is simply the size of this set. In order to determine the ideal threshold values we ran an experiment in which we clustered our data while sweeping all possible values for t_1 and t_2 in steps of 0.5 of a degree from 0 to 90. We then graphed the RSS and number of singleton clusters. The goal is to minimize the number of singletons, a result of having lax threshold values, while also minimizing the RSS, a result of very strict threshold values.

Clustering Output and Processing

With the output of the clustering process we can begin to fine tune our weighting vector. We ran another experiment to try and determine how effective our three metrics are at describing an event. We have clustered 4 sets of feature vectors using varying values for parameters α, β, and γ. This was done in a way in which we considered each metric solely and the 4^{th} set had equal importance of all three metrics: values of 0.33. We then looked at the average number of non-singleton clusters, RSS distribution, and inter-cluster distance distribution. With these results we hope to determine the significance of each metric on clustering output.

Data Visualization

In this stage we consider methods for visualizing our cluster output. A lot of these are futures works yet to be tested but still need to be mentioned in order to express the significance of this system.

Graph Event Summaries

Up to this point we've discussed methods for quantizing the tweets, clustering them into crowds, and tracking their lifespan and position. Unfortunately these events are just masses of data with no obvious meaning. We must define a method that ranks the tweets from most to least informing. The method we suggest for ranking is again a simple cosine similarity. Tweets in the cluster that are closest to the event center best define the topic of the event.

Tracking Event Movement

Tracking of events would be useful in many applications. For instance, if our system were looking for events related to the release of a new consumer product and the manufacturers were interested in how it was being received by the public, they could utilize our tools to track how their event moves through time and space. Experimentation could be made as to determine what factors makes their product spread the fastest and remain popular for the longest amount of time.This is made possible by looking for similar clusters in adjacent time periods. Groups of similar clusters would form an event chain.

Cosine Similarity: This method was used in the clustering step to compare tweets to cluster centroids. The same can be applied when comparing cluster centroids of one time period to those of the next. In order to declare two events as being equal the angle between their centroids must meet a predefined threshold. This value is not well defined so for the time being we are assuming that 80% similarity is sufficient. The case of an event being 80%

similar to more than one event in the following time period is unlikely because that would be a result of split cluster in the following time period (these clusters should be merged into one cluster).

CHAPTER III

RESULTS

The following are the results of the experiments mentioned in chapter II. A few of the process described in the methods chapter where not tested nor implemented due to error propagation. As mentioned before each step in this system is additive. Since we are still testing feature weighting, we should not expect to see good results for tracking or summarization.

Ideal Threshold Testing

As mentioned in the previous chapter, we need to find ideal values for the threshold values t_1 and t_2. Figure III.1 depicts the effect of the threshold values on the intra-cluster distances. The same can be seen in figure III.2 in the case of singletons. It is obvious that the two values are minimized at opposite extremes, so what we did in figure III.3 is normalized the previous graphs by their maximums and took their difference to determine what threshold values minimized both quantities. We found that $t_1 = 80$ and $t_2 = 45$ was a suitable selection and ran all other test using these parameters. This was done with the evenly weighted feature space, $\alpha = \beta = \gamma = 0.33$

Feature Weighting

When we clustered the feature spaces that consider PoH and Entropy only, we only generated 2 and 1 clusters. However, clustering the features space that considered IDF only and the space with equal consideration produced close to the same number of clusters. Figure III.4 shows the distribution of non-single clusters in each feature space. We began to suspect that IDF may have a major influence in feature weighting since these distributions are so similar

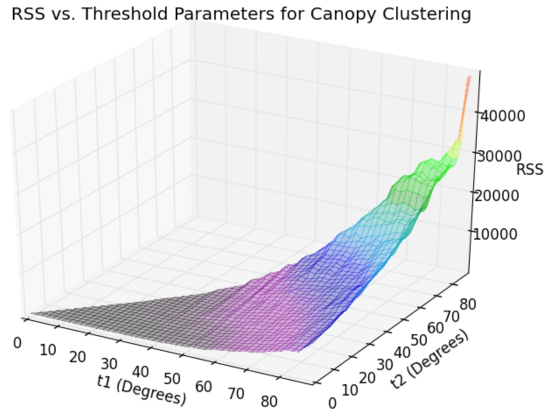


Fig. III.1. RSS for various values of t_1 and t_2. Notice the values for which the graph is minimized.

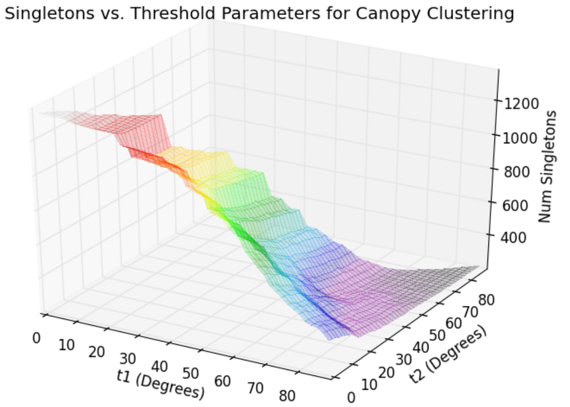


Fig. III.2. Total number of singletons for various values of t_1 and t_2. Again notice the values for which the graph is minimized.

while the other two spaces had completely incomparable distributions. Figure III.5 is also oddly similar. However the pattern ends when we examine figure III.6. In this graph we see that the equally weighted feature space has a much tighter distribution of RSS values. Either PoH, Entropy, or both have some effect on the clustering output that decreases intra-cluster

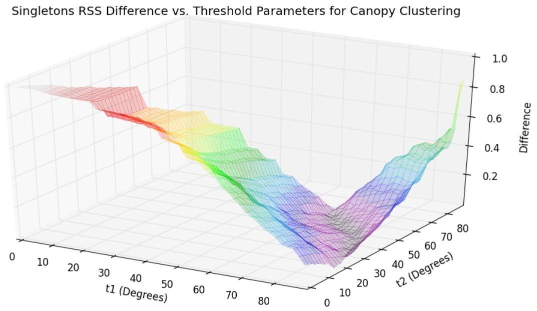


Fig. III.3. Sequential K-means Algorithm.

differences. Regardless of two spaces is it important to note that a large majority of the

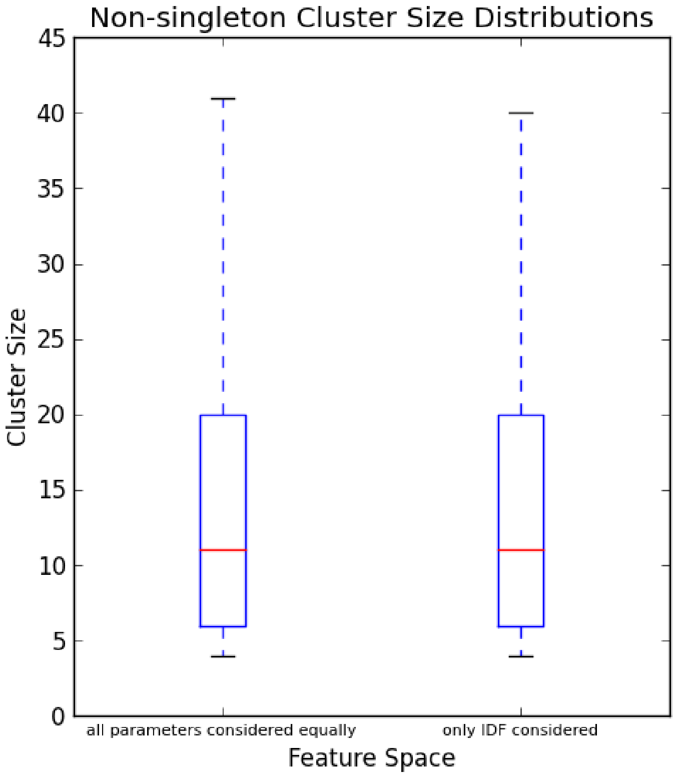


Fig. III.4. Strangely these distributions are very similar.

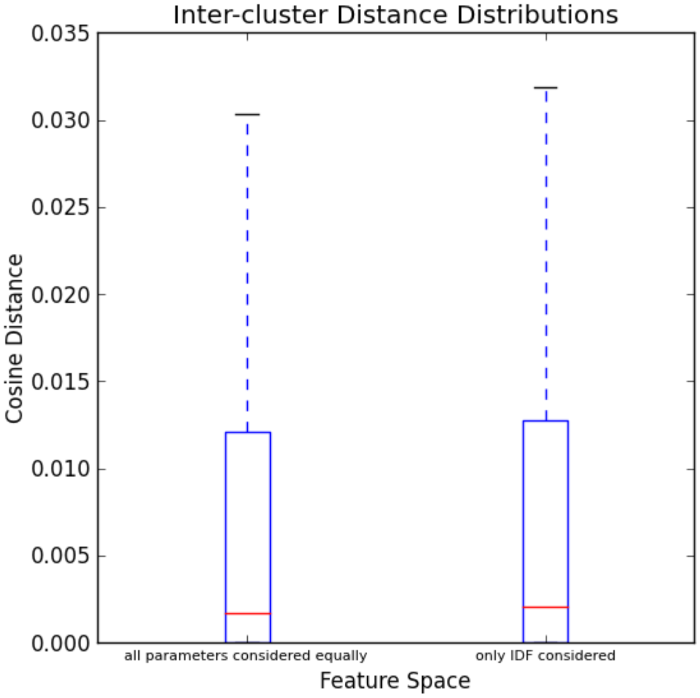


Fig. III.5. Again, strangely these distributions are very similar.

clusters have an inter-cluster distance of greater than 87 degrees. This means that most clusters are more or less "orthogonal" in topic to one another. This is significant evidence that our feature space contains definite and clear crowds within the time period.

Clustering Output

Here is a sample event that was extracted using our system from the time period 2:00 to 2:15 on May 3rd, 2012:

```
Tweet ID: 197871422360657920
"@jeremysarber i get that and see the value. my concern is the movement to find injustice where there truly is none. fear breeds fear."
"dTermFreqs" : { "none" : 1, "movement" : 1, "value" : 1, "truly" : 1, "injustice" : 1, "fear" : 2, "concern" : 1, "breeds" : 1 }

Tweet ID: 197868093563863040
"bigg knutt/no fear out here.. #filipinoheritagenight #sfgiants http://t.co/jupbzq5k"
dTermFreqs" : { "knuttno" : 1, "bigg" : 1, "fear" : 1, "sfgiants" : 1 }

Tweet ID: 197869253548978176
```

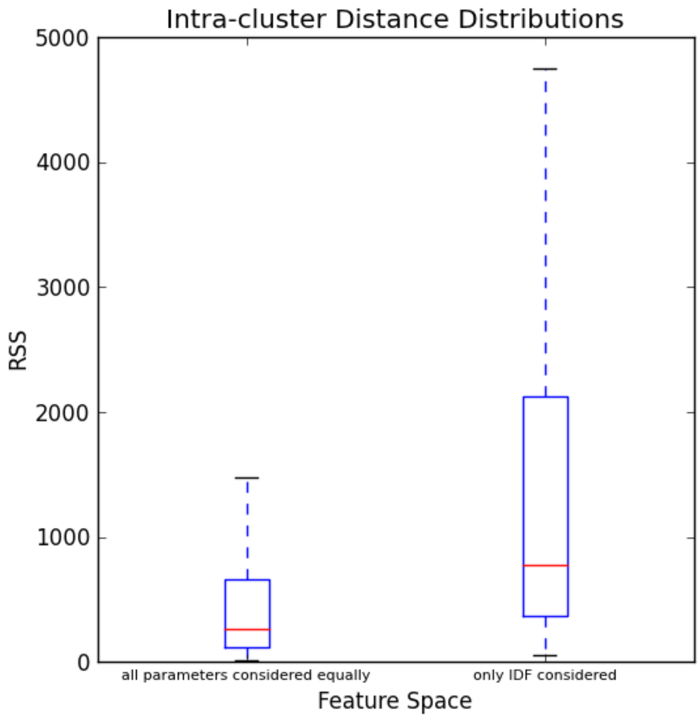


Fig. III.6. The equally weighted space has a tighter distribution.

```
"when in doubt, nip it out. #youdontknowwhatthismeans #donttrytofigureitoit #noitsnotaboutboobs"
"dTermFreqs" : { "nip" : 1, "doubt" : 1, "noitsnotaboutboobs" : 1 }

 Tweet ID: 197869098384891904
"contents" : "yu show me all your dreams, &amp; im here to help you get to the top were everything you do brings no doubt &amp;or fear"
"dTermFreqs" : { "help" : 1, "brings" : 1, "top" : 1, "doubt" : 1, "im" : 1, "amp" : 1, "ampor" : 1, "yu" : 1, "fear" : 1, "dreams" : 1 }
```

As you can see there are four tweets in this event. The second line of we tweet indicates the complete contents of the tweet while the third is a dictionary of the filter keywords and their TF. It can be inferred that this cluster center is somewhere about the “doubt” and “fear” plane. Though we did cluster these tweet based on similar content, the content did not fully convey the meaning of the tweets. This led to a cluster that didn’t make much sense, however there are hundreds more that we must search through to determine the efficiency of our system.

CHAPTER IV
CONCLUSION

Our system is still a long way from our end goal. Feature weighting needs to be explored in more detail, different clustering algorithms should be tested, and various other methods should be researched. The following includes a list of issues we faced in our experiments, suggestions for improvement, and other avenues we should consider in future work.

Keyword Selection

As we saw in the example cluster, the events have distinct keywords in common. However similar keywords don't exactly imply similar topic. To improve this we suggest using k-grams instead of single words. K-grams consider groups of k consecutive words as a token. For example, 2-grams of "The cat ran home." would be:

- "the cat"
- "cat ran"
- "ran home"

This scheme implies that grouping of words are more important than the a single word. This is intuitive considering, for example, that "I'm lovin it." is associated with the fast food chain McDonalds. Tests would be run to see what value for k yields the most meaningful event clusters.

Also Mahout provides word stemming functionality. This will allow for us to take into account the many ways a single word could potentially be spelled. "helloooo" and "hello" would be hashed to the same root word. This would cut down on feature space dimensionality and hopefully improve cluster quality as well.

Data Collection and Partitioning

Currently we collect tweets from all around the world. Since the number of tweets generated at one instance across the world is too large to completely retrieve, Twitter only gives us a sample. Our experiments are run over tweets posted from within the United States, so we are testing on a sample of a sample. This can lead to errors in our data since it does not accurately represent what is happening in the United States consistently. What we can do to remedy this is tell Twitter, through their API, that we only want tweets within a geo-located bounding box. This will increase the tweet density within the United States thus improve our results.

It also may benefit to change our time period length. Originally 15 minutes was chosen in order to limit the number of tweets to cluster and the size of the feature space, since it scales with the number of keywords. With the consideration of stemming and k-grams, we may be able to increase the time period to 30 minutes or maybe an hour. Entropy and PoH my change in significance since they are more readily to change in frequency over larger time intervals and not small ones

Feature Weighting

It is still unknown as to why the equally considered weighting parameters gave such a drastically smaller variation and distribution of RSS values. There seems to be some property of Entropy or PoH which either "pulls" the feature vectors closer together. Further experimentation must be done to determine the exact cause.

Summarization

Our current method of displaying tweets in order of closeness to cluster centers is too naïve of an approach to this problem. Instead it would be more effective to display the keywords, or k-grams, in the feature space that are major components of the event's centroid. We could also take into account other meta data attached to the tweet such as the user. If we have enough of a particular user's history we would also have a history the events he's been a part of. An assumption can be made that if a known user x is a part of an event then there may be some probability that the event pertains to a topic that the user has been a part of in the past.

Crowd Evolution

We could not run experiments for this step because of the amount of error that would propagate to the end results due to clusters with nonsensical topics. However, we can infer that using cosine similarity would be just as effective as when it was used for the clustering stage. Also what we can consider is the population of users in the cluster. If the population stays the same then the event are potentially similar. Interesting dynamics that should be studied are how much does and event change in long event chains; how similar are the first and last event clustering in a chain of similar events 10 time periods long.

REFERENCES

[1] R. Long, H. Wang, Y. Chen, O. Jin, and Y. Yu, “Towards effective event detection, tracking, and summarization on microblog data,” 2011.

[2] A. McCallum, K. Nigam, and L. H. Ungar, “Efficient clustering of high-dimensional data sets with application to reference matching,” Online; accessed 5 April 2013.