Clustering and Topic Analysis

Final Report

7 December 2016

CS 5604 Information Storage and Retrieval

Virginia Polytechnic Institute and State University

Blacksburg, VA

Fall 2016

Submitted by

Abigail Bartolome abijbart@vt.edu

Md Islam mdi7@vt.edu

Soumya Vundekode soumyav@vt.edu

Additional Support From

Eric Williamson ericrw96@vt.edu

Instructor

Professor Edward A. Fox

Abstract

The IDEAL (Integrated Digital Event Archiving and Library) and Global Event and Trend Archive Research (GETAR) projects aim to build a robust Information Retrieval (IR) system by retrieving tweets and webpages from social media and the World Wide Web, and indexing them to be easily retrieved and analyzed. The project has been divided into different segments - Classification (CLA), Collection Management (tweets - CMT and webpages - CMW), Clustering and Topic Analysis (CTA), SOLR, and Front-End (FE).

In building IR systems, documents are scored for relevance. To assist in determining a document's relevance to a query, it is useful to know what topics are associated with the documents and what other documents relate to it. We, as the CTA team, used topic analysis and clustering techniques to aid in building this IR system. Our contributions are useful in scoring which documents are most relevant to a user's query. We ran clustering and topic analysis algorithms on collections of tweets to identify the most discussed topics and grouped them into clusters along with their respective probabilities. We also labeled the topics and clusters.

This report covers the background, requirements, design and implementation of our contributions to this project. We evaluate the quality of our methodologies and consider improvements or future work that could be done on our project. Furthermore, we include a user manual and a developer manual to assist in any future work that may follow.

Table of Contents

List of Figures……..……... 4

List of Tables …… 5

I. Overview……... 6

II. Literature Review…….. 7

A. Introduction to Information Retrieval…… 7

B. Text Data Management and Analysis……... 8

C. DBSCAN…….. 9

D. Latent Dirichlet Allocation……... 9

E. Integrating Document Clustering and Topic Modeling……..... 9

F. Document Clustering for IDEAL…….. 10

G. Extracting Topics from Tweets and Webpages for the IDEAL Project..... 10

H. Clustering and Social Networks for IDEAL…… 10

III. Requirements……... 12

A. Cleaning……..... 12

B. Clustering…… 13

C. Topic Analysis…… 13

D. External Expectations…….. 14

E. Internal Project Exploration……..... 15

IV. Design……... 16

V. Implementation……..... 19

A. Selecting a Clustering Algorithm……... 19

B. Topic Analysis…… 21

C. Cluster Labeling…….. 23

VI. Timetable…….. 26

VII. User Manual …….. 27

A. K-Means……... 27

B. Topic Analysis……. 28

C. Cluster Labeling…….. 29

D. Pulling Tweets from HBase…….……. 30

E. Writing Results to HBase…….……. 31

VIII. Developer Manual……... 34

A. Clustering……. 34

B. Topic Analysis…….. 35

C. Cluster Labeling and Probabilities……..... 36

D. Pulling from HBase…….. 38

E. Writing to HBase…….. 38

F. File Inventory……... 39

G. Future Work…….. 39

IX. Acknowledgements…….. 41

X. References……..... 42

List of Figures

Figure 1 An Illustration of Flat Clustering and Hierarchical Clustering……..…… 7

Figure 2 Sample Database Plots……... 9

Figure 3 Workflow Diagram……. 12

Figure 4 Manual Clustering Distribution……... 18

Figure 5 Tweets Distribution Over Clusters ……. 20

Figure 6 Cluster Probabilities……..... 20

Figure 7 Topic Probabilities for New York Firefighter Shooting……..... 22

Figure 8 Distribution of Topics for New York Firefighter Shooting…… 22

Figure 9 Sample Clustering Input ……. 27

Figure 10 Sample Clustering Output……. 28

Figure 11 Sample LDA Output ……. 29

Figure 12 Sample LDA Document Output ……... 29

Figure 13 Cluster Labeling…….. 30

Figure 14 Cluster Probabilities…….. 30

Figure 15 Mean Frequency Matrix …….. 37

List of Tables

Table 1 Work Assignment……... 6

Table 2 Schema for Expected Output…….. 14

Table 3 Manual Clustering Results ……. 17

Table 4 Topic Analysis Results ……. ….. 23

Table 5 Clustering Result Summary …… 24

Table 6 Timetable……. 26

Table 7 Clustering Sbt Configuration for Clustering…… 27

Table 8 Sbt Configuration for Topic Analysis…….. 28

Table 9 Sbt Configuration for Pulling Tweets from HBase…….……... 31

Table 10 Sample Output for Pulling Tweets from HBase…….…….. 31

Table 11 Real World Event Decimal Mapping ……. 32

Table 12 Sbt Configuration for Writing Result to HBase ……... 32

Table 13 Installation Verification……..……... 34

Table 14 Cluster-Topics Array……..……... 37

Table 15 File Inventory……..…….…… 39

I. Overview

The Information Storage and Retrieval course aims to build a state-of-the-art information retrieval (IR) system to support the Integrated Digital Event Archiving and Library (IDEAL) and Global Event and Trend Archive Research (GETAR) projects. The IR system should be capable of searching webpages and/or tweets based on their full text, searching based on facets connected with metadata, and searching based on facets connected with the type of event, topics implicit in the collection, and groupings inherent in the data. The IR system should rank search results, consider social networks in page indexing, and eliminate non-relevant documents. The semester's work is divided into six teams: Classification (CLA), Collection Management for Tweets and Webpages (CMT and CMW), Clustering and Topic Analysis (CTA), Apache Solr (SOLR), and Front-end (FE).

As the CTA team, we were responsible for running clustering and topic analysis algorithms on approximately 1.4 billion tweets to improve the relevance and efficiency of an Information Retrieval system. The CLA team classified the tweets into real world events. We found the topics associated with each real world event. Then, we clustered the documents for each real world event and used the topic analysis results to label the clusters. The topic analysis results and clustering results were then reported to HBase for the SOLR team to use.

Task	Primary Developer
Clustering tweets using K-means and incorporating metadata into feature vectors	Eric Williamson Soumya Vundekode
Cluster labeling	Soumya Vundekode
Manually clustering tweets	MD Islam
Clustering tweets using DBSCAN	Eric Williamson
Latent Dirichlet allocation and automated labeling	Abigail Bartolome
HBase integration	Abigail Bartolome

Table 1: Work Assignment

II. Literature Review

A. Introduction to Information Retrieval

Chapters 16 and 17 of the course textbook, *Introduction to Information Retrieval*, were extremely useful for explaining the differences between flat clustering and hierarchical clustering, as well as the various techniques for implementing those two types of clustering. Flat clustering creates a set of clusters such that the clusters have no relationship to one another, whereas hierarchical clustering creates a tree-like relationship of clusters. Flat clustering allows the developers to pre-select the number of clusters in the corpus. Algorithms for flat clustering include K-means clustering and DBSCAN. Hierarchical clustering does not require the developers to pre-select the number of topics, however, this type of clustering is also less efficient. Algorithms for hierarchical clustering include hierarchical agglomerative clustering, a bottom-up clustering algorithm, and the top-down approach called divisive clustering [5]. We initially thought that hierarchical clustering would be best for our project given that it is more informative about the relationship between clusters. However, given that our tweet collections are already grouped by event, the cost of efficiency to implement hierarchical clustering in the project seemed unnecessary. We made the decision to focus on flat clustering, where we would compare K-means clustering and DBSCAN.

K-means clustering represents the documents as real-valued vectors and clusters the documents based on seeded points in a cluster, which we will refer to as centroids. The K-means clustering algorithm reassigns documents to the cluster with the closest centroid and recomputes the centroid based on the current members of its cluster. This process is iterated until either: (a) a fixed number of iterations has been completed, (b) the reassignment of documents to clusters no longer changes between iterations, (c) centroids do not change between iterations, or (d) the residual sum of squares (RSS) falls below a prespecified threshold [5].

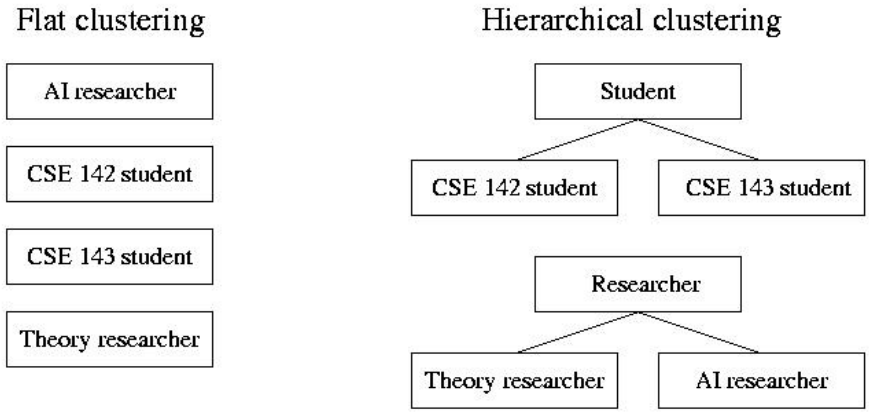

Figure 1. An Illustration of Flat Clustering and Hierarchical Clustering [1]

For the clusters to be of use for human users, they must be labeled so that the topics within each cluster can be known. One approach to this is differential cluster labeling where the cluster labels are made by comparing the term distribution between clusters. Differential cluster labeling identifies labels that characterize one cluster in contrast to other clusters. It is important to note that some words in the English language are rarely used, for example, the word didacticism, the idea that art and literature should be instructive as well as pleasurable. The presence of the word "didacticism" does not necessarily mean that it is a topic in the cluster, rather, it just so happens to contain a rare term. If "didacticism" is identified as a feature in two documents of the cluster containing 100 documents, it should not necessarily be a label of the cluster. Instances such as this should be considered in differential cluster labeling. In cluster-internal labeling, the clusters are labeled based solely on the terms internal to the cluster without considering the other clusters. While the labels will reflect the topics within the cluster, they may be duplicated by terms that are present throughout the collection of documents [5].

B. Text Data Management and Analysis

Chapter 14 of *Text Data Management and Analysis* focuses on text clustering— term clustering as well as document clustering. Term clustering focuses on grouping related terms, such as 'soccer' and 'basketball' because they are sports, or 'assessment' and 'evaluation' because they are synonyms. Document clustering groups documents either by their similarity or their latent structure. Similarity-based clustering is a type of hard clustering such that each document can only belong to one cluster. Model-based clustering captures the latent structure of the data to design a probabilistic model and forms clusters based on that model. This is a type of soft clustering that allows documents to belong to more than one cluster [10]. Our collections of tweets are grouped by event. Since we knew that a collection already has the event as a common label, we decided to use hard clustering for our project. This gave us distinct differences between the clusters.

One similarity-based clustering method that interested us is a top-down divisive approach to clustering. This iteratively applies a flat clustering algorithm to partition the data until it meets the criteria for it to stop partitioning. Our primary interest was in the K-means algorithm which identifies K number of clusters and iteratively assigns a document to each cluster until the change in cluster assignment is minimal [10].

Chapter 17 of *Text Data Management and Analysis* focuses on topic analysis and was a very helpful guide in Latent Dirichlet Allocation. We define a topic as the main idea discussed in a document, which is represented as a distribution of words. We can look at the task of topic analysis as two distinct parts: discovering topics and seeing which documents fit which topics. Topics can be defined as terms such as science or sports; when defined this way, we can see the occurrence of the terms within the document. We

can score the terms we choose as topics by using TF-IDF weighting so that the topics chosen will be terms that are fairly frequent but not too frequent. Another way to represent the topics is as a word distribution that allows the topics to be more expressive and deal with more complicated topics. This is normally done by making a probability distribution over the words. By using LDA we will represent the topics that are representative of this set as a probability distribution. Additionally, LDA can be used as a generative model to apply to new unseen documents [10].

C. DBSCAN

Another clustering approach, DBSCAN, clusters based on the density of points. The algorithm groups sets of points that are tightly packed together in high density regions while eliminating points in low density regions as noise. This had advantages over K-means clustering as it allows clusters to have arbitrary shapes. Consider Figure 1, where the document clusters can form irregular shapes but their density leads to a natural clustering of the data. DBSCAN does not require an explicit definition of the number of clusters and so it gives the optimal number of clusters for the given parameters [3].

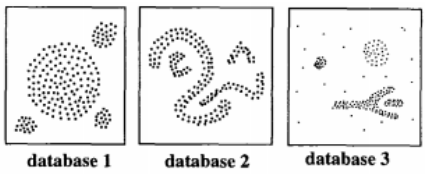

Figure 2. Sample Database Plots

D. Latent Dirichlet Allocation (LDA)

Latent Dirichlet allocation is a popular topic modeling approach in natural language processing, so we decided to read "Latent Dirichlet Allocation" by David M. Blei, Andrew Y. Ng, and Michael I. Jordan. It was useful in giving us an in depth understanding of how LDA works. We define a topic to be a distribution of words. LDA is a probabilistic model of a corpus and treats the documents as random mixtures over latent topics. LDA uses a three-level model with a topic node that is repeatedly sampled, thus allowing documents to be associated with multiple topics [2].

E. Integrating Document Clustering and Topic Modeling

This paper discusses how when used jointly, topic modeling and clustering will have much higher cluster accuracy than if used separately from each other. It proposes that semantics discovered by topic models can effectively facilitate accurate similarity measure, which is helpful in obtaining coherent clusters. In addition, they found that by using them jointly, clustering can help us infer more coherent topics and separate topics into group specific and group independent topics [9].

We chose to use our topic modeling on each of our clusters to help with the labeling of the clusters. We implemented this by mapping the topics produced by our topic modeling tool back to the documents in the cluster, so we were able to determine which topics belonged to which documents. Then, by knowing which topics belong to each document, we used the topics to label the clusters.

F. Document Clustering for IDEAL

The Spring 2016 clustering team used silhouette scores to evaluate the cluster results [4]. Silhouette scores measure how similar the documents are within each cluster and how dissimilar the documents are in different clusters. In the previous project, they obtained positive silhouette scores for all of the satisfactorily clustered datasets. Due to high sparsity in the dataset, the silhouette scores were low. However, feature transformation methods like Latent Semantic Analysis (LSA) can be applied to transform the dataset into lower dimensional space, thus decreasing the sparsity and increasing the silhouette scores. In the last project, it was found that the silhouette scores of the webpages were higher than those of the tweets. This was because the lengths of the webpages were much longer than those of the tweets. We planned to use silhouette scores to evaluate our clustering results as well.

G. Extracting Topics from Tweets and Webpages for the IDEAL Project

The Spring 2016 topic analysis team used Apache Spark's MLlib library to build the LDA model from the preprocessed data [8]. LDA was used to get a topic distribution for each of the documents and words contained in each topic. From this topic distribution a document matrix can be used to determine the most likely topics that these documents fall under. The former Topic Analysis team was able to automate the process for labeling these topics by selecting the most relevant word for the topic label. If the most relevant word had already been claimed by another topic, the second most relevant topic was used. Due to the large number of collections of tweets, we wanted to employ an automated method for topic labeling and chose to use this method.

H. Clustering and Social Networks for IDEAL

In "Clustering and Social Networks for IDEAL", they used K-means algorithm for clustering which takes the number of clusters, number of iterations and the distance measure as inputs [7]. The time complexity of the K-means algorithm is $O(knI)$, where k is the number of clusters, n is the number of objects and I is the number of iterations. They primarily focused on using hashtags (#) and username mention (@) in the tweets as the grouping attributes for feature extraction.

They used two approaches for evaluating the clustering algorithm. The first approach was to use the Word2Vec model for vector generation since it provides a more uniform clustering compared to the TF-IDF based model. They used Within Set Sum of Squared

Errors (WSSE) for deciding the optimum number of clusters in the dataset. The second approach was to use the output of topic analysis to check how well the documents have been clustered. The output of topic analysis represented a document-topic matrix and that of clustering represented a document-cluster matrix. Data from both matrices was aggregated by picking one third of topics with highest weightage for every document and grouping the documents by clusters. A mean topic frequency matrix over all clusters individually was computed and the deviation of each topic from the topic frequency matrix across all documents in a cluster was calculated. Once it was done for all clusters, the top 3 topics with least deviation were picked for each cluster. This evaluation process helped (a) figuring out the probability of a document belonging to a cluster, (b) cluster labeling, and (c) hierarchical mixture of topics and clusters. They calculated the probability of a document belonging to a cluster by adding the original probability weights of topics for a document with respect to the cluster it belongs to.

For our evaluation, we used the Word2Vec model since the previous team had already proven it to be better than the TF-IDF model [6]. We analyzed the clustering and topic analysis results by using their second approach to see how relevant and tightly coupled each topic is to the cluster. We calculated the probabilities of documents belonging in each cluster using the same approach.

III. Requirements

Our group focused on clustering and topic analysis on tweets. Clustering will be used to provide retrieval speedups, as well as give additional data for scoring the results. Topic modeling will focus on providing a label to these collections so they can be indexed by the SOLR team and used as a facet for the Front-End team.

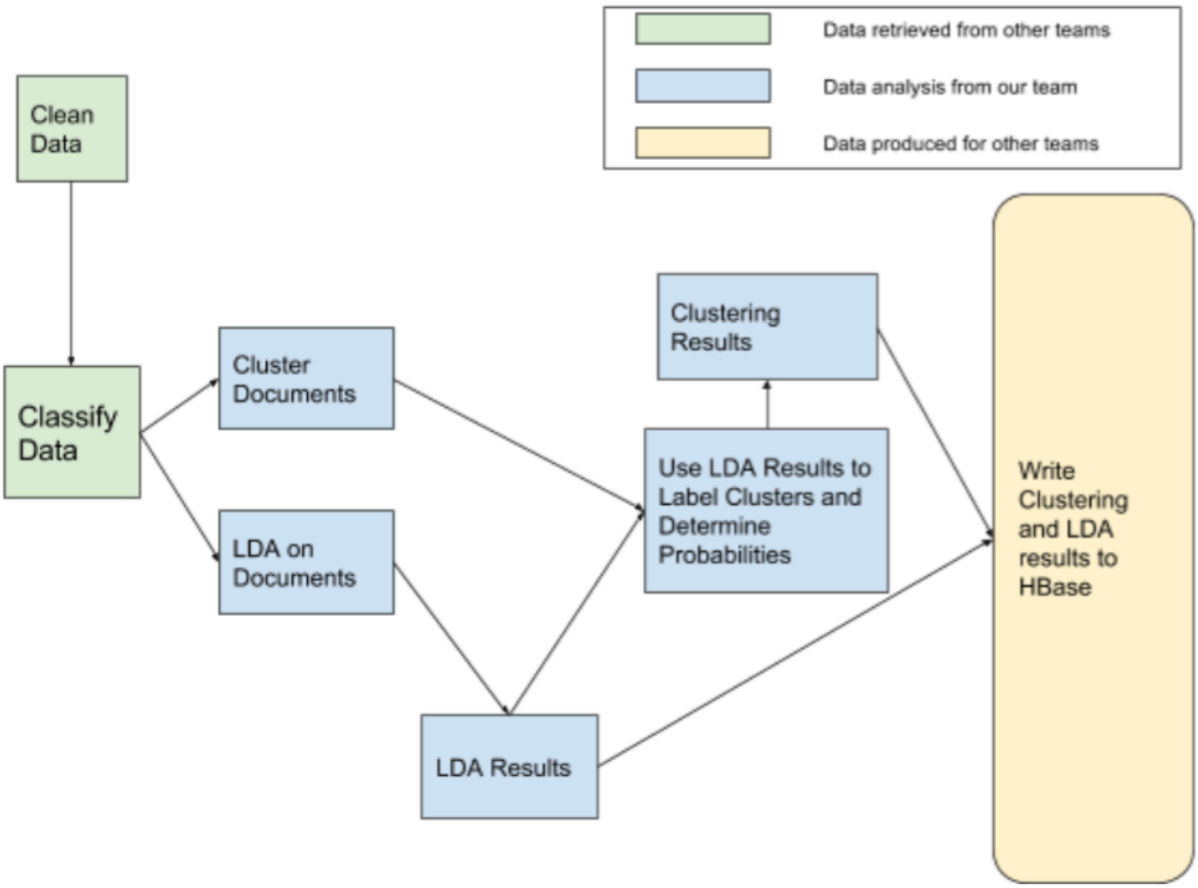

Figure 3. Workflow Diagram

A. Cleaning

A necessary step in both topic analysis and clustering was cleaning the raw tweets and webpages that we received. This is important so that we only deal with the relevant terms and metadata within each document. The Collection Management teams had planned to provide us with cleaned data— removing profanity and stopwords and lemmatizing the text. However, we were ready to begin running experiments on our clustering and topic analysis tools before the cleaned tweets were available on HBase, so we applied cleaning scripts from the CLA team to clean the data.

The CLA team classified tweets into real world events and assigned each tweet a classification tag. We pulled tweets based on their classification, and then clustered them and extracted topics on each class of tweets.

B. Clustering

The requirements for clustering were divided into feature extraction, the chosen clustering algorithm, and the cluster labeling. Careful selection for each of these allowed us to choose the method to cluster our data.

In order to cluster similar documents together, we first needed to determine the attributes that we will use from each document. The considered features were the document's metadata, as well as the text contained within the document. Drawing on the results of the previous semester, we used Word2Vec [6] to transform the document text into a feature vector. The feature vectors represented the documents in a way that could be used by each clustering algorithm.

We needed to choose the clustering algorithm that works best with our dataset. For this purpose we considered two different algorithms: K-means and DBSCAN. The K-means algorithm clustered into K clusters which were tuned by graphing the sum of squared errors within each cluster. The DBSCAN algorithm clustered based on the density of the documents and did not require a predetermined cluster count. The goal was that through cluster evaluation metrics, we would determine which algorithm was better for our dataset.

After the documents have been clustered we needed to label them both so that a human user could intuitively understand what the cluster is about, and so specific clusters can be used to satisfy queries. By being able to label our clusters well, we were able to give the SOLR team the information needed to use our cluster labels as facets for the user to discover the documents having that relationship. The cluster label was created as a collection of words that describe the documents within that cluster.

C. Topic Analysis

The requirements for topic analysis were to give topics for sets of documents and also to find how much certain documents relate to those topics. We wanted to answer questions such as "how much is this document about this topic?" and "what documents talk the most about this topic?". By running LDA as our topic analysis algorithm, the Front-End and SOLR teams were able to use the topic models to answer these questions for the users.

A number of parameters needed be tuned so that the topics can be the best representations of our data. These parameters included the number of iterations, the terms to represent

each topic, and the number of topics to select. Choosing the right number of iterations was a balance between the runtime of the algorithm and the accuracy requirements of our topics. Selecting the number of terms will let us balance the expressivity of our topics with how specific and relevant they can be to the dataset. By tweaking the number of topics we choose, we can make sure we specify the different topics that represent each document set while not including topics that are redundant.

There are two approaches to labeling the topic models produced by our topic analysis: the manual approach and the automated approach. Because each event had so many tweets (40,000 - 600,000 tweets), it was not productive to manually label each topic. The CS 5604 Topic Analysis team from Spring 2016 [8] was able to automate the process of topic labeling by selecting the most probable word from each topic.

D. External Expectations

To meet the requirements for the SOLR team, we needed to produce the top topics that we believed a document was about. In addition, we needed to give each document a cluster that it belongs to and a name that was descriptive to humans. These clusters could be used both for speeding up indexing and also as a facet for the Front-End team. Our external expectations are illustrated in Table 2 below.

Column Family	Column	Description	Example	Stored	Indexed	Facet
doc-topic	label-list	1. labels generated by LDA model 2. extract the top two labels from each topic	Signed,students; event,excited; today,register; april,thanks; community,little	Yes	label_list_s	No
	doc-probability	each value presents the probability of the tweet belongs to a certain topic	0.29112; 0.01820; 0.12435; 0.02572; 0.54058	No	probability_list_s	No
doc-cluster	cluster-label	label of the cluster to which the doc belongs	NAACP,stories	Yes	cluster_label_s	No
	cluster-probability	the probability of the doc in the cluster	0.55167	No	doc_probabaility_f	No

Table 2. Schema for Expected Output

E. Internal Project Exploration

After reading last semester's "Clustering and Social Networks for IDEAL", we saw an opportunity to merge our efforts on clustering and topic labeling [7]. We saw an opportunity to evaluate our clustering by comparing it to our topic analysis results. We also saw an opportunity to automate our cluster labeling by using the top terms from each topic that related to the cluster.

IV. Design

Our primary concern was to achieve all external project expectations, since the main priority is to build a functional IR system for event archiving. To do this, we decided to focus on building a clustering tool that satisfactorily clusters the tweets, and a topic analysis tool that satisfactorily models the topics in the tweets. By building these, we had the tools that would be required to provide the SOLR team with table entries that identify the probabilities for each document, their topic, and the clusters that they belong to (see Table 2).

Once those tools were finalized, we focused our efforts on using topic analysis to evaluate our clustering. By being able to map topics back to their documents and knowing which documents belonged to which clusters, we were able to see if the documents in each cluster are actually similar based on their topics. Similarly, we automated the process for cluster labeling. Once we were satisfied with the performance of our clustering tool, we used the topics produced by our topic analysis tool to label the clusters and generate cluster probabilities. In this report, we refer to the probability of a tweet belonging to its assigned cluster as the cluster probability of that tweet. The algorithm used to label clusters and generate probabilities is detailed in the developer's manual section. Since we knew the topics for all the documents in each cluster, we labeled the cluster as a set of topics found within the cluster.

We also performed clustering and cluster labeling manually to see if the clusters and labels generated by our tools were similar to those clustered and labeled by a human. We manually clustered collection 42 - the Connecticut school shooting. It had 32,768 tweets. The tweets were manually clustered based on keywords, discussion patterns, and sentiments. Each tweet was only assigned to one category. After clustering the tweets in the Connecticut school shooting collection, we labeled them based on the sentiment in the tweet, keywords, as well as the content of the entire tweet. Tweets were clustered based on if they were related to a news agency (shooting related news), public reaction to the shooting (public reaction), condolence, etc. Table 3 below shows each manual cluster, the number of occurrences, and the percentage of the collection that it made up. Figure 4 shows a visual representation of the clusters in the collection. These manual labels were compared to the results of the K-means clustering tool. We then realized that the manual clustering might differ from the results of the tool because the human will consider the sentiments and source of the tweet, while the tool only looks at the feature terms.

Cluster Name	# occurrences	Collections (%)	Example of Tweet
Shooting related news	7641	23.32	After the shooting in Connecticut I appreciate school more. #begrateful
Public reaction	7120	21.73	_NoShaneNoGain: This girl is making me so mad talking about how Connecticut school shooting wasn't real and how this isn't real and ...
Post Shooting [News/ Reaction]	6556	20.01	At least one dead in shooting at Connecticut school
Gun laws discussion	4615	14.08	Newtown residents unite on gun control: Several Newtown Connecticut residents decided to drop the... http://t.co/mwSRYjez
N/A	1812	5.53	http://t.co/u2uth5pshT
Funeral	1090	3.33	Connecticut school shooting: Funerals begin for victims http://t.co/Fckb0qxU
President's reaction [Discussion]	916	2.80	Connecticut school shooting: Barack Obama declares 'we can't tolerate this any more': America is failing to meet... http://t.co/r5mhifzt
Police Related	856	2.61	'People's hearts are broken' state police official says http://t.co/Z1PFaVtd
Legal Matters	673	2.05	Man pleads guilty to threatening Newtown residents after Sandy Hook School shooting; requesting limited sentence: http://t.co/sWTIG5OTCq
Prayer	590	1.80	@whitehouse Westboro Church is going to picket funerals in Newtown.\x0Ahttp://t.co/Sarbao0Y\x0ANewtown needs help to protect the families
Security	211	0.64	Local schools vamp up security http://t.co/A5yX34QZ #tbtoday
Porn words	206	0.63	A time of peace & last goodbyes ruined by some sick ***. Wish I could be there to defend with the rest of CT.\x0Ahttp://t.co/nWSGbisa
Movie	195	0.60	Connecticut School Shooting: Syfy Pulls Violent Haven Episode in Wake of Newtown Tragedy http://t.co/fHRA7tjh
Hope/ Awareness	114	0.35	My thoughts on the school shootings in Connecticut:\x0Ahttp://t.co/qhnsLABa\x0A#hope #connecticut #newtown
Live Video/ Update	59	0.18	Live video: Connecticut Gov. Dannel Malloy holds a news conference - @NBCNews http://t.co/aABAGsR0
Upset	40	0.12	It upsets me even more that this race was in honor of the Connecticut school shooting .
Children Reaction	37	0.11	'Hug them;' 'cry with them': The children. Initial reports of a shooting inside a Newtown Connecticut elementa... http://t.co/9baCOdLI
Political	37	0.11	LISTEN: The @PoliticalJones Show talking with @DrKamalaUzzell about the Connecticut School Shooting & Mental Health

Table 3. Manual Clustering Results

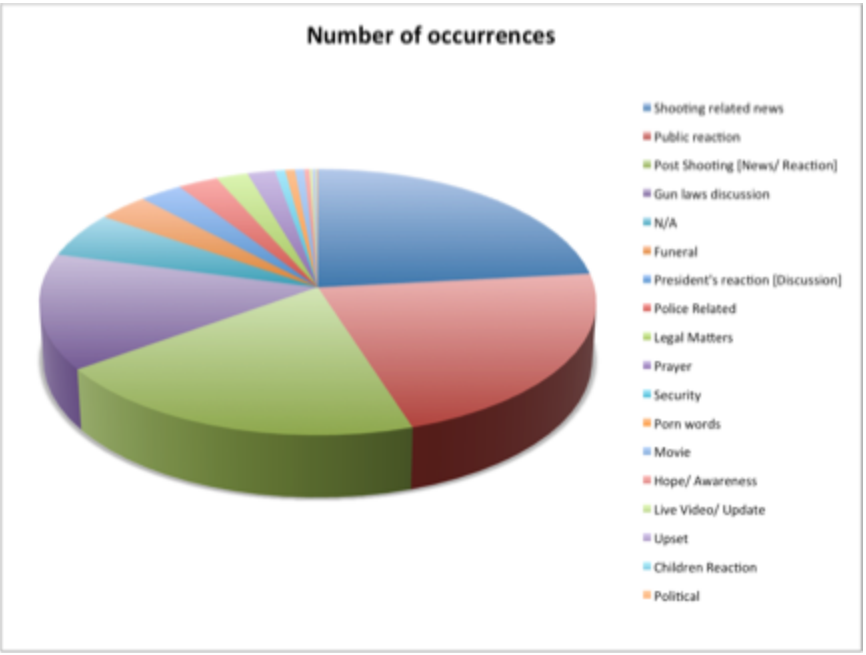

Figure 4. Manual Clustering Distribution

We pulled sets of tweets based on their classification tags and ran our clustering and topic analysis tools on each set. Then, we wrote those results to HBase, formatted according to the schema in Table 2.

V. Implementation

A. Selecting a Clustering Algorithm

There was an initial interest in considering DBSCAN for our clustering algorithm. Since DBSCAN allows clusters to take any shape and does not require a pre-specification for number of clusters, we felt confident that DBSCAN would provide a more accurate clustering. Since DBSCAN was not offered as one of the built-in unsupervised Machine Learning algorithms on Spark, we sought out a third party implementation of DBSCAN. We looked at Irving Cordova's implementation of DBSCAN [11] and Mostofa Patwary's implementation of DBSCAN [12]. Unfortunately, both implementations required a newer Spark version than Spark 1.5.0, which is currently on the cluster. We decided that it would be a better use of our time to continue looking for more DBSCAN implementations, rather than implementing our own. pyParDis DBSCAN uses sklearn to implements DBSCAN in pyspark (a python wrapper for the scala spark libraries) [13].

pyParDis was able to run on our cluster, however we ran into limitations by pyspark that did kept us from proceeding. This implementation only works with small datasets that have low feature vectors. The pyspark wrappers did not allow us to distribute our Word2Vec model onto the cluster, as it could not access the SparkContext objects within them. This means that almost all of the processing would have to be done locally. When we tested this on the small datasets that were given to us, it did not complete until we decreased the word vector size that Word2Vec would produce to very low values. This meant that we would lose a lot of data as we attempted to cluster the data with very small word vectors.

Upgrading the cluster's Spark version or rewriting the pyParDis DBSCAN code could yield a better DBSCAN performance, but we ultimately decided to continue our clustering efforts by using K-means clustering.

In the implementation of K-means clustering tool, we used Word2Vec for feature extraction and ran our feature vectors through MLlib's K-means clustering algorithm [10]. Once we could successfully use our tool to cluster datasets, we needed to decide on the parameters like number of clusters and number of iterations. To fine tune them for the most efficient results, we decided to use the cluster probabilities of tweets as attributes to determine the efficiency of the tool, since tweets belonging to their assigned cluster with maximum probability basically marks the efficiency of the algorithm. We used our clustering tool with different values of K (K=4,5,6) and generated cluster probabilities for each collection. Since cluster probabilities indicate the possibility of tweets belonging to their assigned clusters, we decided to measure the efficiency and accuracy of the

clustering algorithm by analyzing their probabilities. The detailed algorithm for this method is given in the developer's manual. After analyzing the results, we picked the K value which resulted in highest mean cluster probability. Figure 5 below shows the distribution of tweets over 6 clusters for the 'Kentucky Accidental Child Shooting' collection as clustered by our tool.

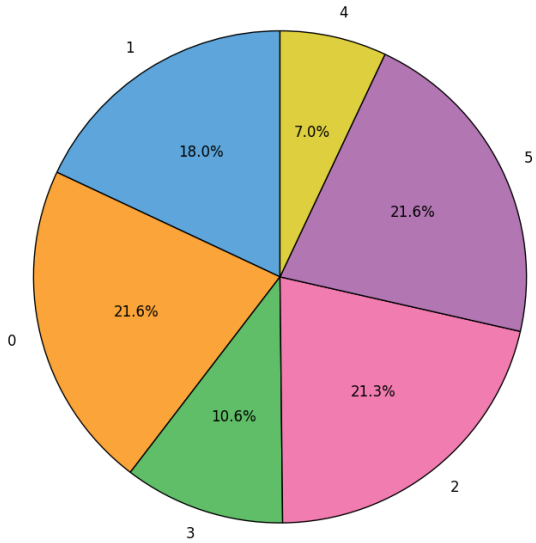

Figure 5. Tweets Distribution Over Clusters

For the same collection, the cluster probabilities of tweets when run for 6 clusters (k=6), were distributed as shown in Figure 6.

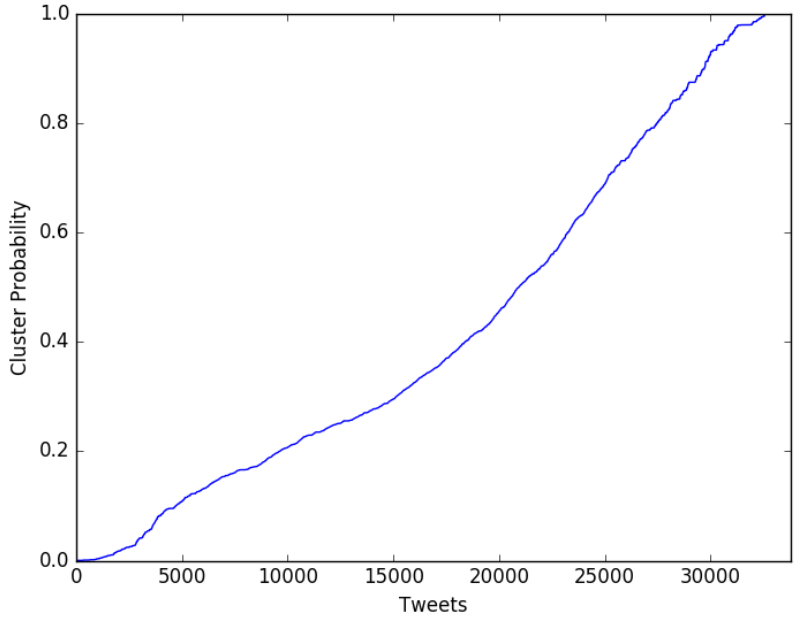

Figure 6. Cluster Probabilities

A detailed summary of the real world event collections that we used our tool on and the number of clusters (K values) which produced the most efficient results for each of them is given in Table 5.

B. Topic Analysis

Using last semester's Latent Dirichlet Allocation script, we were able to generate topics and map topic probabilities back to their respective documents. We added stop words based on the name of the real world event. For example, if we were performing topic analysis on the New York Firefighter Shooting, we already know that the documents will likely be about New York, or firefighters, or shootings. Thus, having a firefighter topic or a shooting topic would not be very meaningful, since it would not give us information that we did not already know. So, we added "firefighter" and "shooting" to the stop words during that particular run of LDA.

We then tokenize the documents to find the feature words of the document, filtering out the stop words that we added. Then, we convert the documents into term count vectors and set LDA parameters— number of topics and number of iterations. For our experiments, we set our number of iterations to 100. It was recommended by the previous semester's topic analysis team. We tried to increase the number of iterations, but it caused an OutOfMemory error, so we found it best to keep the iterations at 100. We ran our experiments by first testing 5 topics and decreasing the number of iterations depending on if we thought there was too much breadth and overlap in the topics, or increasing the number of topics if we thought the topics were too condensed and that there were more topics that could be used for discussing the real world event.

For the New York Firefighter Shooting, we wanted to see what the probability distributions were for each document. That is to say, we plotted a graph, sorting by document ID, where the y-axis was greatest probability that a topic corresponded to a particular document (the document's greatest topic probability). We found that some of the probabilities were as high as 0.99, which is promising because that meant that some of documents were very closely related to the topic. We also saw some of the probabilities were as low as being close to 0.13. We were initially concerned by the low probability; however, it is important to note that the New York Firefighter Shooting set of tweets has 8 topics. So if a document were equally probable across all 8 topics, each topic would have a 0.125 probability. So the low probability is not as alarming as we had suspected, because that shows that the document closely encompasses all of the 8 topics. The graph can be seen in Figure 7 below.

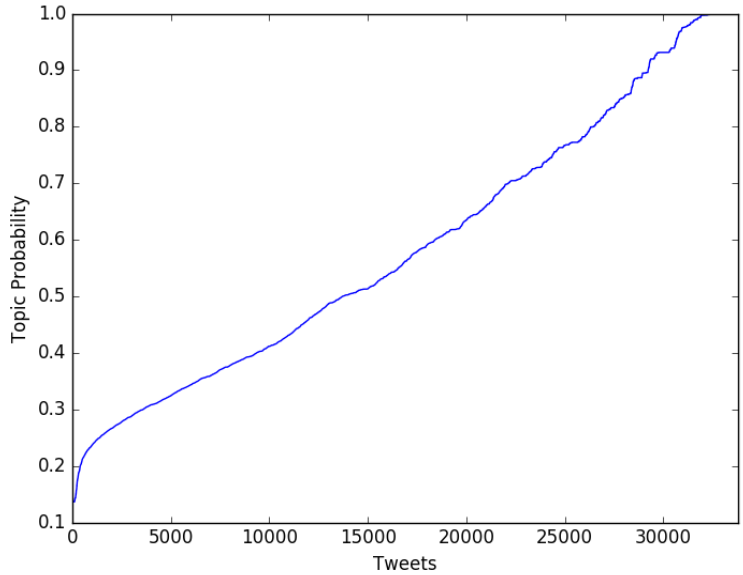

Figure 7. Topic Probabilities for New York Firefighter Shooting

We were also interested in seeing the distribution of the topics across the New York Firefighter Shooting documents. Based on Figure 8 below, we can see that topics were relatively evenly distributed.

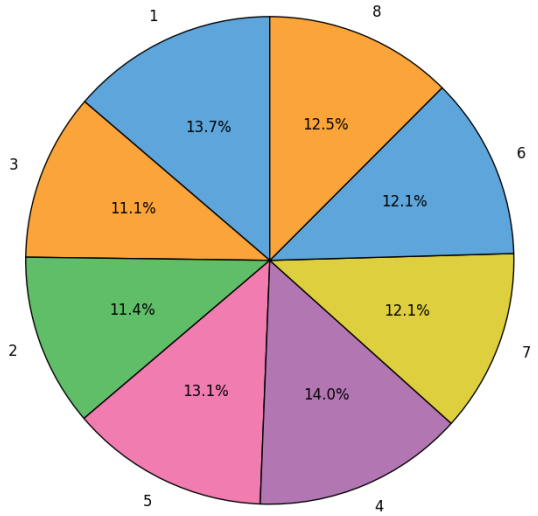

Figure 8. Distribution of Topics for New York Firefighter Shooting

We ran LDA on 9 real world events. Results can be found in Table 4 below. If some topics seem less significant (still, three, etc.), the word can be added to a list of stop words. However, these results should be discussed because "three" might be a significant number pertaining to the New York Firefighter shooting. Or, "still" might be a significant topic because it reflects that the people in the area were continuing to feel the effects of Hurricane Arthur.

Real World Event	Number of Topics	Number of Tweets	Topics
NewYorkFirefighterShooting	8	33788	Video, Shoot, Firefighter, Shooting, Three, Injure, Police Death
KentuckyAccidentalChildShooting	8	62959	Field, Connecticut, Police, Throw, Wisconsin, Shoot, Sister
NewtownSchoolShooting	8	481648	School, Kentucky, Harlem, Victim, Newtown, Obama, Report, Elementary
ManhattanBuildingExplosion	4	656103	Harlem, Centralpark, Building, Brooklyn
ChinaFactoryExplosion	8	17479	People, Sandy, Computer, Media, Black, Kentucky, Police, Hurricane
TexasFertilizerExplosion	10	72202	Federal, Cause, Firefighter, Report, Boston, Video, Explode, Obama, First, Blast
HurricaneSandy	8	33246	Manhattan, Amazing, Newyork, Isaac, Skyline, Speak, Brooklyn, Latest
HurricaneArthur	8	280937	Sandy, Power, Merlin, Texas, Still, Minha, North, Missingmerlin
HurricaneIsaac	8	90583	Sandy, School, Please, Power, Storm, History, Since, Victim

Table 4. Topic Analysis Results

C. Cluster Labeling

To automate the cluster labeling, we wrote a Python script to use clustering results along with topic analysis results and generate cluster probabilities of tweets along with labeling the clusters in each collection. We generated a mean topic frequency matrix to determine the two most frequent topics in each cluster and labeled the clusters by combining the labels of these topics. We also calculated the cluster probabilities of tweets based on the topic analysis results. A detailed explanation on how this was done can be found in the developer's manual section.

A summary of all the collections we clustered and the cluster labels in each collection and produced by our tool is given in Table 5.

Real World Event	Number of Clusters	Cluster Labels
NewYorkFirefighterShooting	6	Cluster 0: “Video,shooting,police,people” Cluster 1: “shooting,firefighter,three,shoot” Cluster 2: “video,shooting,firefighter” Cluster 3: “firefighter,ambush,death,shooting” Cluster 4: “shoot,kentucky,three” Cluster 5: “three,shoot,kentucky”
KentuckyAccidentalChildShooting	6	Cluster 0: “Police,trooper,connecticut,people” Cluster 1: “connecticut,people,throw,shoot” Cluster 2: “wisconsin,shoot,throw” Cluster 3: “sister,shoot,guard” Cluster 4: “shoot,guard,sister” Cluster 5: “field,percent,wisconsin,shoot”
NewtownSchoolShooting	6	Cluster 0: “Report,school,newtown” Cluster 1: “harlem,school,report” Cluster 2: “school,newtown,victim” Cluster 3: “obama,school,newtown” Cluster 4: “kentucky,police,harlem,school” Cluster 5: “victim,school,newtown”
ManhattanBuildingExplosion	6	Cluster 0: “centralpark,harlem,brooklyn,queens“ Cluster 1: “harlem,newyork,building“ Cluster 2: “building,harlem,centralpark” Cluster 3: “newyork,centralpark“ Cluster 4: “harlem,newyork,brooklyn,queens” Cluster 5: “building,harlem,newyork”
ChinaFactoryExplosion	5	Cluster 0: “media,think,hurricane,sandy” Cluster 1: “kentucky,state,computer” Cluster 2: “black,white,computer,kentucky” Cluster 3: “people,romney,police,rifle” Cluster 4: “sandy,hurricane”
TexasFertilizerExplosion	6	Cluster 0: “first,responder,report,massive” Cluster 1: “report,massive,firefighter,investigation” Cluster 2: “boston,bombing,firefighter,investigation” Cluster 3: “federal,still,firefighter,investigation” Cluster 4: “explode,firefighter,report,massive” Cluster 5: “cause,criminal,blast,facility”
HurricaneSandy	6	Cluster 0: “skyline,manhattan,latest” Cluster 1: “newyork,manhattan,amazing” Cluster 2: “speak,manhattan,latest” Cluster 3: “skyline,manhattan,amazing” Cluster 4: “isaac,manhattan,speak” Cluster 5: “manhattan,harlem,speak”

HurricaneArthur	6	Cluster 0: “minha,lindo,still,storm” Cluster 1: “sandy,death,texas” Cluster 2: “missingmerlin,merlin” Cluster 3: ”north,storm,still” Cluster 4: ”texas,sandy,power,canada” Cluster 5: ”missingmerlin,merlin,texas,sandy”
HurricaneIsaac	6	Cluster 0: “storm,right,power,sandy” Cluster 1: ”school,sandy,history,deadliest” Cluster 2: ”victim,sandy,history,deadliest” Cluster 3: ”power,sandy,school” Cluster 4: ”storm,right,history,deadliest” Cluster 5: ”since,sandy,school”

Table 5. Clustering Results Summary

We also manually clustered the documents of one collection, labelled the clusters and compared the results to the results obtained from our tool. We noticed that manual clustering resulted in labels being based more on the labeler’s perception of the sentiments to keywords in the tweets, while the results from our tools were mostly based on keywords themselves.

VI. Timetable

Survey Literature	September 11, 2016
Interim Report 1	September 20, 2016
Clustering Performs Satisfactorily on Small Datasets	October 3, 2016
Determine which Clustering Algorithm will be Used	October 10, 2016
Interim Report 2	October 11, 2016
Topic Analysis Performs Satisfactorily on Small Datasets	October 14, 2016
Interim Report 3	November 3, 2016
Evaluate Cluster Labeling Using Topic Analysis	November 3, 2016
Evaluation - Finetune Clustering and Topic Analysis Parameters	November 14, 2016
Assess Cluster Labeling Automation via Topic Analysis	November 14, 2016
HBase Integration Complete	November 23, 2016
Final Project Presentation	December 1, 2016
Final Project Report	December 7, 2016

Table 6. Timetable

VII. User Manual

This section serves as the guide for readers interested in executing our code and obtaining clustering and topic analysis results for a collection of tweets. The users need to have Spark installed in order to be able to run our code. The details regarding installation and documentation of Spark and Scala are provided in the beginning of the developer's manual section. The complete workflow of the project is shown in Figure 3.

A. K-means

To run our K-means clustering algorithm using the Spark framework, the user needs to have all the tweets saved in a CSV file in the format given in Figure 9. The tweets need to be pulled from HBase as described in section D (Pulling tweets from HBase). Each tweet should have a unique identifier. The user needs to provide the CSV file with tweets as input to our K-means clustering tool.

```
77-412623177278169088, push regulation force clinic close shrug safety regulation httâ
77-413271957182771200, texas regulatory response fertilizer factory level town nothing
77-430743598443675648, one will get hurt company regulation
77-456493772981800960, community ask ammonium nitrate store west fertilizer near school residence rememberwest
77-456717511736586241, explosive news west seattle gas lac megantic
77-458627975714725888, now need time machine fertilizer plant blast kill preventable safety board say
77-458662356479148033, westtx fertilizer plant blast kill preventable safety board say
77-458662490491334656, westtx fertilizer plant blast kill preventable safety board say
77-458749693234008064, west refinery disaster result politician war safety regulation dead injured well
77-458969871057817600, welcome fire emergency plan fertilizer plant blast kill preventablesafety board say
77-486346756238491648, fire code legislature meet every year fertilizer wooden building great nyt west httâ
```

Figure 9. Sample Clustering Input

To run the K-means clustering tool, the user needs to go to the project directory (Clustering/ClusteringExperiments/kmeans/) containing the sbt file and src directory with scripts, and execute the following:

```
% sbt package
% spark-submit <jar file path> <input .csv file path>
```

The sbt file needs to have the following dependencies:

```
name := "Kmeans"
version := "1.0"
scalaVersion := "2.10.4"

libraryDependencies ++= Seq("org.apache.spark" %% "spark-core" %
"1.5.0","org.apache.spark" %% "spark-mllib" % "1.5.0")
```

Table 7. Sbt Configuration for Clustering

Our main algorithm can be found in src/main/scala/ClusteringWithWord2Vec.scala inside the project directory.

The K-means clustering tool will write the clustering results to an output file as shown in Figure 10. In the figure, 42 is the tweet collection, 305948159341887488 is the tweet ID, and that tweet belongs to cluster 3.

```
(42-305948159341887488,3)
(42-280116280084791296,3)
(42-285018175920300032,1)
(42-280866110369325056,0)
(42-544159433673420800,1)
(42-279786938565804034,3)
```

Figure 10. Sample Clustering Output

While clustering, the user can modify the algorithm code to change parameters like the value of K (number of clusters), number of iterations, etc. to yield the most efficient results. The results of K-means clustering can be used in calculating cluster probabilities of tweets and automating cluster labeling as explained in Section C. In this project, we measured the efficiency of the algorithm by analyzing the cluster probabilities of tweets for different values K. This is explained in better detail in Section C.

B. Topic Analysis

To run our topic analysis code, the user should go to the directory that contains our Latent Dirchlet Allocation tool (lda.scala). The user should open spark-shell and run the tool with the following commands:

```
% sbt package
% spark-submit <jar file> <input file> <topic words file>
        <intermediate document topic file>
        <final document topic file> <number of topics>
         <number of iterations>
```

The following dependencies should be in the config.sbt file.

```
name := "LDA-Web22"

version := "1.0"

scalaVersion := "2.10.4"

libraryDependencies ++= Seq(
"org.apache.spark" %% "spark-core" % "1.0.0",
"org.apache.spark" %% "spark-mllib" % "1.5.0"
)
```

Table 8. Sbt Configuration for Topic Analysis

Note that the jar file can usually be found in the target directory when the project is packaged with the sbt command (sbt package). The input file is the file containing the document identifier and the probabilities that each topic corresponds to it. The program writes the document topic probabilities to an intermediate file, and then maps those document identifiers to the document topic probabilities.

Sample output for the tool is shown below. The topic labels are “connecticut”, “school”, “shooting”, “newtown”, and “obama”. Topic 1, “connecticut”, consists of the keywords: “connecticut”, “school”, “shooting”, “report”, “victims”, “families”, “church”, “children”, “westboro”, and “gunman”. There is a 10.55% chance that “connecticut” belongs to Topic 1, 10.10% chance that “school” belongs to Topic 1, and so on.

```
connecticut    connecticut,school,shooting,report,victims,families,church,children,westboro,gunman
0.1055376639219627,0.10101739802034791,0.09948557050990624,0.022659203527663124,0.015694495103379683,0.015
108312777007065,0.014876276226704855,0.013687039983885894,0.013547147570134593,0.01288129659804989
school  school,newtown,connecticut,shooting,elementary,sandy,lanza,students,shooter,residents
0.11953461058313528,0.07758897665471845,0.06707797079616572,0.05908203251279089,0.04868613451168881,0.0454
8692711742721,0.029299343754394233,0.020026920922790306,0.019945942214100298,0.011082036040643345
shooting       school,connecticut,shooting,victims,massacre,newtown,calls,funerals,shootings,armed
0.1331974474841733,0.102318119697827,0.06271527305306174,0.04995613154938051,0.028995378042185348,0.022578
72781117121,0.018804310186679447,0.015418686550106092,0.013102236347400691,0.011037676262221957
newtown school,connecticut,shooting,newtown,sandy,video,victims,national,first,elementary
0.11603979260551882,0.0851659548082822,0.06185517141759922,0.04854284028272705,0.02610626513875809,0.01486
1039390980432,0.013217809132434692,0.013043124449878946,0.012203536429226206,0.010843227125509612
obama   school,shooting,connecticut,obama,president,newtown,control,today,elementary,barack
0.10418467005718914,0.09275769274964034,0.08629696621612143,0.04056751586433503,0.01614930471745316,0.0124
14990347788257,0.011890382878858609,0.01082196296705905,0.009973336056566062,0.009882517782741568
```

Figure 11. Sample LDA Output

If we open the files in the specified directory in the first parameter, we will find rows of data that resemble Figure 10. The first term is the collection ID (42), then the tweet ID (e.g. 287646893926932482), separated by a hyphen. The next five decimals are the probabilities that the tweet pertains to each topic. There is a 13.7% chance that tweet 287646893926932482 pertains to Topic 1 (connecticut), 43.6% chance that the tweet pertains to Topic 2 (school), and so forth. Note that this directory will be written to the Hadoop file system.

```
(42-287646893926932482,,[0.13707776636822683,0.43585409857682506,0.1584351010747109,0.11976254032020717,0.1488704936600301])
(42-281038116687605760,,[0.1716649322143387,0.15415449222281777,0.20482323027208987,0.2861751437060074,0.18318220158474624])
(42-280781250262888449,,[0.12372359116813096,0.4655054896270968,0.1665123440867802,0.1167743283208475,0.12748424679714 46])
(42-281513391363870720,,[0.104270049966495,0.13317679089789436,0.331476804104532,0.27229783376423344,0.15877852126684522])
(42-279886345856483328,,[0.06422664162099553,0.6394721377344293,0.07987104574868312,0.1273046454589442,0.08912552943694782])
(42-554919539549683712,,[0.20750846952869348,0.08292628769937178,0.18298083877540536,0.2941253175339311,0.2324590864625982])
(42-299703118826987520,,[0.03352639176455314,0.3908174387819337,0.10812104832242196,0.4267161318225273,0.04081898930856394 5])
(42-282187185925525505,,[0.158211203170114,0.3901465047525664,0.04854906789011 3345,0.07916842386128094,0.3239248003259254])
(42-289540362551304194,,[0.18698104069999263,0.17582291562862476,0.26179368168886963,0.18971615486397608,0.18568620711853692])
(42-279856499537375232,,[0.12988945344051267,0.3963498235691175,0.1630054834324129,0.16357806120138166,0.14717717835657532])
```

Figure 12. Sample LDA Document Output

C. Cluster Labeling

If the results of K-means clustering and topic analysis are available, the user can use them to generate cluster probabilities of tweets and perform automated cluster labeling. The detailed methodology and algorithm we used to generate cluster probabilities and label clusters using the results of clustering and topic analysis are presented in the developer’s

manual. We have written a Python script (Clustering/CL_8T.py) to perform the said tasks. The user needs to provide 3 files as inputs to be able to execute the script.

1. Clustering results output text file in the format as given in Figure 10.
2. Topic analysis results output text file with document IDs and their topic probabilities as given in Figure 12.
3. Topic labels text file with topics, their labels, and their probabilities as given in Figure 11.

The user needs to change the names of the input files appropriately before executing our Python script. The Future Work section of this report suggests refactoring the code to accept a configuration file that sets these names so that the code does not have to be altered. The script performs the task and produces two output files. One has the generated cluster labels, as given in Figure 13. The other has tweets, their assigned clusters, and their calculated cluster probabilities, as given in Figure 14.

```
Cluster Labeling :
Cluster 0 : shoot,kentucky,three
Cluster 1 : video,shooting,police,people
Cluster 2 : firefighter,ambush,death,shooting
Cluster 3 : video,shooting,firefighter
Cluster 4 : shooting,firefighter,three,shoot
```

Figure 13. Cluster Labeling

```
['41-290234068744937472', 'obama,school,newtown', 0.8970219661704786]
['41-281566052788338689', 'school,newtown,victim', 0.28684395479427005]
['41-676669896533655552', 'report,school,newtown', 0.24879228743426823]
['42-281236677513596930', 'obama,school,newtown', 0.09330286085602313]
['42-279966385352867841', 'victim,school,newtown', 0.512085158937202]
['42-280178811692277760', 'report,school,newtown', 0.308378807483225]
['23-287056634843394048', 'obama,school,newtown', 0.3715146807905034]
['41-280740518579810304', 'school,newtown,victim', 0.265374852071531]
['41-282054392813862912', 'obama,school,newtown', 0.9092899863557071]
['42-279994885015212032', 'victim,school,newtown', 0.512085158937202]
['41-280528288051523584', 'victim,school,newtown', 0.4870410407398223]
```

Figure 14. Cluster Probabilities

D. Pulling Tweets from HBase

Pulling documents from HBase requires HBase.scala, HBaseInteractions.scala, DataRetriever2.scala, and CleanTweet.scala. Modification to DataRetriever2.scala will have to be made. The String variable, _class, represents the real world event that we need to pull tweets from. This variable should be updated so that it is set to the real world event as written in HBase (e.g., NewYorkFireFighterShooting, NewtownSchoolShooting, etc.).

```
% sbt package
% spark-submit --master local --jars
        jars/stanford-corenlp-3.4.1.jar,jars/stanford-corenlp-3.4.1-models.jar
         --master local <jar file> "read" <file to write results to> 4.0
```

Table 9 shows the dependencies in the sbt config file.

```
name := "CTA_project"

version := "1.0"

scalaVersion := "2.10.4"

libraryDependencies ++= Seq("org.apache.spark" %% "spark-core" % "1.5.0","org.apache.spark" %%
"spark-mllib" % "1.5.0")
resolvers ++= Seq(
  "Cloudera repos" at "https://repository.cloudera.com/artifactory/cloudera-repos",
  "Cloudera releases" at "https://repository.cloudera.com/artifactory/libs-release"
)
libraryDependencies += "eu.unicredit" %% "hbase-rdd" % "0.7.1"
libraryDependencies += "edu.stanford.nlp" % "stanford-corenlp" % "3.4.1" artifacts (Artifact("stanford-
corenlp", "models"), Artifact("stanford-corenlp"))
libraryDependencies ++= Seq(
  "org.apache.hbase" % "hbase" % "1.2.3",
  "org.apache.hbase" % "hbase-common" % "1.0.0-cdh5.5.1" % "provided",
  "org.apache.hbase" % "hbase-client" % "1.0.0-cdh5.5.1" % "provided",
  "org.apache.hbase" % "hbase-server" % "1.0.0-cdh5.5.1" % "provided",
  "org.json4s" %% "json4s-jackson" % "3.2.11" % "provided"
)
```

Table 9. Sbt Configuration for Pulling Tweets from HBase

When the following commands are entered, the tweets will be written to the specified file in HDFS. The output file shows the document identifier, then the cleaned text, separated by a comma. A sample of the output can be seen in Table 10.

```
173-581204417925394432, dramatic photo show escape moment build engulf flames
173-581204467586043904, fog smoke blanket downtown nyc view world trade center photo
173-581204469171380224, dramatic photo show escape moment build engulf flames
173-581204562234777600, dramatic photo show escape moment build engulf flames
173-581204882616647680, smoke overwhelming fdny fight blaze east village photo via
173-581205175605411840, fog smoke blanket downtown nyc view world trade center photo
173-581205305134039041, dramatic photo show escape moment build engulf flames
```

Table 10. Sample Output for Pulling Tweets from HBase

E. Writing Results to HBase

Writing results to HBase requires HBase.scala, HBaseInteractions.scala, DataWriter.scala, DataWriter2.scala, and WriteCluster.scala.

The following commands should be run to write the topic analysis results to HBase.

```
% sbt package
```

% spark-submit --master local <jar file> “write” <results file>
<real world event number>

Note that the real world event number is the decimal that corresponds to a real world event. The table below shows the mapping for which decimal (real number) corresponds to which real world event.

Decimal	**Real World Event**
0.0	NewYorkFirefighterShooting
1.0	ChinaFactoryExplosion
2.0	KentuckyAccidentalChildShooting
3.0	ManhattanBuildingExplosion
4.0	NewtownSchoolShooting
5.0	HurricaneSandy
6.0	HurricaneArthur
7.0	HurricaneIsaac
8.0	TexasFertilizerExplosion

Table 11. Real World Event Decimal Mapping

Table 12 shows the dependencies in the sbt config file.

```
name := "CTA_project"

version := "1.0"

scalaVersion := "2.10.4"

libraryDependencies ++= Seq("org.apache.spark" %% "spark-core" % "1.5.0","org.apache.spark" %%
"spark-mllib" % "1.5.0")
resolvers ++= Seq(
  "Cloudera repos" at "https://repository.cloudera.com/artifactory/cloudera-repos",
  "Cloudera releases" at "https://repository.cloudera.com/artifactory/libs-release"
)
libraryDependencies += "eu.unicredit" %% "hbase-rdd" % "0.7.1"
libraryDependencies += "edu.stanford.nlp" % "stanford-corenlp" % "3.4.1" artifacts (Artifact("stanford-
corenlp", "models"), Artifact("stanford-corenlp"))
libraryDependencies ++= Seq(
  "org.apache.hbase" % "hbase" % "1.2.3",
  "org.apache.hbase" % "hbase-common" % "1.0.0-cdh5.5.1" % "provided",
  "org.apache.hbase" % "hbase-client" % "1.0.0-cdh5.5.1" % "provided",
  "org.apache.hbase" % "hbase-server" % "1.0.0-cdh5.5.1" % "provided",
```

```
  "org.json4s" %% "json4s-jackson" % "3.2.11" % "provided"
)
```

Table 12. Sbt Configuration for Writing Results to HBase

The following commands should be run to write the clustering results to HBase.

```
% sbt package
% spark-submit --master local <jar file> “write” <results file> 0.0
```

Note that these commands should be run in the directory that contains DataWriter2.scala and WriteCluster.scala, and these two files should be in a separate directory from DataWriter.scala and HBase.scala. This directory should contain HBaseInteraction.scala and the same sbt config file from Table 12.

VIII. Developer Manual

This section is to help the readers understand our code, especially those who are planning to continue further development on the project. The data flow, design, and implementation methodology have already been described in the previous sections. The data will be stored in HBase, a database supported by Solr and Spark. Having Scala and Apache Spark installed is a basic requirement to implement our code. Java and Scala installations are required to install Apache Spark. To verify the installation of Java and Scala, execute the two commands in Table 13.

Command	Desired output (if already installed)
$java -version	java version "1.7.0_71" Java(TM) SE Runtime Environment (build 1.7.0_71-b13) Java HotSpot(TM) Client VM (build 25.0-b02, mixed mode)
$scala -version	Scala code runner version 2.10.4 -- Copyright 2002-2013, LAMP/EPFL

Table 13. Installation Verification

In this project, we are using Spark 1.5.0 and Scala 2.10.4. Slight modifications to the code will be required if other versions are used. A complete reference manual for Scala and Spark 1.5.0, including the guidelines for installation, can be found at http://www.scala-lang.org/ and https://spark.apache.org/docs/1.5.0/, respectively.

In our project, we use Spark MLlib's RDD-based APIs for K-means clustering and LDA. MLlib is Spark's Machine Learning library. It consists of common learning algorithms and utilities including classification, regression, clustering, collaborative filtering, as well as lower-level optimization primitives and higher-level pipeline APIs. MLlib aims to make practical machine learning scalable and easy.

A. Clustering

K-means:

To develop our K-means clustering tool, we require an input file with tweet IDs and corresponding cleaned tweets in a CSV file, as shown in Figure 9. The parameters, like number of clusters K, number of iterations, and the size of the vectors to be created by Word2Vec, are tunable and can be chosen by developers. We experimented with different values of K and chose the ones which yielded the maximum mean cluster probabilities for each collection.

To apply the tool, the first step is to save the dataset in a Spark RDD before filtering it. Developers can use the spark.mllib.feature.Word2Vec feature to run Word2Vec on the filtered data for feature extraction and then normalize it before running K-means on it.

For K-means, developers can use the spark.mllib.clustering.KMeans feature. To run it on the processed data with 'K' number of clusters and 'R' number of iterations, use the command given below:

```
Val clusters = KMeans.train(Data, numClusters, numIterations)
```

The developer can compute the Within Set Sum of Squared Errors by using:

```
val WSSSE = clusters.computeCost(Data)
```

This can also be used to evaluate the clustering algorithm. The tool shall generate results as shown in Figure 10. Developers can refer to our algorithm in Clustering/ClusteringExperiments/kmeans/src/main/scala/ClusteringWithWord2Vec.scala.

DBSCAN:

The algorithm used for DBSCAN is already described in the implementation section. This project was developed using Python 2.7.10, Pyspark 1.5.2, Sklearn 0.16.0, Scipy 0.14.1 and Numpy 1.9.2. Earlier versions may not work. Optionally, Matplotlib - 1.4.3 can be used to run examples and make the included plots.

B. Topic Analysis

We performed topic analysis using Latent Dirchlet allocation. We started by using last semester's Latent Dirichlet allocation script and modified it to meet our needs. For each run, the following variables must be determined based on the developer's' needs: input, topicwordsfile, doctopicfile, doctopicfilecombined, doctopicfinal, numoftopics, and numofiterations. This script accepts arguments that populate these variables so that any user can run our tool.

The array stored in the variable stopWords can be added to, so that more stop words are filtered out of the text. The code is written to filter out "about", "means", "after", and "their". To filter out an additional word, such as "around", we can add it to the stopwords array as such:

```
val stopWords = Array("about","means","after","their", "around").
```

The script also filters out words that are less than four characters long. We then transform the documents into count vectors and run Spark MLlib's LDA algorithm according to the parameters that we chose.

Our script prints the topics with the top keywords associated with each topic and the probability that each keyword occurs in that topic. We then select the topic label by choosing the most probable keyword in the topic. If that keyword has already been chosen for a topic label, the next most probable word is chosen.

We write the probabilities that each topic belongs to each tweet in the files that are stored in the directory specified in the doctopicfilecombined variable. Each row in each file has the tweet ID and then 5 corresponding probabilities. The first probability belongs to Topic 1, the second to Topic 2, the third to Topic 3, etc.

It is important to acknowledge that some tweets do not have enough significant text in them to be classified. That is to say that when the documents are tokenized, the RDD that corresponds to the list of key tokens for that document is empty. This means that the content of the document was not significant enough to contribute to the topics, which also means that the document does not have a topic. In a collection of 43,000 tweets, about 880 tweets did not have document. This is because the terms in the tweet were not significant enough to have any key terms. This is not necessarily related to the length of the tweet.

C. Cluster Labeling and Probabilities

We use the results from clustering and topic analysis to label the clusters. For each tweet, we have the cluster that the tweet is assigned to, from the clustering results, and the possible topics that the tweet may relate to, with the corresponding probability that each topic pertains to the tweet. Please note that, since not all tweets have topics associated to them (due to lack of feature words), we need to remove such tweets from clustering results as well, to maintain the same set of tweets in both topic analysis and clustering results, before proceeding further.

We select two topics with the highest probabilities for each tweet and generate a mean-frequency matrix for each cluster from the data. It holds the frequency and mean probability of each topic being related to the cluster. We pick the two most frequent topics for each cluster and label the cluster by combining the labels of those topics. We also calculate the probabilities of tweets belonging to their clusters by adding the topic probabilities based on the clusters they belong to.

In this project, we had a Python script implementing the above logic. Developers may refer to the logic in Clustering/CL_8T.py. We saved the topic probabilities and clustering results into two different arrays. We then created a numpy array called aggregation with tweet ID, assigned cluster, top two topics that the tweet might belong to, and those topics' probabilities. We preferred numpy arrays to regular ones to be able to use sort and counter features, useful for generating plots, directly in our logic. From the aggregation

array, we generated a mean-frequency array with clusters, the frequency and mean probabilities that the cluster might be related to each of those topics. A sample mean-frequency matrix is given in Figure 15.

Cluster	Topic1	Topic2	Topic3	Topic4	Topic5	Topic6	Topic7	Topic8
'0'	1304, 0.0123	16, 0.2365	12, 0.2597	76, 0.3677	67, 0.3330	15, 0.2789	1293, 0.9820	21, 0.3886
'1'	2118, 0.4291	1917, 0.4732	1020, 0.3329	3757, 0.4035	2218, 0.3263	869, 0.5131	787, 0.3231	502, 0.2842
'2'	1737, 0.3432	644, 0.5752	925, 0.4522	1307, 0.2970	1284, 0.4866	449, 0.5321	211, 0.4113	219, 0.3853
'3'	2966, 0.3934	544, 0.1530	7167, 0.2850	3211, 0.3421	1646, 0.3275	4738, 0.4051	3181, 0.2583	4939, 0.448
'4'	1175, 0.3495	4728, 0.4354	1049, 0.3444	969, 0.2935	3194, 0.2907	829, 0.3041	2119, 0.4327	1165, 0.3759
'5'	62, 0.3107	327, 0.1498	36, 0.2762	68, 0.240	561, 0.6632	23, 0.3062	98, 0.2303	13, 0.2706

Figure 15. Mean Frequency Matrix

From the mean-frequency matrix, we generated a cluster-topics array by assigning the two most frequent topics related to each cluster. For the mean-frequency matrix in Figure 15, the cluster-topics array created will be as shown in Table 14.

Cluster - Most frequent topics	
Cluster	**Topics**
'0'	Topic 1, Topic 7
'1'	Topic 4, Topic 5
'2'	Topic 1, Topic 4
'3'	Topic 3, Topic 8
'4'	Topic 2, Topic 5
'5'	Topic 5, Topic 2

Table 14. Cluster-Topics Array

We then labelled the clusters by concatenating the labels of the two main topics assigned to each. To generate the cluster probabilities of tweets, we note the main topics of the cluster that each tweet belongs to, and add the probabilities of the tweet belonging to those topics (which we obtained from topic analysis).

The developer can use the logic explained above to generate cluster probabilities and cluster labels. The script is expected to create two output text files - one with all the tweets, their clusters and their cluster probabilities as specified in Figure 14, and other with cluster labels as shown in Figure 13.

D. Pulling from HBase

To read from HBase, we used the HBaseIntegration.scala script that was written by Matthew Bock and then modified by Eric Williamson. Using techniques from the CLA team, we wrote the DataRetriever2.scala script, which called HBaseIntegration.scala. The _tableName variable is the name of the HBase table that we wish to pull from. The variable _colFam is the column family of the column that we need to read from, while the variable, _col, is the column in HBase that we want to read from. The variable, _classColFam, is the column family that we are using to read from. The variable, _classCol, is the column that we are using to pull from, and _class is the real world event of documents that we want to pull.

HBase.scala contains the main method, which cleans the tweets retrieved from DataRetriever2.scala. The tweets are cleaned via the CleanTweet class, which is defined in CleanTweet.scala. The CleanTweet.scala script was adapted from the CLA team.

E. Writing to HBase

To help the CTA team write topic analysis and clustering results to HBase in parallel, we separated the scripts for writing topic analysis results to HBase and the scripts for write clustering results to HBase into different directories. Both directories will require HBaseIntegration.scala, as mentioned in the section on pulling from HBase.

Writing topic analysis results requires DataWriter.scala and HBase.scala. In DataWriter.scala, the variable _tableName corresponds to the HBase table that we want to write to. The variable _colFam1 refers to the column family that we want to write to, and _colLabel1 and _colProb1 correspond to the columns that we want to write to. We keep a labelMapper() function that maps the real world events to a number to easily write the topic labels to HBase. If a new real world event is added, the developer should add an additional mapping of a double value to a list of topic labels, separated by a semicolon.

Writing clustering results requires DataWriter2.scala and WriteCluster.scala. In DataWriter2.scala, the variable _tableName corresponds to the HBase table that we want

to write to. The variable _colFam2 refers to the column family that we want to write to, and _colLabel2 and _colProb2 correspond to the columns that we want to write to. We do not have to worry about mapping labels based on real world events, because we read the cluster labels from the results file itself.

F. File Inventory

File(s)	Comment
HBase/HBaseInteraction.scala	Code for interacting with HBase
HBase/DataRetriever2.scala	Code for pulling documents from HBase
HBase/CleanTweet.scala	Code for cleaning tweets
HBase/DataWriter.scala	Code for writing topic analysis results to HBase
HBase/HBase.scala	Driver for writing topic analysis results to HBase
HBase/Cluster/DataWriter2.scala	Code for writing clustering results to HBase
HBase/Cluster/WriteCluster.scala	Driver for writing clustering results to HBase
LDA/lda.scala	LDA code for topic analysis
Clustering/ClusteringExperiments/kmeans/src/main/scala/ClusteringWithWord2Vec.scala	K-means clustering code with Word2Vec feature extraction
Clustering/ClusteringExperiments/pypardis/*	DBSCAN code - Note that this does not run efficiently on the cluster and cannot handle large feature vectors with the current cluster
Clustering/CL_8T.py	Python script for cluster labeling, generation of cluster probabilities and generating visualizations
Clustering/pick_cluster.py	Python script for evaluation. Takes clustering results directory as input, calculates the mean probabilities for each of them, writes the output to a text file

Table 15. File Inventory

G. Future Work

With regard to DBSCAN, we propose either porting the implementations listed to work with the Spark version running on the cluster, or upgrading the Spark running on the

cluster so that we can efficiently run DBSCAN and have a comparison with the K-means algorithm.

Our intent was to perform experiments on both tweets and web pages. Our tools can handle both tweets and webpages, however, the webpages were not available on HBase in time for us to work with. In future semesters, parameters for topic analysis and clustering should be tuned so that the web pages on HBase can have topics and clusters associated with them for the IR system.

We had also intended to do more thorough evaluations of our clusters. We had intended to evaluate our clustering with silhouette scores and our topics with parity, however, due to time constraints, clustering evaluation was done using mean cluster probability values and topic analysis evaluation was mainly with human judgment. We looked for word intrusion in the word distributions for each topic, and we wanted topics that had minimal word overlap in their word distributions. In evaluating our clusters, we aggregated the probabilities for and chose cluster counts with the highest probabilities. It is also suspected that future CLA and future CTA teams could work together for evaluations. We noticed that "kentucky" was one of the topics of the China Factory Explosion. Perhaps Kentucky was a popular topic in discussions regarding the China Factory Explosion. However, it is also possible that many documents about the Kentucky Accidental Child Shooting were actually misclassified as being about the China Factory Explosion. Our results could help future CLA teams evaluate their classifiers. Conversely, classification results could help future CTA teams in their evaluations. The future CLA team could give the future CTA team a sample of documents evenly distributed across N real world events. The future CTA team could perform clustering and topic analysis on N clusters, and the future CTA team could evaluate if the documents were clustered correctly and if the topics were reasonable.

Finally, future efforts should work on streamlining the clustering and topic analysis workflows. The current workflow is that the LDA script and the K-Means clustering script are run. Then the results from those two scripts are evaluated in another script to find the cluster labels and cluster probabilities. A program should be written that brings all of these scripts together so that the work is streamlined and only a single program must be run. Additionally, the LDA and K-means clustering scripts should be written to read tuning parameters from configuration files, so that the variables do not have to be changed within the code.

IX. Acknowledgements

We would like to thank Professor Edward Fox for his guidance toward the successful completion of our project. We would also like to thank the Graduate Research Assistant, Dr. Sunshin Lee, for his advice on technologies and techniques, which helped us complete this project. We would also like to thank all of the project teams in the CS 5604 Fall 2016 class, as well as the Topic Analysis team from CS 5604 Spring 2016 and the Clustering teams from CS 5604 Fall 2015 and CS 5604 Spring 2016. We also acknowledge and thank the National Science Foundation (NSF) for supporting the Integrated Digital Event Archiving and Library (IDEAL) project with grant number IIS-1319578, and Global Event and Trend Archive Research (GETAR) project with grant number IIS-1619028). Finally, we acknowledge and are grateful for the problem based learning approach of the CS 5604 class.

X. References

[1] Anderson, C. (2003, February 1). Adaptive Website Research. In *Corey's Adaptive Web Site research*. Retrieved September 11, 2016 <http://www.the4cs.com/~corin/research/aws/>

[2] Blei, David M., Andrew Y. Ng, and Michael I. Jordan. "Latent Dirichlet Allocation." *Journal of Machine Learning Research*(2003): 993-1022. Web. 15 Sept. 2016. <https://www.cs.princeton.edu/~blei/papers/BleiNgJordan2003.pdf>.

[3] Ester, M., Kriegel, H., Sander, J., & Xu, X. (1996). A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. *AAAI Press*, 226-231.

[4] Kalidas, R., Thumma, S. R., & Torkey, H. (2015 May 13). Document Clustering for IDEAL (Master's thesis). < https://vtechworks.lib.vt.edu/handle/10919/52341>

[5] Manning, C. D., Raghavan, P., & Schütze, H. (2008). *Introduction to Information Retrieval* (Online ed., pp. 349-401). Cambridge, England: Cambridge University Press.

[6] Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013, January 16). Efficient Estimation of Word Representations in Vector Space [Electronic version]. *Computation and Language*. <https://arxiv.org/pdf/1301.3781.pdf>

[7] Tang, L., Thorve, S., & Vishwasrao, S. (2016, May 3). *CS5604: Clustering and Social Networks for IDEAL* <http://vtechworks.lib.vt.edu/handle/10919/70947>

[8] Mehta, S., Vinayagam R. (2016, May 4). *CS5604: Extracting Topics from Tweets and Webpages for IDEAL*. <https://vtechworks.lib.vt.edu/handle/10919/70933>

[9] Xie, P., & Xing, E. P. (2013). Integrating Document Clustering and Topic Modeling. *Proceedings of the Twenty-Ninth Conference on Uncertainty in Artificial Intelligence*, 694-703.

[10] Zhai, C., & Massung, S. (2016). *Text Data Management and Analysis*. San Francisco, CA: Association for Computing Machinery and Morgan & Claypool Publishers.

[11] Cordova, Irving. *GitHub*. N.p., 13 June 2016. Web. 11 Oct. 2016. <https://github.com/irvingc/dbscan-on-spark>.

[12] Patwary, Mostofa, and William Hendrix. *Parallel Data Clustering Algorithms*. Northwestern University, 25 Mar. 2016. Web. 11 Oct. 2014. <http://cucis.ece.northwestern.edu/projects/Clustering/index.html>.

[13] O'Neill, Benjamin. *GitHub*. N.p., 3 Feb. 2016. Web. 11 Oct. 2014. <https://github.com/bwoneill/pypardis>.

[14] Fox, Edward A. 2016. *CS 5604 Information Storage and Retrieval* [Syllabus]. Location: Department of Computer Science at Virginia Tech.