

Clustering and Social Network

Final Project Report

4 May 2016

CS5604 Information Storage and Retrieval

Virginia Polytechnic Institute and State University

Blacksburg, Virginia

Spring 2016

Submitted by

Tang, Lijie

lijie@vt.edu

Thorve, Swapna

swapna6@vt.edu

Vishwasrao, Saket

saketv02@vt.edu

Instructor

Prof. Edward A. Fox

Abstract

The Integrated Digital Event Archiving and Library (IDEAL) project of Virginia Tech provides services for searching, browsing, analysis, and visualization of over 1 billion tweets and over 65 million webpages. The project development involved a problem based learning approach which aims to build a state-of-the-art information retrieval system in support of IDEAL. With the primary objective of building a robust search engine on top of Solr, the entire project is divided into various segments like classification, clustering, topic modeling, etc., for improving search results. Our team focuses on two tasks: clustering and social networks. Both these tasks will be considered independent for now.

The clustering task aims to congregate documents in groups such that documents within a cluster would be as similar as possible. Documents are tweets and webpages. The k-means algorithm is employed for clustering the documents. Two methods were employed for feature extraction, namely, TF-IDF score and the word2vec method. Evaluation of clusters is done by two methods – Within Set Sum of Squares (WSSE) and analyzing the output of the topic analysis team to extract cluster labels and find probability scores for a document. The later strategy is a novel approach for evaluation. This strategy can be used for assessing problems of cluster labeling, likelihood of a document belonging to a cluster, and hierarchical distribution of topics and cluster. We execute our algorithms on different collections and present the results. The social networking task will extract information from Twitter data by building graphs. Graph theory concepts will be applied for accomplishing this task.

Using dimensionality reduction techniques and probabilistic algorithms for clustering, as well as improving the cluster labelling and evaluation, are some of the things that can be improved on our existing work in the future. Also, the clusters that we have generated can be used as an input source for Classification, Topic Analysis and Collaborative filtering tasks in order to obtain more accurate results.

Contents

Table of figures	1
Table of tables.....	3
1. Overview	4
2. Literature review	5
2.1 Why analyze tweets?	5
2.2 Clustering	7
2.2.1 Flat clustering	7
2.2.2 Hierarchical clustering.....	7
2.2.3 k-means clustering algorithm.....	9
2.3 Social network graph	11
3. Requirements	14
3.1 Functional requirements	14
3.1.1 Clustering.....	14
3.1.2 Social network	16
3.2 Collaboration with other teams.....	17
3.2.1 Clustering.....	17
3.2.2 Social network	17
3.3 Technology requirements.....	17
3.3.1 Clustering.....	17
3.3.2 Social network	18
4. Design and implementation.....	19
4.1 Clustering	20
4.1.1 Design and implementation	20
4.2 Creating social network	23
5. Evaluation.....	28
5.1 Within Set Sum of Squared Errors - WSSE	28
5.2 Analyzing the output of topic analysis team	33
6. Milestones and work progress	39
6.1 Clustering.....	39
6.2 Social network.....	41
7. Developer manual.....	42

7.1 Using Scala and SBT as build tool for Eclipse on CDH 5.3	42
7.2 Feature Extraction and Normalization.....	43
7.3 Clustering using MLlib	45
7.4 Evaluation using LDA (topic analysis) and clustering output	46
8. User manual	54
8.1 Running the Scala-Spark program on shell	54
8.2 Evaluation of clustering result with LDA output.....	56
9. Future work	61
Appendices	62
Appendix A - Abbreviations	63
Appendix B - File Inventory	64
Acknowledgement	65
References	66

Table of figures

No	Figure title
1.	What happens in a Twitter minute
2.	Hurricane Sandy conversation on Twitter
3.	Clustering in 2-D plane
4.	Flat and hierarchical clustering
5.	k-means clustering
6.	A sample social network graph
7.	Influence factor on for information query
8.	IF calculation using network data
9.	Functional component of clustering
10.	Schema design for clustering
11.	Sample output of Twitter accounts crawler
12.	Flow diagram of the system
13.	Clustering of tweets based on tweet data
14.	Clustering of users based on hashtags
15.	The Twitter social network
16.	Network and clustering
17.	Comparison of cluster sizes
18.	Evaluating the cluster number using WSSE for z700 collection
19.	WSSE scores vs number of clusters
20.	#NAACPBombing
21.	#GermanWings
22.	#houstonflood
23.	#Obamacare
24.	#4thofJuly
25.	Evaluation of clusters based on topic analysis output
26.	Hierarchical representation 1 of evaluation output
27.	Hierarchical representation 2 of evaluation output
28.	Comprehensive overview of social network task timelines and functional requirements
29.	Scala IDE for Eclipse Juno
30.	Vector generation using TF-IDF
31.	Training word2vec model
32.	Generating document vectors
33.	Main clustering steps

34.	Clustering output for tweets-541
35.	LDA output for tweets-541
36.	LDA output for tweets-541, with top 5 topic words and their probabilities
37.	Intermediate evaluation output for tweets-541 mentioning top 2 topics resembling the cluster
38.	Final evaluation output for tweets-541
39.	Conceptual workflow of evaluation
40.	Input parameters
41.	Reading clustering output
42.	Reading LDA output
43.	Calculating topic frequency per cluster
44.	Finding mean and deviation of topics in cluster
45.	Calculate probability score for each document
46.	Clustering results based on tweets
47.	Clustering results based on users
48.	Clustering based on webpages, indexed by tweet ID
49.	Clustering is webpages indexed by URL
50.	Clustering output for tweets-541
51.	LDA output for tweets-541
52.	LDA output for tweets-541, with top 5 topic words and their probabilities
53.	Intermediate evaluation output for tweets-541 mentioning top 2 topics resembling the cluster
54.	Final evaluation output for tweets-541
55.	Input parameters

Table of tables

No	Figure title
1.	Evaluation of users for tweets with WSSE
2.	Output of topic analysis team, document-topic matrix
3.	Output of clustering team
4.	Aggregation of topic analysis data and clustering data
5.	Mean topic frequency across clusters
6.	Evaluation output defining local and global topics and cluster labels
7.	Heuristic cluster labeling after extracting top 2 topics for clusters in collection tweets 602
8.	Clustering task breakup
9.	Files for evaluation
10.	Folder structure for evaluation task
11.	HBase schema design
12.	Files for evaluation
13.	Folder structure for evaluation task
14.	HBase schema design

1. Overview

Social media provides staggering amounts of data. Extracting knowledge from these volumes requires automation. Computing quickly over this data is a challenge for both algorithms and architectures. Recently, microblogging has become a popular trend which is responsible for a large amount of information dissemination. The most prevalent microblogging service is Twitter - a popular tool for short, frequent communication. Twitter users tweet about any topic, status or event and follow others to receive their tweets. Anyone around the world can use Twitter to talk about daily activities and seek information. Observing such a high usage of Twitter has urged many businesses and research groups to analyze these tweets to evaluate possible connections and outcomes (predictions). Several APIs and tools have been developed around this notion to predict future events.

As the importance of Twitter rises, the associated research in the fields of information science, corporate and civil society sectors is booming. We attempt to develop a network graph (based on graph theory) considering users as nodes. Edges between them would attribute to characteristics like following, re-tweet, etc. We also make an effort to group these tweets into clusters in an unsupervised manner via flat clustering. The technology stack will primarily include Java, Scala, Spark and its associated libraries.

We realize that analyzing tweets is challenging due to the short text and non-conformance to grammatical rules. These short messages are generated at a prodigious rate, which makes us consider timestamps of the tweets. A tweet can originate from various sources like newspaper, website, TV, etc. Each tweet shows the interest of the publisher. So, it is natural to consider human factors in the Twitter data analysis.

Keeping in mind these challenges, we will first try to build a social network, which shows the interaction between user accounts and the importance of each account. Then, we can use that information to find the similarity between contents to reveal their importance. For clustering, we will have to be cautious in identifying the words that form the essence of the tweets and use this further for extracting URLs/webpages. We will employ clustering for grouping tweets w.r.t. multiple attributes. Clustering tweets effectively will influence the search functionality of the system.

The remainder of this report provides further background on clustering in information retrieval and graph representations for social networks.

2. Literature review

2.1 Why analyze tweets?

Social media has made social networks ubiquitous, and also given researchers access to massive quantities of data for empirical analysis. These data sets offer a rich source of evidence for studying dynamics of individual and group behavior, the structure of networks and global patterns of the flow of information. For example, Facebook consists of over 400 million active users sharing over 5 billion pieces of information each month. [5] The microblogging services like Twitter, are causing a worldwide craze.



Figure 1: What happens in a Twitter minute.

Courtesy: <http://www.adweek.com/socialtimes/happens-Twitter-minute-infographic/207992>

Over 280 million monthly active users send more than 500 million tweets per day, and 80 percent of those active users send tweets from mobile, according to information

from the company [4]. With more than 1.45 million queries every minute, search is one of the most popular activities on Twitter. The next most popular activity is timeline viewing — there are 1.22 million timeline views every minute. The volume of tweets is also enormous, with more than 347,000 140 character messages going out every 60 seconds [4]. Tweets can be categorized into trends, and are related with tags and follower/following social relationships. The categorization is neither accurate nor effective due to the short length of tweet messages and noisy data corpus.

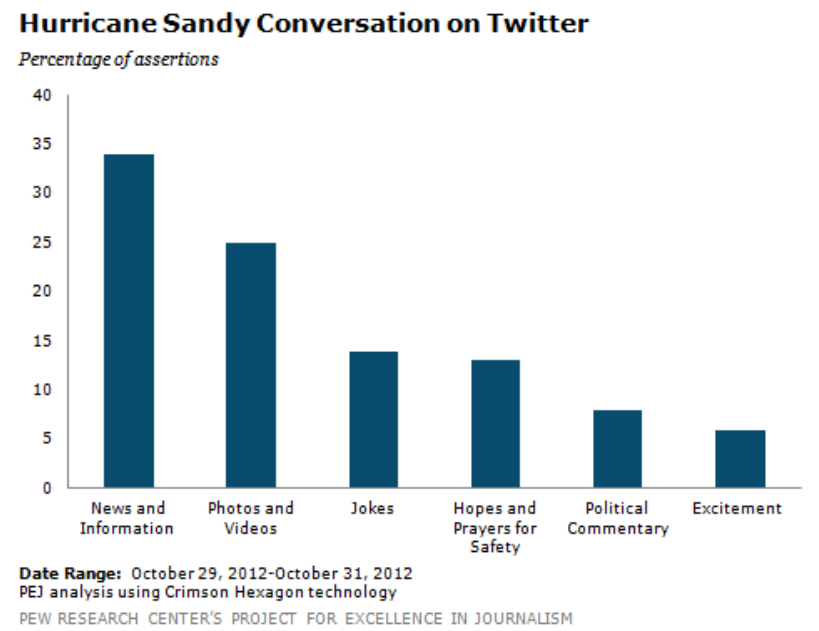


Figure 2: Hurricane Sandy conversations on Twitter
 Figure courtesy – [6]

Twitter data has been a center of attraction for deriving meaning of the short messages posted by people. We can group people belonging to similar geo-locations who are tweeting on particular events. We can derive a public opinion about certain happenings. Above all of these, we can even help predict certain events to come in the future or what is its likeliness. Twitter gives us a great power to predict elections, analyze natural hazard data or social TV interactions [6][7][8]. Twitter has opened up a whole new arena for exploring various domains.

As a part of the IDEAL project at Virginia Tech, our team will develop an approach for clustering tweets as well as clustering webpages from tweets. With such an overwhelming amount of data, it is definitely challenging to identify and group webpages resulting from crawls or sited in tweets. Setting up crawlers and finding data for building a network also tends to be challenging.

2.2 Clustering

Clustering is a process of partitioning a set of data (or objects) in a set of meaningful sub-classes, called clusters. It helps users understand the natural grouping or structure in a data set. Clustering has a wide range of applications, from spatial data analysis to market research. The key input to a clustering algorithm is the distance measure. In Figure 3, the distance measure is distance in the 2D plane. This measure suggests three different clusters in the figure. In document clustering, the distance measure is often also Euclidean distance. Different distance measures give rise to different clusters. Thus, the distance measure is an important means by which we can influence the outcome of clustering.

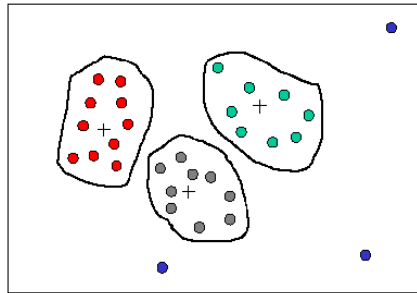


Figure 3: Clustering in 2-D plane

Clustering can be broadly classified into flat clustering and hierarchical clustering.

2.2.1 Flat clustering

Flat clustering, also called as partitioning clustering, creates flat set of clusters without any explicit structure that would relate clusters to each other. Flat clustering methods are conceptually simple, but they have a number of drawbacks. Most of the clustering algorithms, like k-means, require a pre-specified number of clusters as input and are non-deterministic.

Flat clustering may be viewed as an optimization problem. An instance is a particular clustering setting: a set of objects, a representation with a similarity measure, and a number of clusters. An assignment is a clustering in this setting. The objective/criterion function returns a value for all clusters and the goal is to find a clustering scenario with an optimal value. In most cases, to find such a clustering would require an exhaustive search, and most partitioning clustering algorithms are local search strategies that are only guaranteed to find a local optimum.

2.2.2 Hierarchical clustering

Hierarchical clustering builds a cluster hierarchy, or in other words, a tree of clusters. Hierarchical clustering output is structured and more informative than flat clustering. Hierarchical clustering algorithms are further subdivided into two types:

- a. agglomerative methods - a bottom-up cluster hierarchy generation by fusing objects into groups and groups into higher clusters.

- b. divisive methods - a top-down cluster hierarchy generation by partitioning a single cluster encompassing all objects successively into finer clusters.

Agglomerative techniques are more commonly used. Hierarchical clustering does not require knowing the pre-specified number of clusters. However, this advantage comes with the cost of the algorithm complexity. Hierarchical clustering algorithms have a complexity that is quadratic in the number of documents compared to the linear complexity of at algorithms like k-means or EM. Figure 4, shows an example of how data can be clubbed for flat and hierarchical clustering.

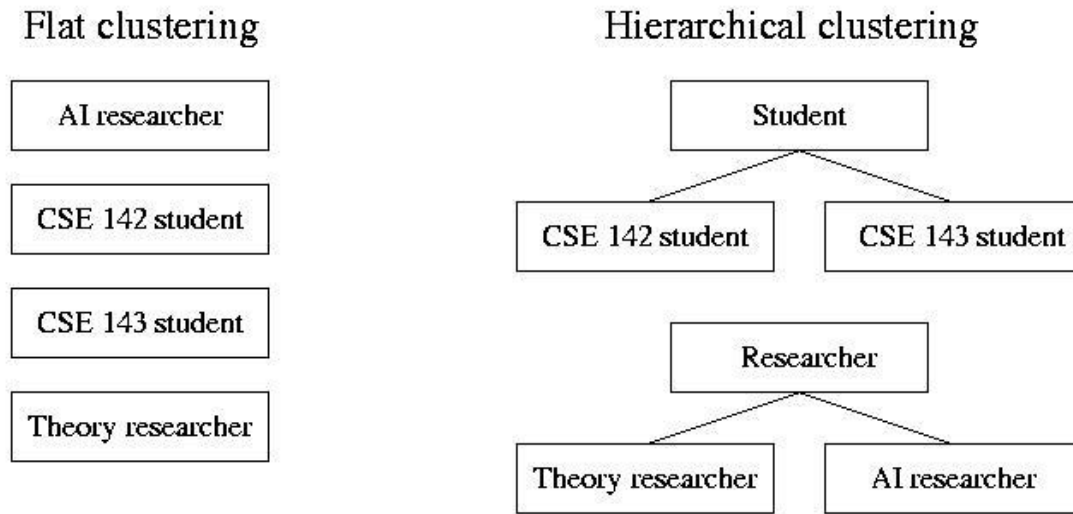


Figure 4: Flat and hierarchical clustering
Image courtesy: <http://www.the4cs.com/~corin/research/aws/>

Here, we will focus on clustering in information retrieval systems. In terms of our task of clustering documents (tweets and webpages) we can say that, the expected output has to be a collection of tweets (documents) that are “similar” to one another and thus, can be treated collectively as one group; but as a collection, they are sufficiently different from other groups. Applications of clustering in the information retrieval domain include search result clustering, collection clustering, cluster based retrieval for high precision, clustering to increase recall in language models and so on. [1]

2.2.3 k-means clustering algorithm

The k-means clustering algorithm is known to be efficient in clustering large data sets. This algorithm is one of the simplest and best known unsupervised learning algorithm. The k-means algorithm aims to partition a set of objects, based on their attributes/features, into k clusters, where k is a predefined constant. The algorithm defines k centroids, one for each cluster. The centroid of a cluster is formed in such a way that it is closely related, in terms of similarity (where similarity can be measured by using different methods such as Euclidean distance or Extended Jacquard) to all objects in that cluster. Technically, what k-means is interested in, is the variance. It minimizes the overall variance, by assigning each object to the cluster such that the variance is minimized. The basic steps of the k-means clustering algorithm are as follows:

- i. Pick k objects at random and let them define k clusters.
- ii. Calculate cluster representatives.
- iii. Make new clusters, one per cluster representative. Let each text belong to the cluster with the most similar cluster representative.
- iv. Repeat from (ii) until a stopping criterion is reached.

Figure 5, below, depicts the algorithm described above.

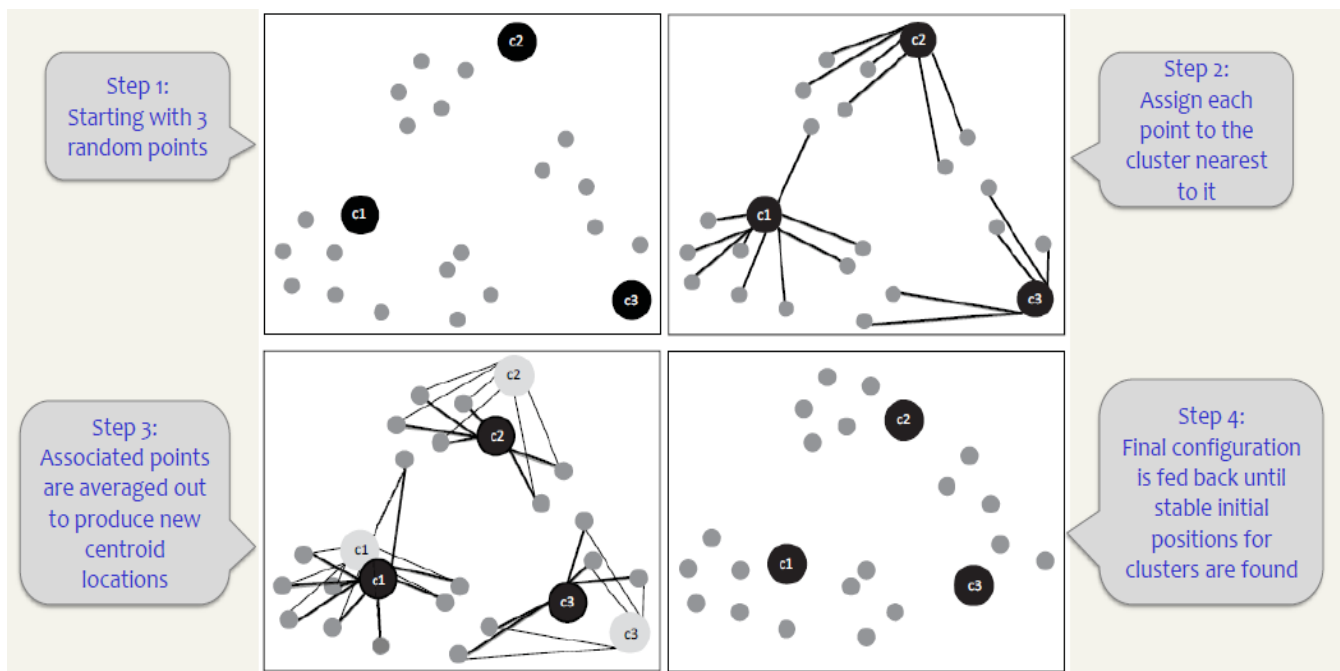


Figure 5: k-means clustering

The first step defines a random initial partition. There are many other ways of constructing it and the result depends on which one is used. As cluster representative the mean (the centroid) of the objects in the cluster is usually used. Other variants are to let the median or a few specific objects represent the cluster. When the clustering is fuzzy, objects may belong to several clusters and thus, the cluster representative may be calculated taking this into consideration.

The stopping criterion is normally when no objects change clusters, or when very few change clusters between iterations. It may also be to stop after a predefined number of iterations, since most quality improvement usually is gained during the first iterations.

The time complexity of the k-means algorithm is $O(knI)$, where k is the number of clusters, n the number of objects and I the number of iterations (which is dependent on the stopping criterion). In each iteration the cluster representatives and the kn similarities between all objects and all clusters must be computed. The k-means algorithm requires a number of clusters as input. That is, one has to guess the appropriate number. Of course it is possible to run the algorithm with several different numbers of clusters and report only the clustering with the best result. In a general partitioning algorithm both splitting and division of clusters are allowed and theoretically the result has the optimal number of clusters.

2.3 Social network graph

Social networks have emerged as a critical factor in information dissemination, search, marketing, expertise and influence discovery, and potentially an important tool for mobilizing people.

What is social network on Twitter

In traditional society research, social network refers to the social connections among people. In the movie “Social Network”, it refers to Facebook or other online tool which extend the social connections in society to the internet. On Twitter, social network is the connections among Twitter accounts. In this research, three kinds of connections: following, followed and retweet are considered.

Social network on information retrieval

Different from Facebook, where people extend their social connections to internet, people on Twitter can follow anyone they want without following back. People follow celebrities, news agencies and authorities to get the latest updates (news information); people retweets important (or interesting) tweets so it flows on the network. In information retrieval, when there is a user try to find information on Twitter using a query, the social network information can be used to rank the importance of all the query results.

With interfaces that allow people to follow the lives of friends, acquaintances and families, the number of people on social networks has grown exponentially since the turn of this century [10]. A social network is a social structure made up of a set of social actors (such as individuals or organizations), sets of dyadic ties, and other social interactions between actors. The social network perspective provides a set of methods for analyzing the structure of whole social entities as well as a variety of theories explaining the patterns observed in these structures [9].



Figure 6: A sample social network graph

Figure 7 shows proposed work on using social network on information retrieval. For each given query, the search algorithm will return a bag of results based on the content similarities to the query. We extract the social network behavior to give an impact factor (IF) to Twitter and tweet. In the end, we present the bag of results ranking from high IF to low IF.

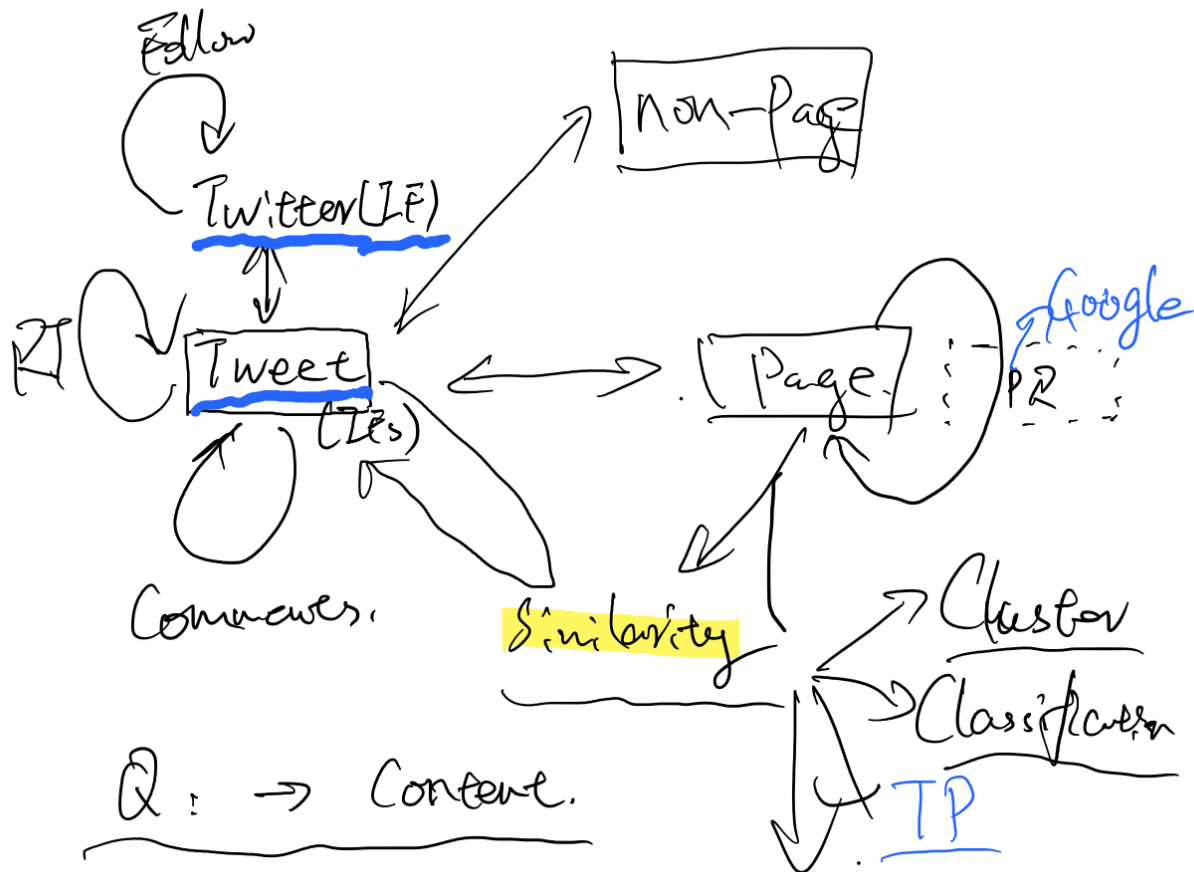


Figure 7: Influence factor on for information query

Figure 8 shows the process of calculating an influence factor from social network data. We mimic the process PageRank by using following, followed and retweet counts. When a tweet is broadcasted or retweeted from a high IF account, it will accumulate higher IF. Also, the IF of Twitter accounts are accumulated from tweets they broadcasted or retweeted.

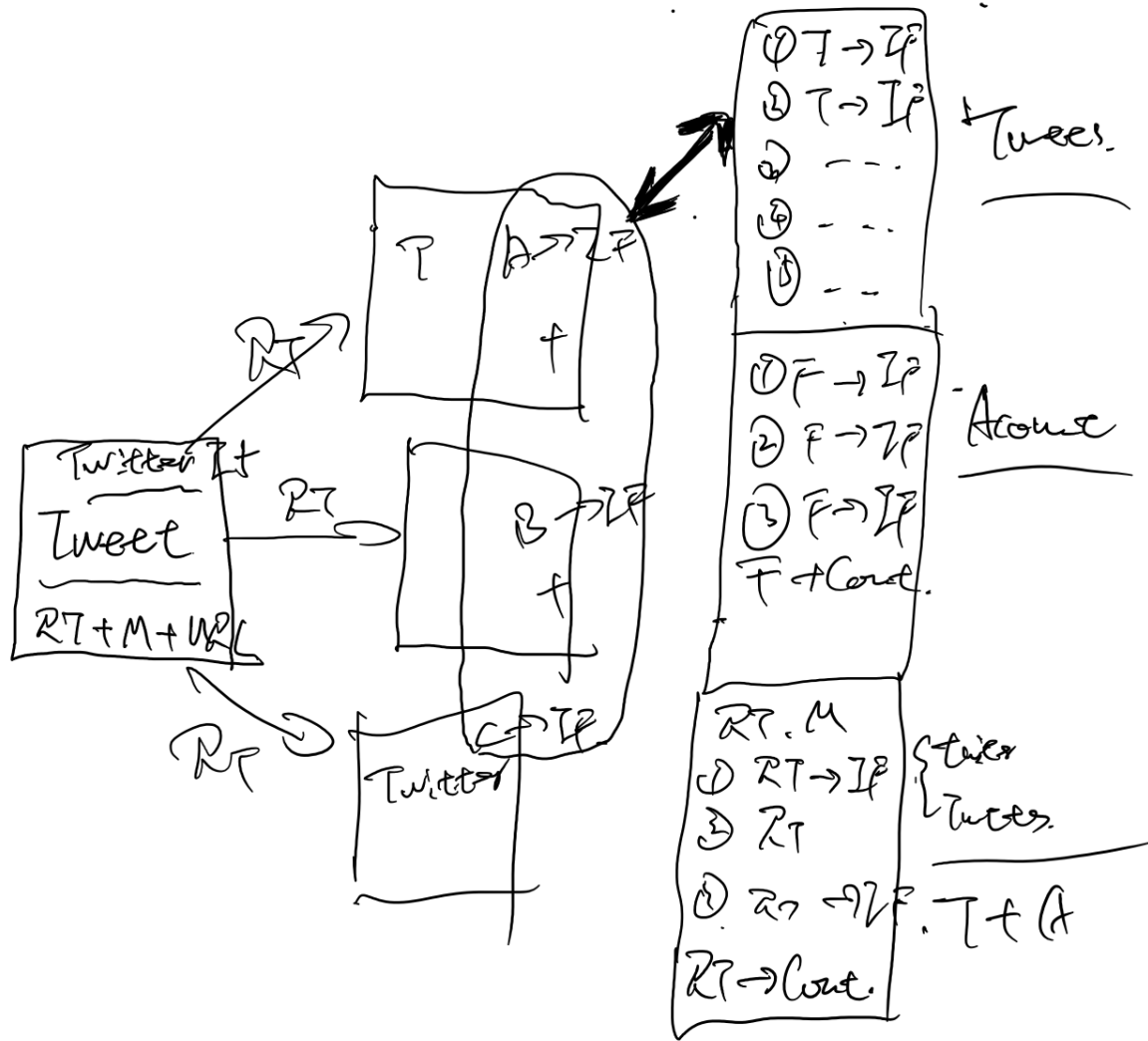


Figure 8: IF calculation using network data

A quick note on the work done:

We crawled the follower counts of all the accounts in our dataset. We crawled all the following information as well. The retweet relationship was extracted using the RT tag and @tag.

3. Requirements

The clustering task will focus on creating meaningful clusters for tweets and webpages which will help the scoring mechanism in order to improve search results. We need to make sure that documents in the same cluster behave similarly with respect to relevance to information needs [1]. The social network graph is going to aid in discovering the connections between user accounts based on various attributes like re-tweet, follower, etc.

3.1 Functional requirements

3.1.1 Clustering

The functional requirements for clustering (shown in Figure 9) can be divided into the following:

- a. Feature extraction
- b. Model building and evaluation
- c. Storing the results

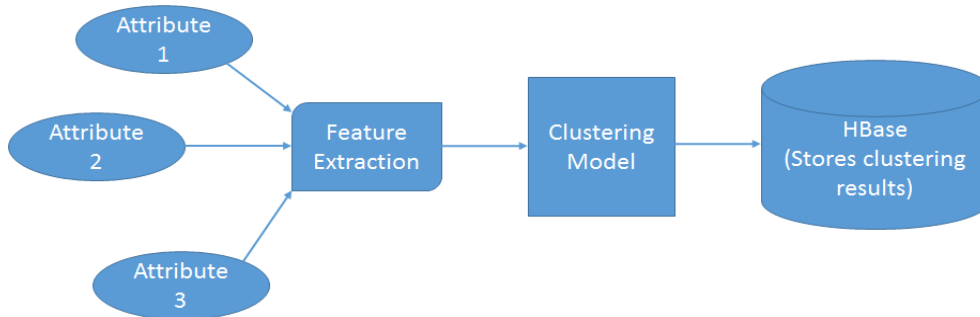


Figure 9: Functional component of clustering

3.1.1.1 Feature extraction

In order to build the model for clustering using k-means as an algorithm, we need to prepare data. For this purpose, we will have to find out which attributes from the documents (tweets and webpages) can be used as grouping attributes. This is the feature extraction process. We will primarily focus on hashtags (#) and username mention (@) in the tweets creating clusters of users. In further versions, we can plan to incorporate the re-tweet information, important terms in the document w.r.t. the hashtags and the latest timestamp of the tweet. This data will be prepared in a matrix form or any other representation required by the clustering model.

3.1.1.2 Model building

The inputs to the clustering algorithm will be the number of clusters, number of iterations and the distance measure (key factor). An iterative approach will be adopted to obtain better accuracy for clusters.

3.1.1.3 Evaluation of clusters

Evaluation of clustering is categorized in 2 broad areas: internal evaluation criterion and external evaluation criterion. Internal criterion evaluates the result with respect to information intrinsic to the data alone. We look for high intra-cluster similarity and low inter-cluster similarity. The external criteria will evaluate the result with respect to a pre-specified structure. It is based on gold standard classes derived from inter-judge agreement. These measures include purity, normalized mutual information, Rand index and F measure [1] [12]. The evaluation section in chapter 4, will provide details about evaluation done for the clustering task.

3.1.1.4 Storing results in HBase

The results are stored in the HBase schema. The proposed schema design as of now comprises of two tables in HBase. We will have a single table for tweets and webpages. The second table will have topic and word information. This table will be populated by the topic analysis team. The class might consider having two more tables in future, one for tracking user activities through the logs and the second to store graph.

The column families for clustering are illustrated in Figure 8:

Row key	Cleaned tweet text	Hashtags	Tweet Cluster number	Tweet Cluster labels	Webpage Cluster number	Cluster labels	Collection
T-6790452792			2	Cat, dog			
T-8096346666			1	Lily, roses, gardens			
W-56241292792					2	Cat, dog	

Tweet/webpage unique ID
Tweet – tweetID
Webpages – URL or something appropriate

Example of collection management column family for tweets

Clustering column family for tweets

Clustering column family for webpages

Figure 10: Schema design for clustering

The clustering task has to deal with only one table – the table for tweets and webpages. This table is illustrated in Figure 10. Clustering task will have two column families, column family for tweets as “clustering_tweets” and column family for webpages as “clustering_webpages”. Each row will be uniquely identified by a rowkey. The rowkey is

decided by the Solr team. The rowkey for tweets will be the tweet id and for webpages it will be URL or a combination of URL and timestamp or anything appropriate. The figure shows a random number as a rowkey for webpages as a mere example. The rowkey for webpages is subject to change in future.

Let us consider the tweet column family. This column family will have several fields (columns) which will be output of various other teams. For e.x., the collection management team might have 'cleaned tweet text', 'hashtag', 'timestamp', 'mention' as fields in the tweet column family. Similarly, the clustering task has two fields which belong to the tweet column family, namely, 'cluster number' and 'cluster labels'. Cluster number will specify the number to which a tweet belongs. This number has no semantic meaning, because clustering is unsupervised and we have unlabeled data. Cluster labels will be terms or phrases that describe the cluster.

Let us consider the webpage column family. This column family will have several fields (columns) which will be output of various other teams. For e.x., the collection management team might have 'URL', 'timestamp', 'title, etc., as fields in the webpage column family. Similarly, the clustering task has two fields which belong to the webpage column family, namely, 'cluster number', 'cluster labels' and 'collection'. Cluster number will specify the number to which webpage belongs. This number has no semantic meaning, because clustering is unsupervised and we have unlabeled data. Cluster labels will be terms or phrases that describe a cluster of webpages. The collection field has no usage right now. This is kept for future use by Solr team or any other group as per the suggestion of Prof. Fox.

3.1.2 Social network

The functional requirements basically involved the crawler which will get information of Twitter accounts required for building the network. Refer to Figure 19, from chapter 5 section 5.2, for the functional requirements. A crawler was written to fetch the account details like number of friends and followers each account has, along with their identifiers. Following is a sample output of the Twitter account crawler which was written by Lijie. Please note that this is just a sample output.

```
"GreenHokie": {"friendCounts": 42, "followerCounts": 55, "followerID":  
[14430993, 2440726016, 3054270807, 29302754, 3100782291, 2880910836,  
10937622, 2882956091, 226117617, 210684204, 2412615320, 296609663,  
1246708812, 2212036884, 2423925391, 2323080786, 28126118, 2887401, 95969660,  
1938549276, 21163085, 23791540, 313338987, 1242266869, 151553857, 2396173435,  
1492776728, 354691252, 38050035, 1446798476, 1050608772, 372480639, 63047856,  
1297395985, 218524147, 809493102, 760459615, 82980034, 787114753, 15866769,  
484037388, 343584449, 330695754, 66393123, 317725123, 35513773, 52396265,  
281693354, 16255393, 81903927, 229163646, 159462573, 38501138, 154008334,  
35543439], "userID": "205276031"}
```

Figure 11: Sample output of Twitter accounts crawler

Explanation of the keys in the sample output :-

GreenHokie : username of the Twitter account.

userID: In this example, it is the identifier of user with username 'GreenHokie'

friendCounts : how many friends does the user 'GreenHokie' have on Twitter.

followerCounts : how many followers does the user 'Green Hokie' have on Twitter.

followerID : Identifiers list of the followers for the user 'GreenHokie'.

3.2 Collaboration with other teams

3.2.1 Clustering

- a. Collection management team
 - This team will provide us cleaned data for our feature extraction task.
- b. Topic analysis team
 - This team will help us with the problem of labelling and evaluation of clusters.
- c. Solr team
 - This team will utilize the results stored in the HBase.
- d. Front end team
 - In future versions, we might be able to give input to the front end team for facets.

3.2.2 Social network

- a. Collection management team
 - We will get cleaned data from this team which will be used for building the network graph.
- b. Front end graph team
 - This team will help us visualize the network created out of the documents (tweets and webpages).

3.3 Technology requirements

3.3.1 Clustering

The clustering task will demand knowledge of HBase, Apache Spark and Scala. Apart from these technologies, we will also require domain knowledge about clustering algorithms in information retrieval.

3.3.2 Social network

The social network task will use Python for building the network graph. The crawling task is written in Python as well. Python is a great programming language for fast text data processing. Active developer communities create many useful libraries that extend this language for various applications. One of those libraries is 'tweepy'. It is open-sourced and hosted on GitHub. Tweepy provides an easy way for your Python code to get the Twitter information through its APIs.

4. Design and implementation

The design of the components of the existing system along with the new flow is illustrated in Figure 12, which was provided by the research assistants supporting the project.

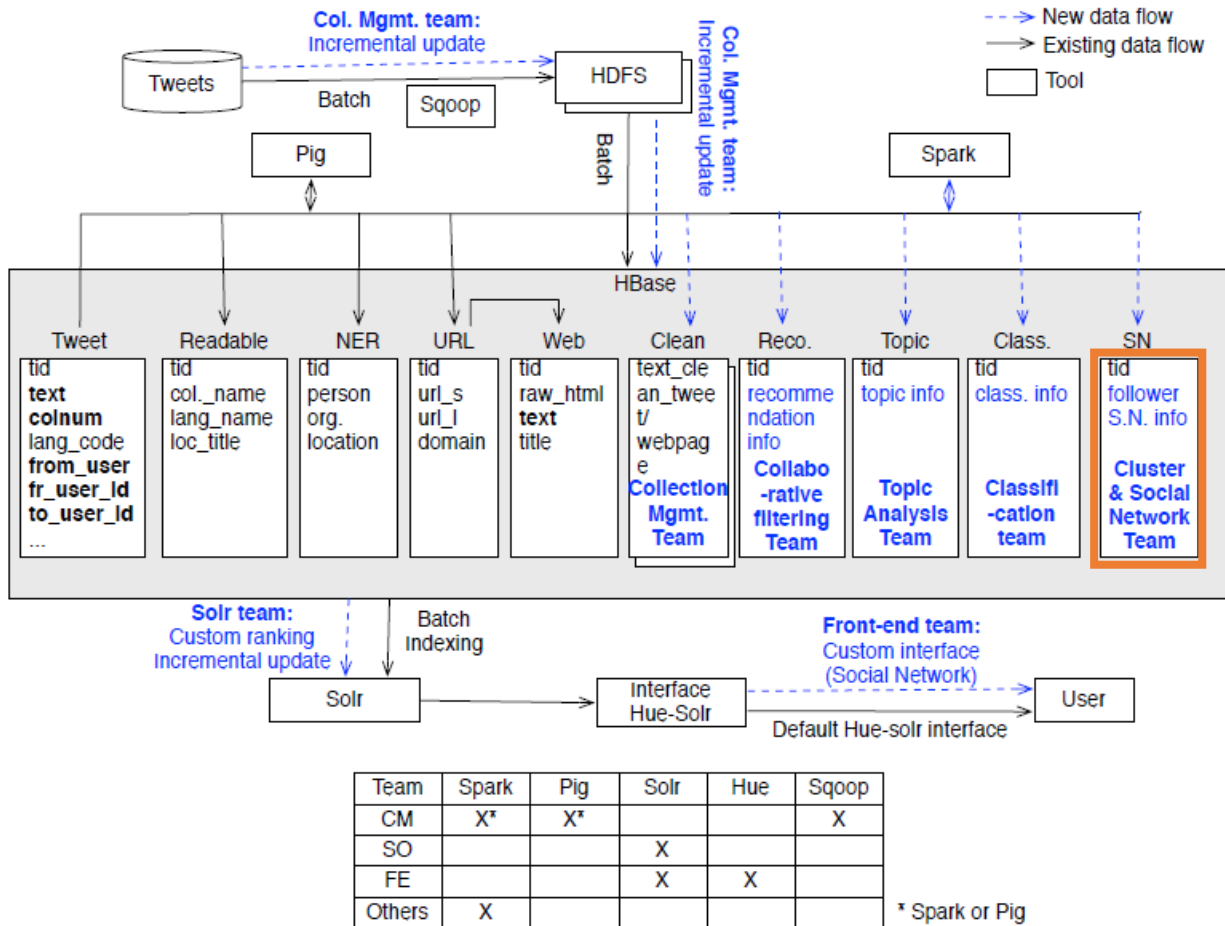


Figure 12: Flow diagram of the system

Before we look into the design of clustering and social networking tasks for our team, we will have a brief overview of where our module is in the overall system. The box highlighted in orange on the right side, is the module on which the team works (clustering and preparing a social network). As mentioned in the 'Requirements' chapter the clustering will use Spark and Scala, whereas social network preparation will use Python. We will have access to HBase for cleaned tweets. We will now see the detailed design, implementation and evaluation of the clustering and social networking tasks of our team.

4.1 Clustering

4.1.1 Design and implementation

We start with flat clustering methods and initially we focus on k-means. We use the MLlib package of Spark for executing our clustering algorithms.

4.1.1.1 Tweet clustering

For tweet clustering based on the tweeted data, we use the cleaned dataset provided by the collection management team. The cleaned dataset is read into a Spark RDD. Each tweet is converted into a key value pair with key being the tweetID and the values being a list of words in the tweet. The list of words is then converted to a numerical vector. The term weights are decided using TF-IDF scores.

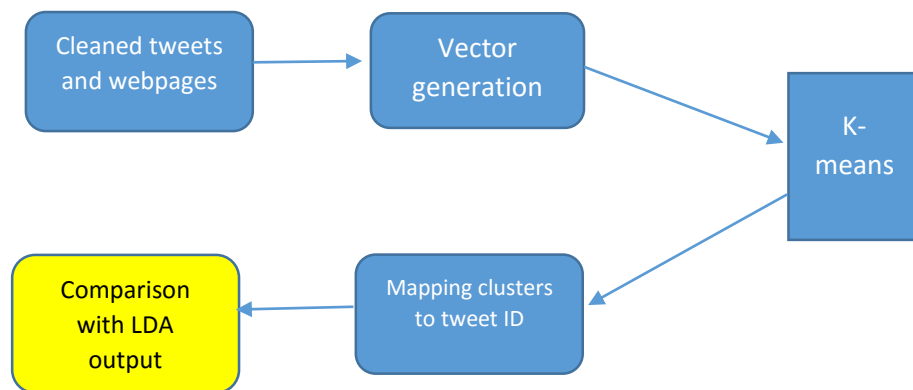


Figure 13: Clustering of tweets based on tweet data

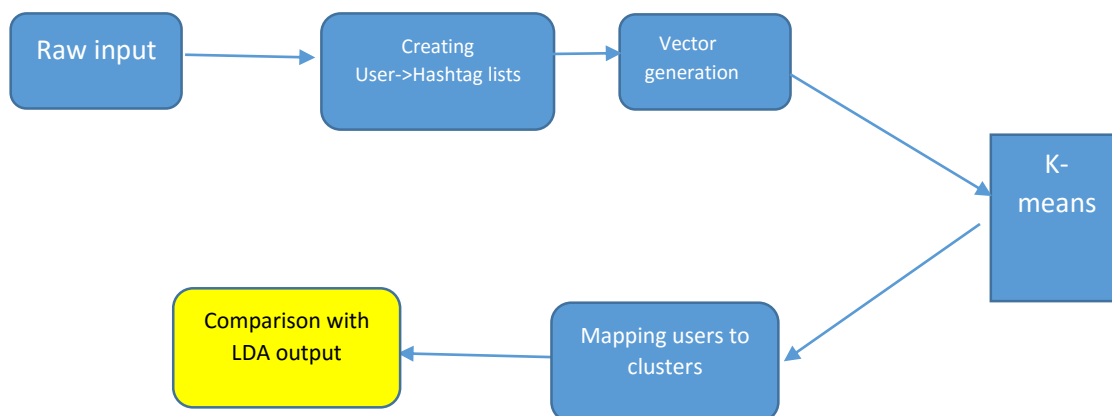


Figure 14: Clustering of users based on hashtags

In the second part, we use the raw input^[a] that we received before we had the cleaned tweet data from the collection management team. We filter the data to extract only the hashtag information and the user name from each tweet. We eventually create a list of key value pairs, where key represents the username and the value represents a list of all the hashtags that user has used in his tweets^[b]. Clustering is performed by converting the lists to vectors using the TF-IDF weighting, and then performing K-means [13].

4.1.1.2 Webpage clustering

We make an initial attempt at clustering the webpage collection where we use the data given by the Collection Management Team. We filter the collection to use only individual webpages without any Twitter information. Table 4 summarizes the results for clustering webpages using the standard TF-IDF model for feature extraction and normalization using the $||L_2||$ norm.

4.1.1.3 Refining feature extraction procedure

One of the deficiencies of the TF-IDF model for vector generation is that it ignores the semantic relation of different words. Each word is treated independently, irrespective of its order in a sentence. To overcome this, we do a few refinements on the work we did till now. First, we normalize the data using the $||L_2||$ norm. We extract feature vectors using a different method called word2vector generation^[c].

This feature extraction procedure is superior because it takes into account the sequence of words in feature vector extraction. The semantic and the syntactic information is maintained during the vector generation. For instance, if $V(\text{"word"})$ represents the vector of "word", then for vectors generated by this model, we have

$$V(\text{"King"}) - V(\text{"Man"}) + V(\text{"Woman"}) \approx V(\text{"Queen"})$$

The word2vec model needs a text collection for training. Here, we treat each tweet as a sentence and combine all of them in a single document for training the model and generating vectors of each word. We create a vector representation of a document by taking a linear combination of all the vectors of the words in the document^[d]. Section 5 summarizes the results and provides comparison of feature vector extraction using TF-IDF and word2vec models. Using word2vec model provides a better and a uniform clustering as compared to the TF-IDF based model.

Notes:

[a] The raw input is a .csv file containing tweet information. Since we had already performed the processing using this dataset, we chose not to change it for this report. We would also be using the cleaned datasets provided by the collection management team for further evaluations.

[b] A user may have multiple tweets. We combine all the hashtags from all tweets of a user. For the moment we ignore the timing information. In the future, a more meaningful clustering can be obtained, if we consider tweets by the same user within a certain time frame only.

[c] word2vec method was invented by Google in 2013 which uses 2-layer neural network to learn patterns of words in text. There are models built using the entire Wikipedia, to give an effective vector representation of each word.

[d] It is ongoing research to find the most effective ways of combining the word vectors into a vector representation for documents. We use a linear combination of word vectors for generating document vectors.

4.2 Creating social network

For our specific task of Twitter social network, the Twitter accounts are used as the social network nodes. We consider three levels of connection between those accounts: 1) following; 2) retweet; and 3) mention (@). The other interactions like comment or 'like' (the action for a tweet) are not considered in our research, because we don't have such data.

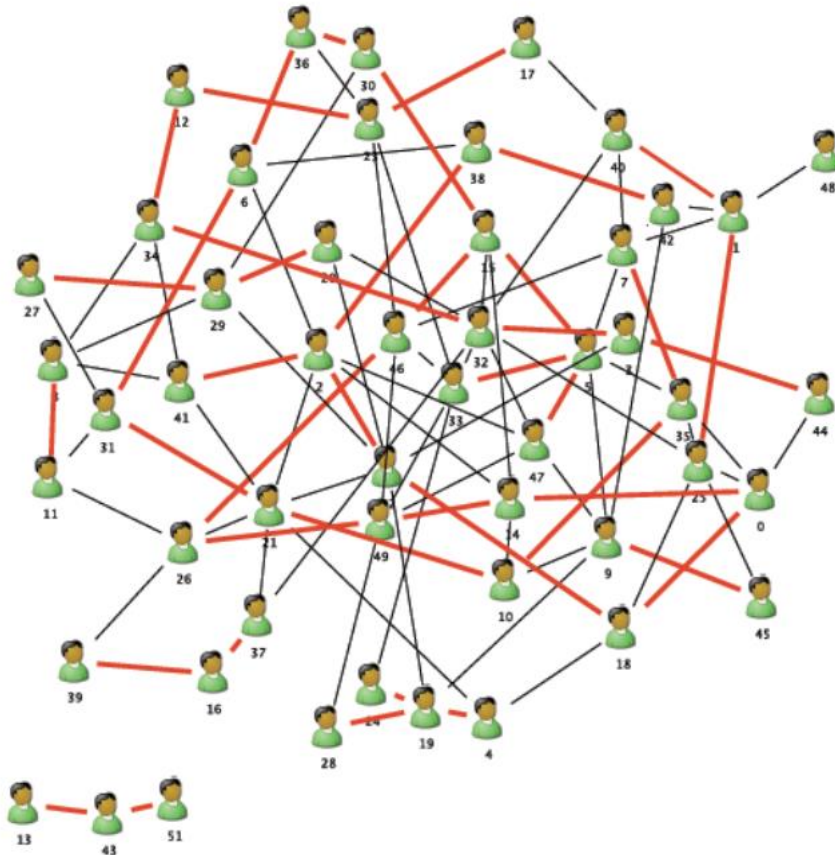


Figure 15: The Twitter social network (Red line is two direction following and black is one direction following) [10]

In those 3 levels of connection, the (1) following is a binary relationship between accounts, and the other two are content related and countable (counts of (2) retweet or (3) mention between accounts). In building the social network, we should combine those three levels by giving them different weights on the analysis.

In Figure 16, we can cluster the social network according to the nodes and edges information. As all the contents on Twitter are generated by those nodes (accounts), the clustering of social network will provide one criteria to cluster the user generated content (e.x., tweets). In the end, the clustering of tweets according to both content and social

network could be used in our IR system to estimate the relevance factor for a given query.
[11]

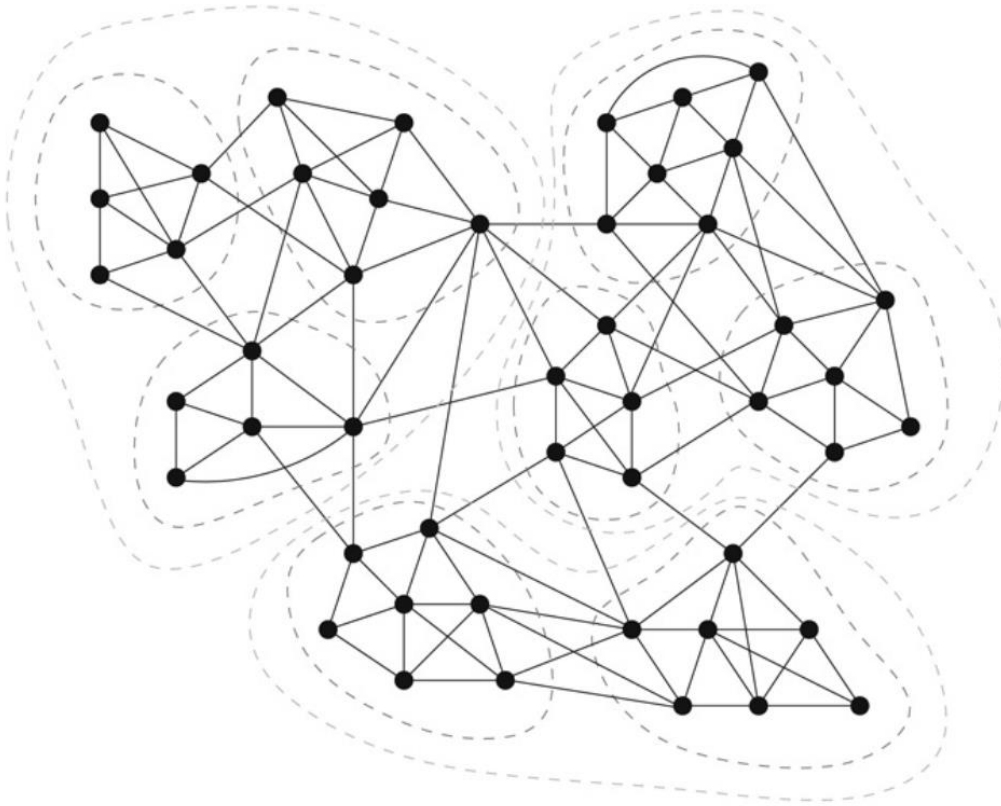


Figure 16: Network and clustering

4.2.1 The data crawler:

Crawler one: user information

Beyond the data from IDEAL project, we crawled additional data from using Python—Tweepy package from Twitter. The code is included in the code package as ‘sncrawler.py’, and Figure 17 shows the header of the code.

```
# -*- coding: utf-8 -*-
"""
Created on Wed Feb 10 20:53:43 2016

This crawler crawls the user information from tweets including:

@author: lijietang
"""

import tweepy
import json
import time
```

Figure 17: Crawler 1: User information crawler

User information crawler will do the following things: 1) get the username / ID from original data; 2) crawl the metadata for this user including: following counts/follower counts; 3) crawl the follower list for each user. According to the Twitter API rule, the crawler sets up two sleep modes: one-minute sleep for large follower list, and, 15 minutes sleep for over the limit. Crawler is designed to start from where it is paused, so it could be kept running until the task is finished.

Crawler two: tweets crawler

Tweets were provided by the original data source. However, it missed the information of account interaction, e.g. favorite, retweet and comment. Those interactions are essentially important in Social Network. Our crawler will go back to the tweets collected, and get the current retweet/favorite counts. This crawler is also included in the code package.

4.2.2 The social network builder:

A social network is a directed network. There is a social network node for each Twitter account, and edges are the interaction between those account. We build the social network using the accounts we have in original data as the nodes and the follow information and retweet information as the edges.

The code is included in the package as 'tweetsSN.py'. The retweet information is gathered using: 1) tweets crawler and 2) the 'RT @' sign in the retweeted tweets.

The product, as showing in Figure 18, is a matrix of Nodes and Edges. Nodes are the account name and the edges are the from/to nodes of retweets. The weight on the edges are the total number of retweets count on this edge.

	A	B		A	B	C
1	user	id	1	Source	Target	Weight
2	ButtFuckNer	0	2	12187	464	2
3	petebachma	1	3	7480	8588	1
4	AhmadNO	2	4	11350	11127	1
5	ColmanMcM	3	5	7087	1831	1
6	grantpearrel	4	6	6	464	1
7	rayeechill	5	7	459	1789	2
8	BlacksburgSt	6	8	6887	6	1
9	acole4vt	7	9	7589	5147	1
10	tmj_roa_hrt	8	10	327	6	1
11	tmj_roa_edu	9	11	2524	3235	1
12	alisonneary	10	12	9468	7281	1
13	HirePhil	11	13	13521	10329	1
14	burntcreative	12	14	10939	9308	1
15	andreabadgl	13	15	10084	9808	1
16	bestjobsonlin	14	16	2899	2898	1
17	BlacksburgW	15	17	7607	11205	1
18	JForeWV	16	18	8776	6931	23
19	IanFerguson	17	19	4729	4730	1
20	internshipjok	18	20	9201	151	1
21	DiamondEve	19	21	10137	862	1
22	Mike_TheBu	20	22	2017	3726	10
23	mthoogs pv	21	23	167	731	1

Figure 18: sample of Social Network Matrix

4.2.3 The social network visualization

With the node and edges information, we can visualize the social network in any platform including: D3, Gephi or nodeXL. We choose Gephi in this work, because: 1) it provide a easy way to cluster the network; 2) the result from Gephi can be exported for D3 visualization, and in this way we can build a better D3 product for the web application. We included a Gephi result in the final submit. And one view from that Gehpi file is shown in figure 19:

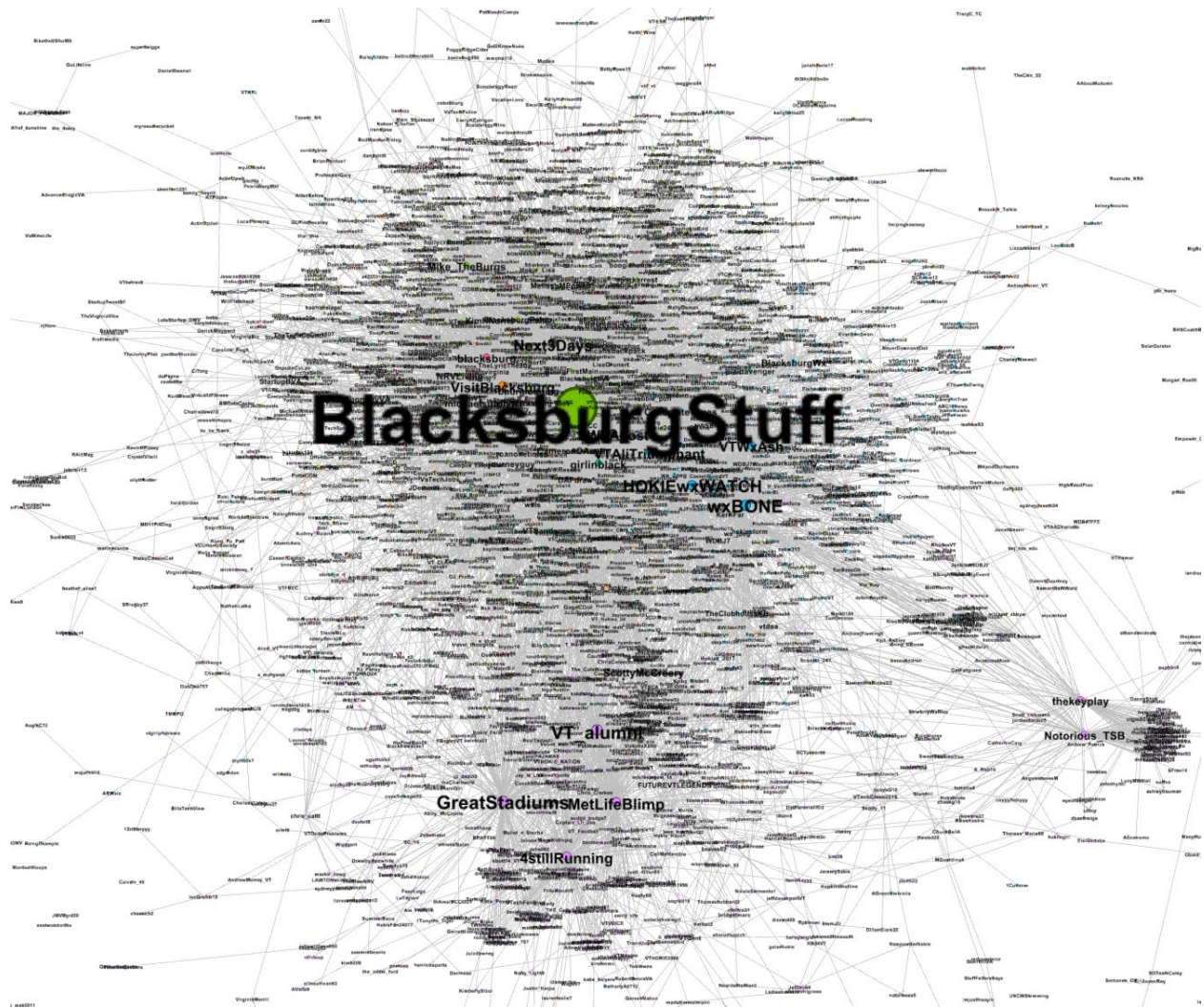


Figure 19: visualization result: entire network

5. Evaluation

In the evaluation phase, we will extend section 3.1.1.2 from chapter 3.

5.1 Within Set Sum of Squared Errors - WSSE

Refining feature extraction procedure

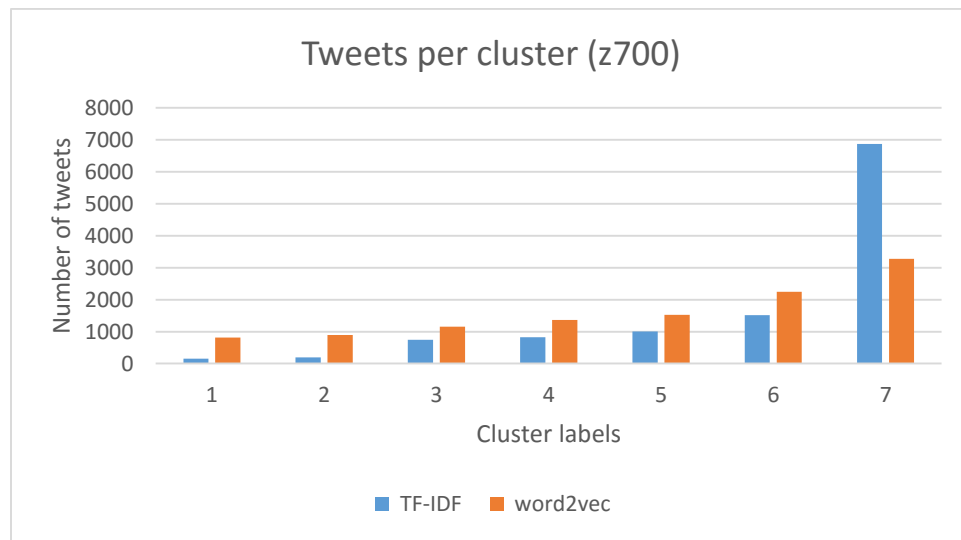
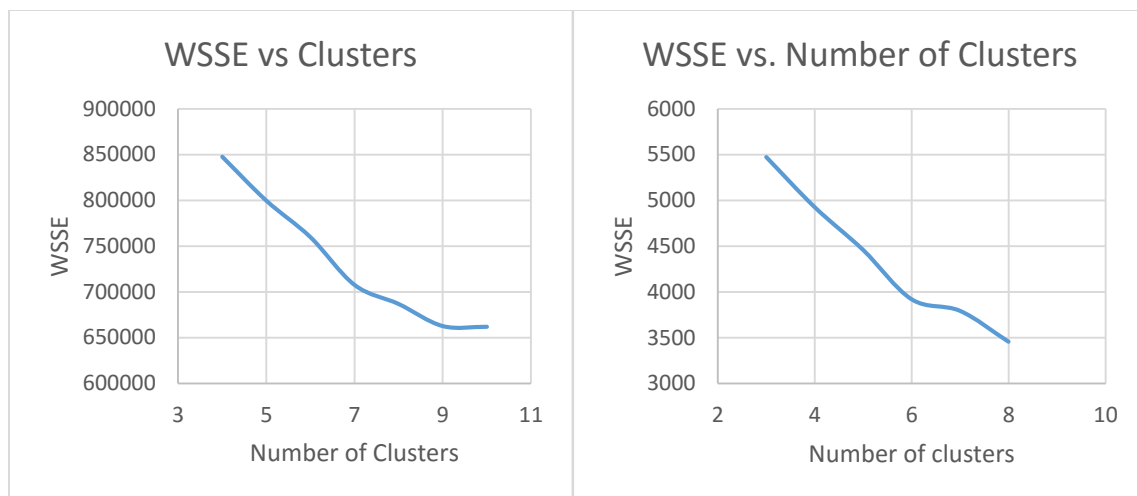


Figure17: Comparison of cluster sizes

Word2vec model provides a better and a more uniform clustering as compared to the TF-IDF based model. We find that the WSSE score is 4637 using word2vec as compared to 7744 for TF-IDF based clustering (with normalization) which is a significant improvement^[c]. Thus we would be using word2vec model for vector generation for rest of our project.

Clustering of tweets

At present we are using within set sum of squared errors (WSSE)^[a] for deciding the optimum number of clusters in the dataset.



A: Using TF-IDF^[d]

B: Using word2vec

Figure 18: Evaluating the cluster number using WSSE for z700 collection

Based on the figure 18 B, 6 is the optimal number of clusters. ^[b] We assume that 6 should be the optimum number of clusters for other collections too, to maintain consistency between collections and also facilitate the Solr team for indexing the collections. Also majority of the clusters have 6 as the optimum cluster numbers

Clustering of users for tweets

Number of Clusters	WSSE for Clustering of Users
3	7.97
4	7.94
5	7.92
6	7.77
7	7.76

Clusters	Users per cluster
1	4
2	146
3	11
4	1
5	34
6	13331

Table 1: Evaluation of users for tweets with WSSE

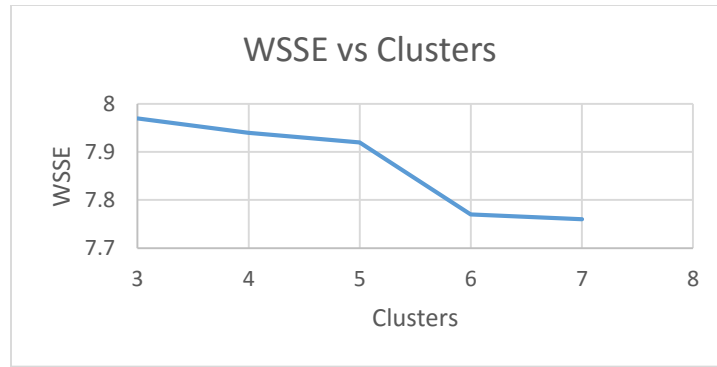


Figure 19: WSSE scores vs number of clusters

The WSSE scores indicate that 6 is the best number of clusters. However, a closer look into the number of members per cluster suggest that most of the users are clustered in one cluster. This is due to several reasons primarily because the dataset used is for a particular topic and a lot of users use common hashtags which results in our clustering algorithm predicting that most of the users are connected.

Notes:

- Individual WSSE scores mean nothing. A high decrease in values w.r.t. increase in number of clusters indicates better clustering.
- Last year's report suggested that 5 was the best clustering size, but their conclusion was based on empirical results and observations. We use a quantitative way of evaluating the optimum number of clusters. The elbow point in the graph indicates optimal number of clusters [11]. After the elbow point the WSSE continues to decrease but at a lower rate.
- We can compare the WSSE scores for the two models since we normalize the data to the same scale.
- The high WSSE scores are because the data was not normalized as this plot was created at the very beginning of the project

Clustering of different collections

Next we present a comparison of clustering on the webpages and tweets of the same collection. Interesting insights can be drawn here. Figure shows z541 collections have similar distribution among tweets as well as webpages. The webpage collection for z602, z668 and z696 collection are dominated by two clusters. Figure shows that the z686 collection for tweets have high noise and that there are 3 main clusters that summarizes the collection.

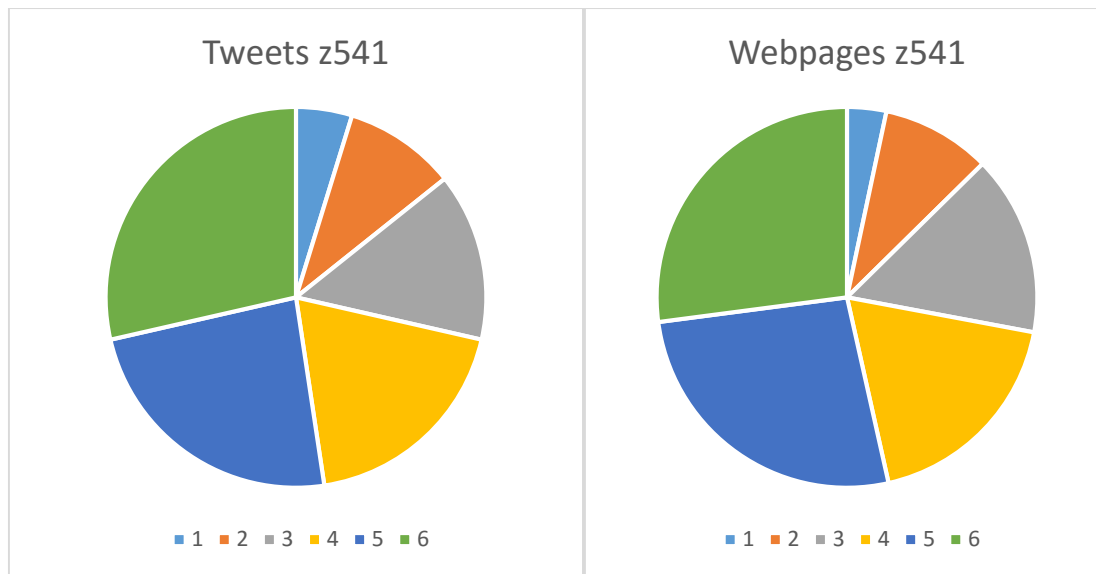


Figure 20: Collection: #NAACPBombing

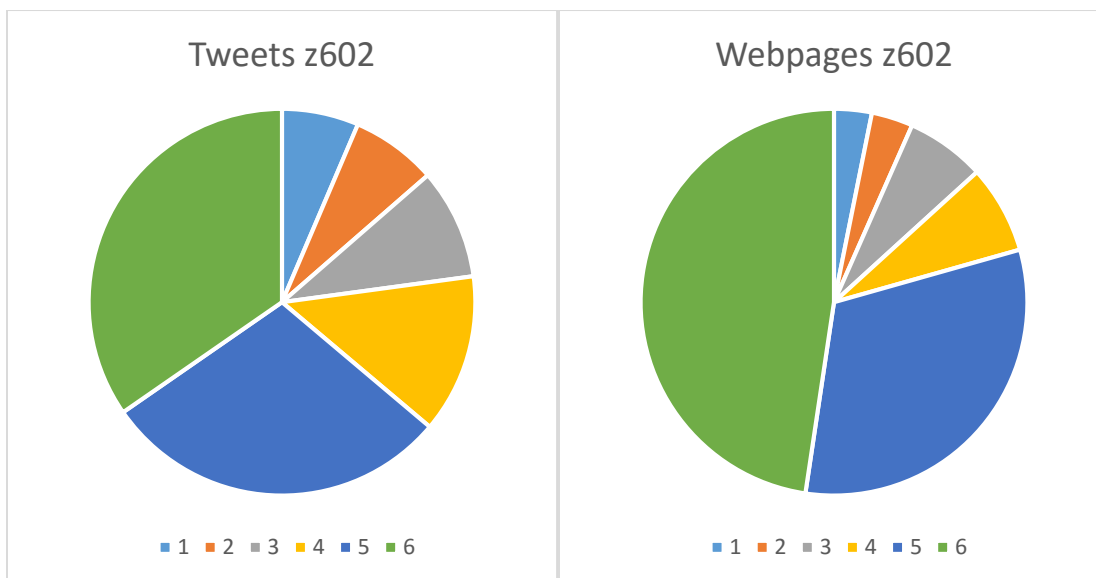


Figure 21: Collection: #GermanWings

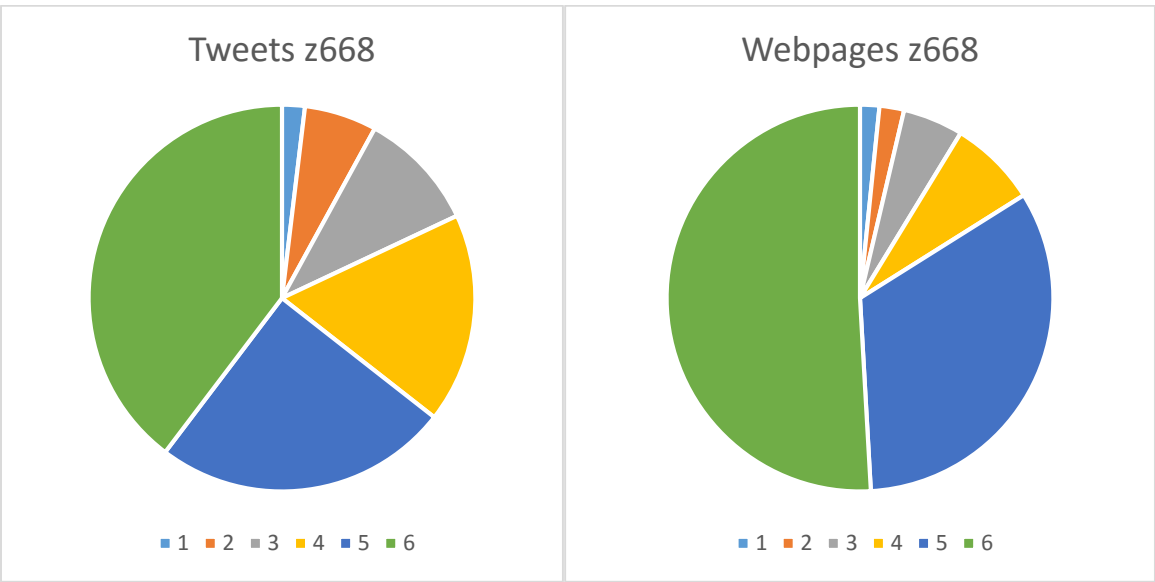


Figure 22: Collection: #houstonflood

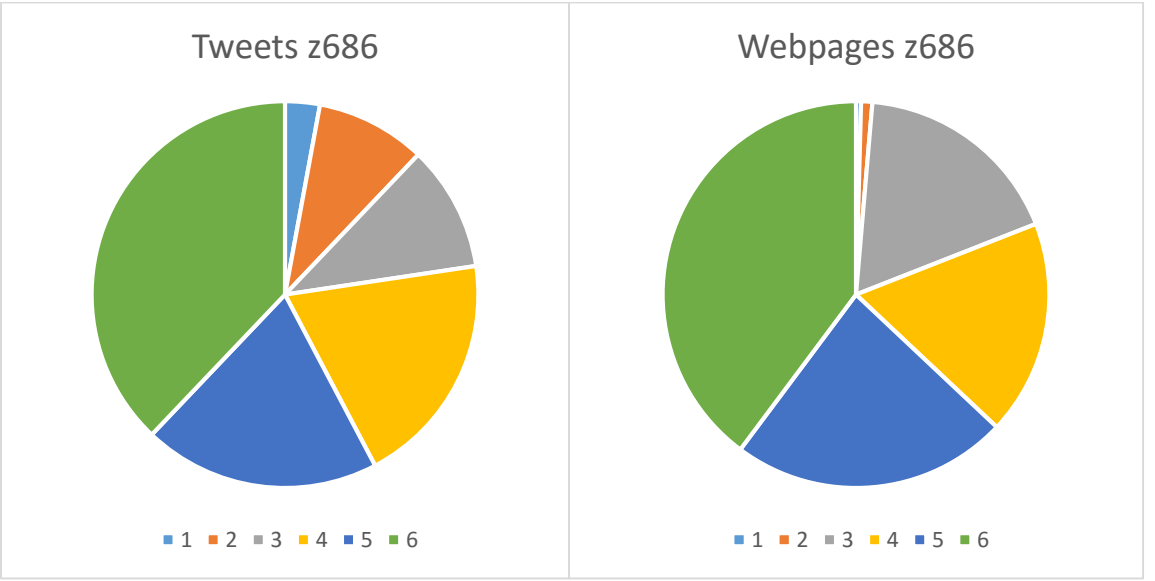


Figure 23: Collection: #Obamacare

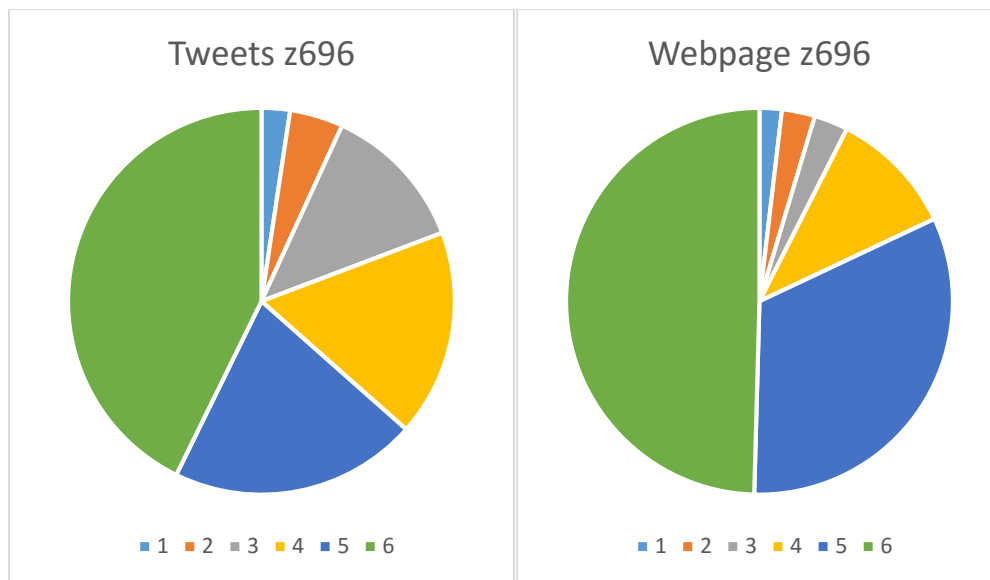


Figure 24: Collection: #4thofJuly

5.2 Analyzing the output of topic analysis team

Our second approach towards evaluation will involve using output of the topic analysis team to check how well the documents have been clustered. We anticipate answers for the following questions -

- Can terms carrying a higher weight across topics help us tune the feature vectors?
- How can we influence the initial seed selection for k-means using the topic association with a document in the cluster?
- Can we evaluate the cluster formation by considering the overall distribution of topics for a document?
- Can we address the problem of cluster labeling at least partially using the output from topic analysis?

The topic analysis team's output represents a document-topic matrix. Clustering output represents a document-cluster matrix. Refer to the Tables 2 and 3.

Document	Topic 1	Topic 2	Topic 3	Topic 5	Topic 6
Doc 1	0.5	0.15	0.02	0.45	0.08
Doc 2	0.45	0.08	0.009	0.18	0.20
Doc 5	0.18	0.20	0.09	0.42	0.17
Doc 7	0.42	0.17	0.18	0.15	0.02
Doc 9	0.15	0.02	0.45	0.08	0.009
Doc 11	0.08	0.009	0.18	0.20	0.09
Doc 12	0.20	0.09	0.42	0.17	0.18

Table 2: Output of topic analysis team, document-topic matrix

Document	Cluster ID
Doc 1	1
Doc 2	2
Doc 5	1
Doc 7	2
Doc 9	2
Doc 11	2
Doc 12	1

Table 3: Output of clustering team

Table 2 gives the topic distributions for each document. For e.g., there is a 50% chance that Doc 1 belongs to topic 1 and so on. Doc 1 represents Topic 1 and Topic 2 better than Topic 3. We understand this based on the probability. Table 3 is the output of our clustering algorithm which tells us which document (tweet) belongs to which cluster. Both these tables show dummy and arbitrary data.

Figure 25 represents the approach we propose for evaluation with topic analysis output. We pick one third number of topics (out of the total number of topics) with highest weightage for every document. We will then group these documents by clusters. We will compute a mean topic frequency matrix over all clusters individually. After this computation, we find the deviation of each topic from the topic frequency matrix across all documents in a cluster. Once this computation is performed for all clusters, we will pick the top 3 topics with least deviation for each cluster.

We pick 2 topic values with highest weightage for every document from Table 2. Then we group the documents by their cluster, with help of information in Table 3. Let us say we get Table 4 after this computation. T<no> represents e.x., Topic 1 = T1

Document	Top Topics	Cluster no
Doc 1	T1=0.5, T5=0.45	1
Doc 2	T1=0.45, T6=0.20	2
Doc 5	T5=0.42, T2=0.20	1
Doc 7	T1=0.42, T3=0.18	2
Doc 9	T3=0.45, T1=0.15	2
Doc 11	T5=0.20, T3=0.18	2
Doc 12	T3=0.42, T1=0.20	1

Table 4: Aggregation of topic analysis data and clustering data

Topic 8 represents the topic frequency matrix which takes the mean of a topic weight in across documents in the cluster and records its frequency of occurrence.

Cluster no	Topic Mean	Topic frequency
1	T1=0.35, T5=0.435, T2=0.20, T3=0.42	T1=2, T5=2, T2=1, T3=1
2	T1=0.34, T6=0.20, T3=0.27, T5=0.20	T1=3, T6=1, T3=3, T3=3, T5=1

Table 5: Mean topic frequency across clusters

We now compute the deviation by using Table 4 and Table 5. Our goal is to find out how relevant and tightly coupled the topic is to the cluster. We will choose the top 2 topics with the highest frequency for that cluster. These are highlighted in blue color in Table 5. Now, we will find the deviation for Topic 1 and Topic 5 in cluster 1. Let us do this for Topic 5. From Table 4, we know that Topic 5 occurs in Doc 1 and Doc 5, having values 0.45 and 0.42 respectively. We calculate deviation by following formula:

Deviation of topic = $\sum |topic\ mean - actual\ weight| / total\ number\ of\ documents\ in\ cluster$

Deviation for T5 in cluster 1 = $(0.435 - 0.45) + (0.435 - 0.42) / 3 = 0.01$

We will similarly find this value for the topics highlighted in blue in Table 8, across all documents by cluster. We will then pick up top k topics with least deviation as the topics that represent the cluster.

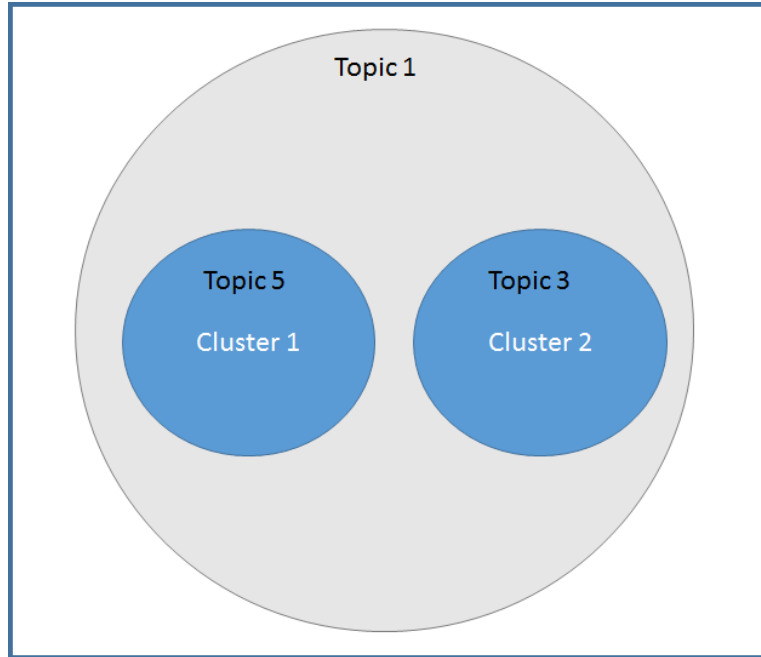


Figure 25: Evaluation of clusters based on topic analysis output

Let us say, for example, cluster 1 resembles Topic 1, Topic 5 and cluster 2 represents Topic 1, Topic 3. We can then further infer; both these clusters have Topic 1 in common. Hence, we see that we can extract local topics specific to each cluster and global topics shared by all clusters, by evaluating the output from the topic analysis team [14] [15]. Figure 25 represents this in a diagrammatic way.

The output of the topic analysis team has another table which comprises of top 10 words for each topic, called as the topic-word matrix. We can further extend the work mentioned above, by extracting the top 5 words for each topic and possibly label the clusters with these words.

In order to summarize, the topic analysis team's output of document-topic matrix will help us evaluate clusters and find local topics specific to each cluster and global topics shared by all clusters. The topic-word matrix helps labeling the clusters.

For number of clusters = 7, and topic analysis team giving us 5 topics for each document for the small dataset z_700, we have the following topic-cluster distribution:

Cluster no	Topics representation	Cluster labels
0	Topic 5, Topic 3	Reporter, watch, youtube, slain, brainstelter
1	Topic 5, Topic 3	Reporter, watch, youtube, slain, brainstelter
2	Topic 2, Topic 3	Suspect, people, victim, youtube, slain, brainstelter
3	Topic 2, Topic 3	Suspect, people, victim, youtube, slain, brainstelter
4	Topic 5, Topic 3	Reporter, watch, youtube, slain, brainstelter
5	Topic 4, Topic 3	Heartbroken, condolence, survivor, youtube, slain, brainstelter
6	Topic 5, Topic 3	Reporter, watch, youtube, slain, brainstelter

Table 6: Evaluation output defining local and global topics and cluster labels from topic words

This evaluation process was run on the following tweet collections - 602, 668, 694, 541, 686.

There are three outcomes of this evaluation process:

- I. Probability of a document belonging to a cluster
- II. Cluster labeling (from topic words as well as heuristic labeling)
- III. Hierarchical mixture of topics and clusters

We will describe each outcome briefly in the below sections.

Probability of a document belonging to a cluster

The probability of each document belonging to a particular cluster is calculated by adding the original probability weight of topics for a document w.r.t to the cluster it belongs. For example, let us calculate the probability for Doc 1, that it belongs to cluster 1. We will add probabilities for the values for Topic 5 and Topic 3 (Refer Table 9). These values will be picked up from Table 5.

Cluster labeling

Cluster labels can be extracted from topic words as well as heuristic methods to label clusters. The extraction of labels from topic words is shown in Table 6. Sample evaluation results for heuristic cluster labeling is shown below.

Cluster no (wordToVec)	Topics	Topic words	Cluster labels
0	Topic 3, Topic 4	crash,germanwings,pilot,victims,plane,Lufthansa germanwings,lubitz,copilot,flight,playing	Germanwings crash and pilot Lubitz
1	Topic 1, Topic 5	germanwings,ripley,believe,click,bigareveal germanwings,world,charliehebdolives,garissa	Germanwings news
2	Topic 4, Topic 5	germanwings,lubitz,copilot,flight,playing germanwings,world,charliehebdolives,garissa	Germanwings crash news
3	Topic 1, Topic 4	germanwings,ripley,believe,click,bigareveal germanwings,lubitz,copilot,flight,playing	Pilot Lubitz of Germanwings
4	Topic 1, Topic 2	germanwings,ripley,believe,click,bigareveal germanwings,copiloto,vctimas,accidente,vuelo	Happenings and accidents
5	Topic 1, Topic 4	germanwings,ripley,believe,click,bigareveal germanwings,lubitz,copilot,flight,playing	Pilot Lubitz of Germanwings

Table 7: Heuristic cluster labeling after extracting top 2 topics for clusters in collection tweets 602

Hierarchical mixture of topics and clusters

We have come up with 2 representations of hierarchy, each highlighting different aspects. The output helps us observe a hierarchical relationship between clusters and also helps us understand how popular certain topics are in the cluster and how they are clubbed together. Observation in Figure X helps us understand that Topic 1 and Topic 4 have been very popular in the dataset. Observation of Figure X helps us understand that we have a hierarchy within topics and clusters.

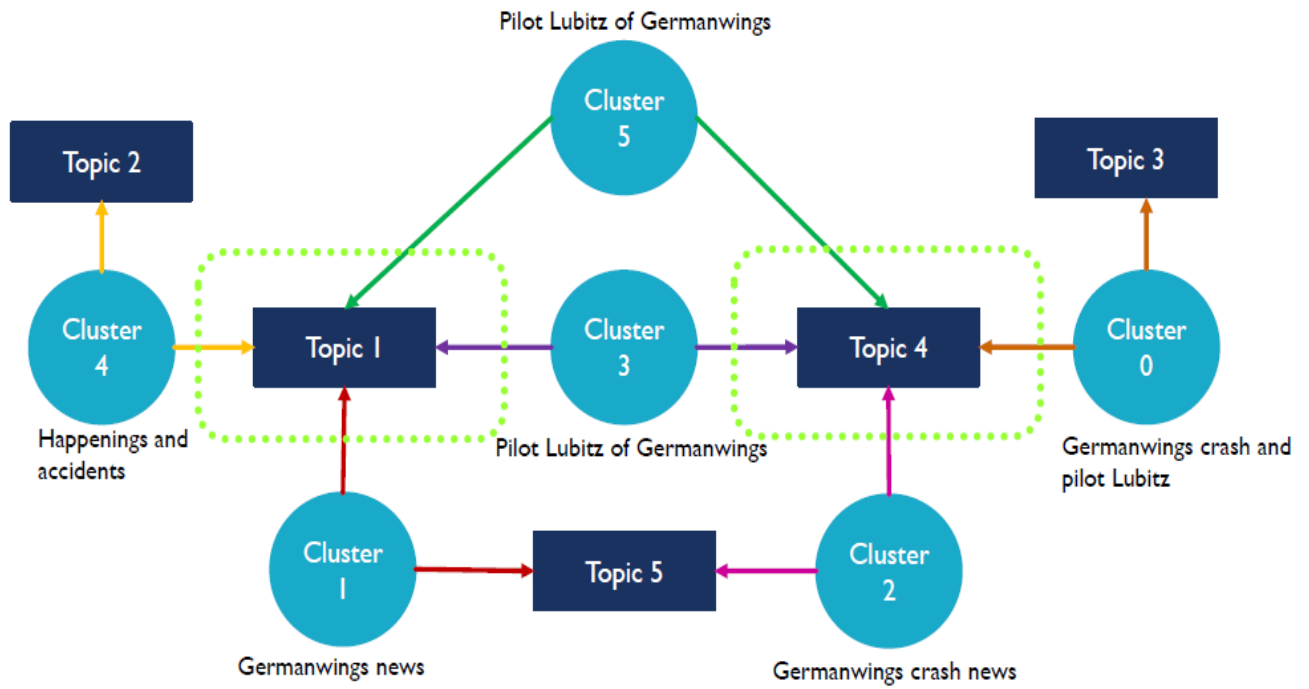


Figure 26: Hierarchical representation 1 of evaluation output

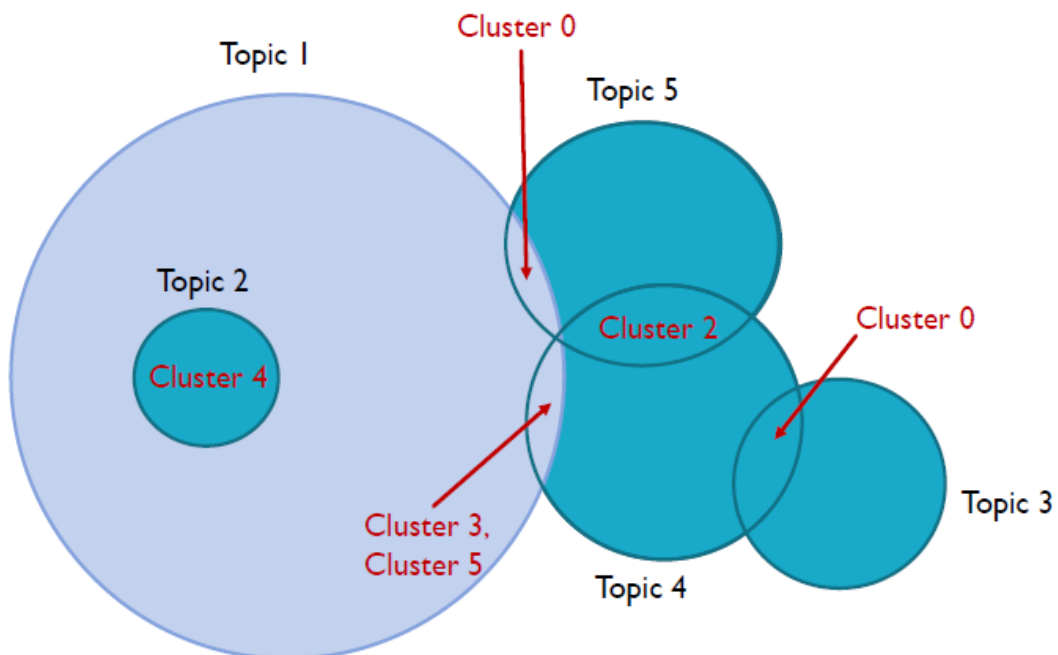


Figure 27: Hierarchical representation 2 of evaluation output

6. Milestones and work progress

6.1 Clustering

No.	Tasks for clustering	Week/Date	Done by
1.	Reading of previous year's reports, finalizing goals for the project	Week 1	Saket, Swapna
2.	Setting up virtual machine in laptops, getting familiar with the development environment	Week 2	Saket, Swapna
3.	Co-ordination between clustering and social network tasks	Will be done throughout the semester	Swapna
4.	Literature Survey	Week 3, Week 5	Swapna
5.	Formalizing deliverables, interim report 1	Week 4	Saket, Swapna
6.	Exploration of clustering algorithms	Week 3, Week 4	Saket, Swapna
7.	Feature extraction from raw input for creating user-hashtag lists	Week 3, Week 4	Saket, Swapna
8.	Vector generation and clustering of users based on hashtags using raw data	Week 4, Week 5	Saket
9.	Vector generation and clustering of tweets using data provided by collection management team	Week 6	Saket
10.	Compilation of the entire interim report 2, incorporating the suggestions from previous reports as well as inputs from social networking component	Week 5, Week 6	Swapna
11.	Report 2 - Design, implementation for clustering. WSSE evaluation for clustering. Developer's manual	Week 5 Week 6	Saket
12.	Clustering of webpages, improving feature extraction.	Week 8-9	Saket
13.	Extraction of cluster labels	Week 10-12	Swapna
14.	Evaluation of clusters Collaborating with topic analysis group to verify and evaluate results in more detail.	Week 10-12	Swapna
15.	Compilation of the entire interim report 3, incorporating the suggestions from previous reports as well as inputs from social networking component	Week 10	Swapna

16.	Report 3 – Design, implementation for clustering WSSE evaluation for clustering webpages	Week 10	Saket
17.	Implementing the clustering algorithms on all datasets for evaluating tweets webpages	Week 13-14	Saket
18.	Compilation of entire interim report 4, incorporating suggestions	Week 12	Swapna
19.	Further exploration for cluster evaluation – assigning probabilities and cluster labeling	Week 13-14	Swapna
20.	Compiling report 4, as per required guidelines, updating various sections, manuals, etc.	Week 14	Swapna
21.	Uploading entire project work in VTechWorks	3 May	Saket
22.	Interim report 3	31 Mar	Done
23.	Interim report 4	20 Apr	Done
24.	Final project presentation and report	3 May	Done

*Week 1 starts from 24 Jan. 2016

Table 8: Clustering task breakup

Note: The social networking component is undertaken independently by Lijie Tang.

6.2 Social network

Following is the task list for building a social network graph:

- Crawl information from Twitter (tweepy package in Python)
- Build re-tweets and mention edges from current tweets data
- Combine above information to create a social network
- Calculate an importance factor for each node in the network.

Work done until now:

- The web crawler on tweets in order to collect the account information
- The social network builder using Python is in place
- Basic visualization is ready for the social network in D3. (Note: This task was done before the front end visualization team existed)

The following diagram gives a comprehensive review of the social networking task:

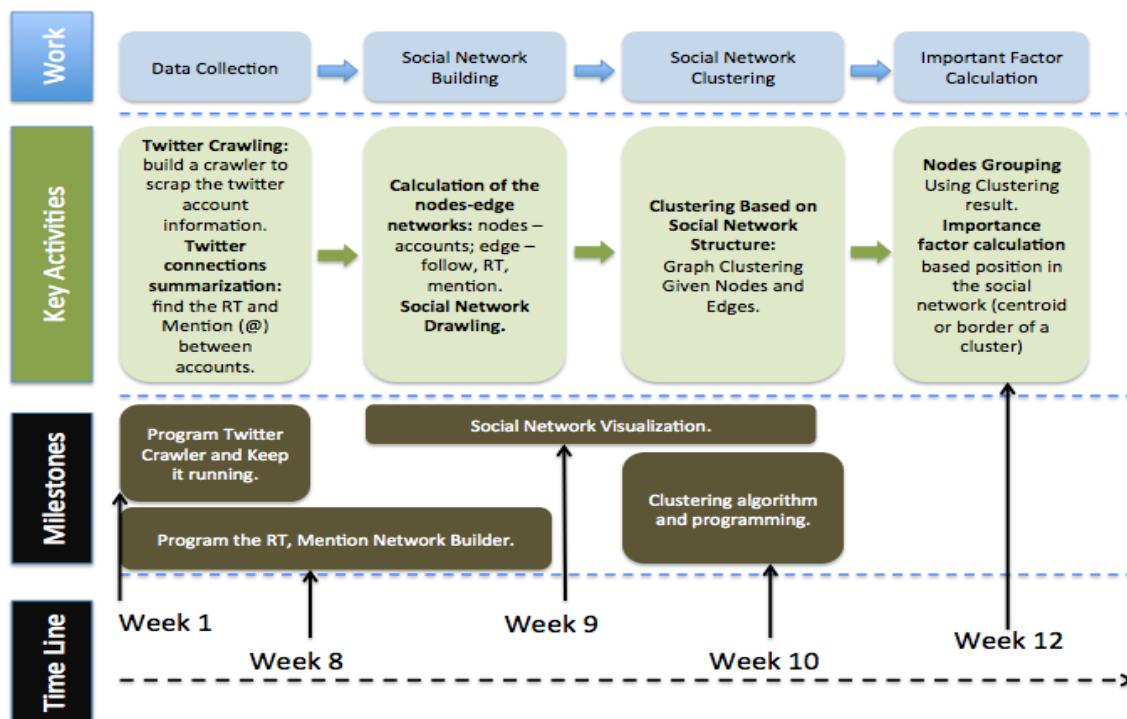


Figure 28: Comprehensive overview of social network task timelines and functional requirements.

7. Developer manual

7.1 Using Scala and SBT as build tool for Eclipse on CDH 5.3

To aid development on local machine and quick debugging, it's better to setup Scala in Eclipse.

- 1) Install Scala IDE for Eclipse Juno

Go to Help->Install New Software

Paste the link for Eclipse Juno and appropriate Scala version from [here](#)

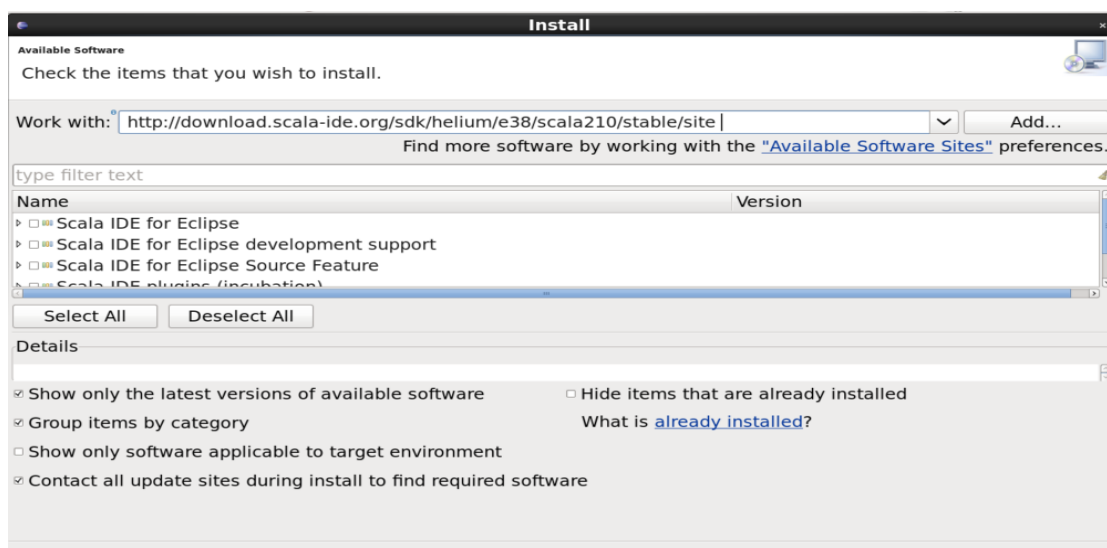


Figure 29: Scala IDE for Eclipse Juno

Select the required packages and click on next tabs to finish installation

- 2) Install SBT for Centos. The instructions are [here](#)
- 3) Install Eclipse plugin for SBT. For this make a new directory at `~/.sbt/0.13/plugins` and create a new file `plugins.sbt`. Add the following line to this file:

```
addSbtPlugin("com.typesafe.sbteclipse" % "sbteclipse-plugin" % "4.0.0")
```

- 4) In the project folder run `sbt eclipse`.
- 5) Make sure that the build.sbt is present and all the dependencies are included in it.
- 6) In Eclipse, import this project. Make sure that Eclipse is running as root, else there would be dependency issues.
- 7) In *Project Properties*, go to *Scala Compiler* and uncheck *Use project settings*
- 8) Run the Scala program, Spark would start inside the Eclipse console. All the auto-correct and suggestion features of Eclipse would be available.

7.2 Feature Extraction and Normalization

We use two different techniques for feature extraction. First is the TF-IDF based technique and the second is the word2vec technique. Spark offers standard libraries for both the methods and comprehensive documentation can be found for both of them [3]. The word2vec models provides better results in terms of feature extraction. These models are shallow, two-layer neural networks, that are trained to reconstruct linguistic contexts of words: the network is shown a word, and must guess which words occurred in adjacent positions in an input text.

7.2.1 TF-IDF weighting

Scala provides us the necessary API for computing the TF-IDF scores for a document collection.

$$IDF(t, D) = \log \frac{|D| + 1}{DF(t, D) + 1}$$

$$TFIDF(t, d, D) = TF(t, d) \cdot IDF(t, D).$$

Here $DF(t, D)$ is the number of documents in which term t occurs in the collection D

```
/******TF-IDF and normalisation******/
```

```
val hashingTF:RDD[Vector] = new HashingTF().transform(values)
val idf= new IDF(minDocFreq=10).fit(hashingTF)
val tfidf:RDD[Vector]= idf.transform(hashingTF.map(_.eachDoc()))

val normalizer = new Normalizer()
val data1= tfidf.map(x=>(normalizer.transform(x)))
```

Figure 30: Vector generation using TF-IDF

Here, the RDD “values” consists of the documents collection in form of $RDD[Seq[String]]$. The *HashingTF* object creates the necessary vector for TF values. We then use the *IDF* object to create the IDF scores. Note that we set the parameter *minDocFreq* =10. This ensures terms that occur in very low number of documents are filtered out. Terms that have very high occurrence are given low IDF scores by definition. Thus we do the necessary filtering here.

The *normalizer* object normalizes the generated vectors using $||L_2||$ norm. This helps in getting better clustering results as k-means provides better results for normalized data

7.2.2 Word2vec

TF-IDF ignores the semantic and the syntactic relation of the words. It simply treats the documents as a bag of words model and computes its scores. To improve feature extraction, we use the word2vec model which uses a neural network to learn bigram relations between words.

```
/******Train word2vec model******/  
val vec = new Word2Vec()  
val model = vec.fit(corpus)
```

Figure 31: Training word2vec model

Corpus can be any collection from which we want to train our word vectors. Using the generated model, we create a vector for each word in a document and sum up all the vectors to get a document vector.

```
val outtest:RDD[Seq[Vector]]= values.map(x=>x.map(m=>try {  
    model.transform(m)  
} catch {  
    case e: Exception => Vectors.zeros(100)//return Vectors.zeros(100)  
}))  
  
val converttest = outtest.map(m=>m.map(x=>(x.toArray)))  
val withkey = keys.zip(converttest)  
val filterkey = withkey.filter(!_._2.isEmpty)  
val keysfinal= filterkey.map(x=>x._1) //this is done to maintain integrity of the RDD  
val valfinal= filterkey.map(x=>x._2)  
val reducetest = valfinal.map(x=>x.reduce((a,b)=>a.zip(b).map(t=>t._1+t._2)))  
val filtertest = reducetest.map(x=>x.map(m=>(m,x.length)).map(m=>m._1/m._2))  
val test = filtertest.map(x=>new DenseVector(x).asInstanceOf[Vector])
```

Figure 32: Generating document vectors

7.3 Clustering using MLlib

MLlib is a Spark library for machine learning methods. It provides different flat clustering algorithms like k-means, streaming k-means, bisecting k-means, etc. However, support for hierarchical clustering algorithms is not available. We use k-means as our initial clustering algorithm. MLlib provides a great API for using k-means which simplifies the task. (source: <http://spark.apache.org/docs/latest/mllib-clustering.html>)

```
import org.apache.spark.mllib.clustering.{KMeans, KMeansModel}
import org.apache.spark.mllib.linalg.Vectors
```

```
val clusters = KMeans.train(parsedData, numClusters, numIterations)
```

Here *k-means* object is used which returns a *k-means* model. Various evaluation methods are available for this model, e.g., WSSE

```
val wssse = clusters.computeCost(parsedData)
```

The k-means object expects the data as RDD[Vector], the number of clusters, number of iterations as its arguments. Our steps for clustering are shown in Figure. We run out clustering algorithm for 10 iterations and then compute the cluster sizes as well as the WSSE scores.

An improvement over the k-means implementation in Mahout is that k-means in MLlib offers support to use algorithms for selection of initial cluster centres like k-means++ and k-means|| rather than randomly picking up the initial centres. Selection of initial cluster centers in k-means has a lot of effect on the eventual clustering. Thus we have better clustering with the use of k-means++ and k-means||.

```
val clusters = KMeans.train(data1, numClusters, 10)
val predictions = clusters.predict(data1)
val clustercount = predictions.map(s => (s, 1)).reduceByKey(_+_ )
val result = keysfinal.zip(predictions)
val wsse = clusters.computeCost(data1)
```

Figure 33: Main clustering steps

7.4 Evaluation using LDA (topic analysis) and clustering output

Software/Tools requirement: Java 1.7 and above

Code structure and description:

Before deploying and running the evaluation code, it is recommended to read the 'readme' file shared with the code base. The instructions and details will be mentioned in this manual as well. The code is a simple standalone java program (1.7 and above), using a parsing library called 'opencsv' for parsing input files for evaluation.

The project consists of 3 files:

Files	Description
src/edu/vt/ir/Main.java	Contains the core logic for deriving results
src/edu/vt/ir/objects/DocumentPoints.java	POJO which contains metadata for a document
src/edu/vt/ir/objects/TopicPoint.java	POJO containing measures for a topic and its metadata

Table 9: Files for evaluation

The project has following important folders that will be needed to run the evaluation:

Folders	Description
clusterOutput	Collection wise clustering output files. (Tweets: 700, 541, 694, 668, 602, 686 Webpages: 686)
ldaOutput	Collection wise LDA output files from topic analysis team. (Tweets: 700, 541, 694, 668, 602, 686 Webpages: 686)
firstRunOutput	Preliminary stage output for a collection, showing the topics to which the cluster correlates.
finalOutput	Final detailed output for a collection. These are tab separated files which are uploaded into the HBase schema.

Table 10: Folder structure for evaluation task

Samples of files in each folder from Table 10, for tweet collection 541 are shown below:

```

(541-552940477469970432, 2)
(541-552940478292049920, 2)
(541-552940484339834880, 3)
(541-552940484574736384, 2)
(541-552940486789697536, 2)
(541-552940488068968449, 2)
(541-552940489826373634, 0)
(541-552940490350661632, 2)
(541-552940490942078976, 2)
(541-552940491810308096, 0)
(541-552940491898384385, 3)
(541-552940500861616133, 2)
(541-552940507991531520, 2)

```

Figure 34: Clustering output for tweets-541

```

541-554539937958473728 naacp,michellemalkin,bombing,naacpbombing,media
0.8676785,0.062945314,0.031863865,0.023184044,0.014328244
541-563233442352336897 michellemalkin,bombing,naacpbombing,media,naacp
0.84447104,0.059827946,0.042748537,0.03249106,0.020461395
541-555164614192033793 naacp,michellemalkin,bombing,media,naacpbombing
0.28545412,0.22317381,0.21378015,0.13973966,0.13785224
541-568926662626299904 bombing,michellemalkin,naacpbombing,media,naacp
0.9024262,0.039140016,0.036312073,0.012420325,0.009701429
541-565491025398231040 michellemalkin,bombing,media,naacpbombing,naacp
0.6455951,0.1792411,0.13786434,0.025684815,0.011614666
541-553596859512791040 naacpbombing,bombing,naacp,michellemalkin,media
0.79115105,0.08607191,0.083971344,0.029752972,0.0090526985

```

Figure 35: LDA output for tweets-541

```

naacpbombing      541
naacpbombing,suspect,sketch,msnbc,paris,
0.13029103464568,0.03049340723161587,0.025262233841054034,0.01843253118501338
,0.017804625106426723,0.016640413003371598

media      541
media,black,naacpbombing,coverage,zellieimani
0.08185242732490461,0.05897674249060026,0.05382613399917965,0.035863113248256
95,0.030825512075630706

```

Figure 36: LDA output for tweets-541, with top 5 topic words and their probabilities

```

cluster no: 0
Topics: michellemalkin media
cluster no: 1
Topics: naacp bombing
cluster no: 2
Topics: naacpbombing michellemalkin
cluster no: 3
Topics: bombing media
cluster no: 4
Topics: naacp naacpbombing
cluster no: 5
Topics: michellemalkin bombing

```

Figure 37: Intermediate evaluation output for tweets-541 mentioning top 2 topics resembling the cluster

```

541-554782118656491520 0 Naacpbombing news in media 0.9502571
541-554115795975168000 0 Naacpbombing news in media 0.67650213
541-553440263805607936 0 Naacpbombing news in media 0.77383299587
541-568851596236304385 1 NAACP stories 0.921410103
541-568895168985890816 1 NAACP stories 0.921346928
541-553102937723895808 1 NAACP stories 0.55167194

```

Figure 38: Final evaluation output for tweets-541

The Table 11, follows the output format in Figure 38 that is uploaded into HBase schema.

HBase schema design:

Document ID	Cluster no.	Cluster label	Probability
String (Tweet-id/URL)	Integer	String	Float

Table 11: HBase schema design

The evaluation process consists of the following steps:

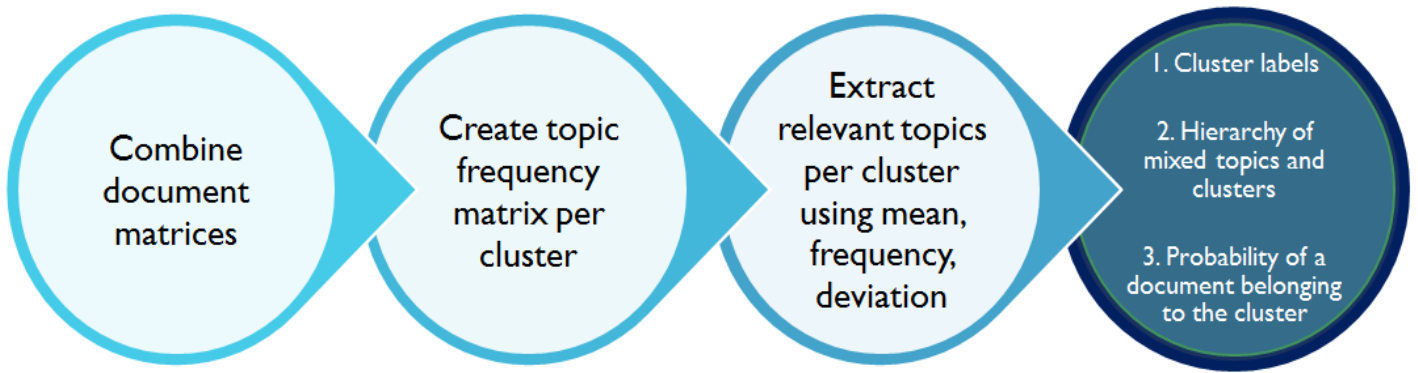


Figure 39: Conceptual workflow of evaluation

Executing the code:

Code has been tweaked multiple times, for matching the output format of both clustering and LDA. Major changes have been commented. Debugging is made easier by putting in log statements, which are commented otherwise.

The 'Main.java' is the main file which does the execution of evaluation. The run is made per collection. (tweets and webpages collections separately)

3 inputs are required for this file to run successfully: -

- I. Cluster output file location
- II. LDA output file location
- III. Output file location
- IV. Collection number

The following snapshot shows the changes to be made in main.java

```

35 // Runs are made per collection
36 static String collectionNumber = "541";
37 static String partialOutput = "firstRunOutput/partial-tweet-evaluation-"+collectionNumber+".txt";
38 static String finalOutput = "finalOutput/TweetSchemaOutput"+collectionNumber+".txt";
39
40 static String clusterOutputFile = "clusterOutput/full-"+collectionNumber;
41 static String ldaOutputFile = "ldaOutput/doctopic_"+collectionNumber+".txt";
42

```

Figure 40: Input parameters

Input files and output files for collections which were shared for evaluation are present in the code at locations specified in Table 10

Clustering output is based on word2vec feature extraction. For collection 668 tweets, we tried it with tf-idf feature extraction as well. The evaluation results show same grouping, hence that gives good confidence about the results.

Two output files will be created.

- a. partial-tweet-evaluation-<collection_number>.txt
- b. TweetSchemaOutput<collection_number>.txt

Cluster labeling is done with heuristics, however, code has a section where labels are generated using LDA topic words as well.

Run the main.java file in steps as described below.

For a new collection, run the main.java once only till step 4. Then with the top topic words generate a label from LDA topic words or using heuristics. Add these labels in the code. Now, run the entire code from step 1 to step 6 again.

These are the 6 steps to execute the code:

- I. Read clustering output
- II. Combine it with LDA output by clusters
- III. Create frequency matrix
- IV. Find relevant topics for documents in a cluster
- V. Get labels from cluster by topic from the table OR heuristics
- VI. Calculate probabilities for documents

These steps have been fleshed out from Figure 39. Following snapshots (Figures 41-45) will show the code snippets of the important calculations pertaining to the 6 steps.

```
51 // 1. read cluster output data
52 CSVReader clusterReader = new CSVReader(new FileReader(clusterOutputFile), ',');
53
54 while((record = clusterReader.readNext()) != null) {
55
56     Integer cno = Integer.parseInt(record[1].substring(0, record[1].length()-1));
57
58     // if cluster not created
59     if(!clusterVsDocs.containsKey(cno)) {
60         Set<DocumentPoints> docPts = new HashSet<DocumentPoints>();
61         clusterVsDocs.put(cno, docPts);
62     }
63
64     // else add entry to the cluster
65     String val = (record[0].substring(1)).split("-")[1];
66     Long docId = Long.parseLong(val);
67
68     DocumentPoints dp = new DocumentPoints();
69     dp.setClusterNum(cno);
70     dp.setDocId(docId);
71
72     clusterVsDocs.get(cno).add(dp);
73
74 }
75 clusterReader.close();
76
```

Figure 41: Reading clustering output

```

89 // 2. read lda output and combine it with clustering output data.
90 // Aggregate data using cluster as key
91 CSVReader reader = new CSVReader(new FileReader(ldaOutputFile), '\t');
92 while((record = reader.readNext()) != null) {
93
94     Long docId = Long.parseLong(record[0].split("-")[1]);
95
96     String[] topicNames = record[1].split(",");
97     String[] topicProb = record[2].split(",");
98
99     // Decide the number of topics to pick up
100     Map<String, Double> topicVsScoreUnsorted = new HashMap<String, Double>();
101     for(int i = 0; i < topicNames.length; i++) {
102         topicVsScoreUnsorted.put(topicNames[i], Double.parseDouble(topicProb[i]));
103     }
104
105     // In case of top k topics selection, sort it
106     //Map<String, Double> topicVsScore = sortMapByValueComparator(topicVsScoreUnsorted);
107
108     findRelevantDocument(docId, topicVsScoreUnsorted, clusterVsDocs);
109 }
110 reader.close();
111

```

Figure 42: Read LDA output

```

311 Integer cluserNum = dp.getClusterNum();
312
313 if(topicFrequency.containsKey(cluserNum) == false) {
314     Set<TopicPoint> topicData = new HashSet<TopicPoint>();
315     topicFrequency.put(dp.getClusterNum(), topicData);
316 }
317
318 Set<TopicPoint> currentTopics = topicFrequency.get(cluserNum);
319 for (Map.Entry<String, Double> topicScore : dp.getTopicVsScore().entrySet()) {
320
321     TopicPoint tp = new TopicPoint();
322     tp = fetchTopicPoint(topicScore.getKey(), topicScore.getValue(), currentTopics);
323     topicFrequency.get(cluserNum).add(tp);
324 }

```

Figure 43: Calculating topic frequency per cluster

```

264     double meanVal;
265     for(TopicPoint tp : topicValues) {
266         meanVal = 0.0;
267         for(DocumentPoints dp : clusterDocs) {
268
269             if(dp.getTopicVsScore().containsKey(tp.getName())) {
270                 // find difference of each of those topic mean from every document in a single cluster
271                 meanVal = meanVal + Math.abs((tp.getMean() - dp.getTopicVsScore().get(tp.getName())));
272             }
273         }
274
275         // take average of this difference
276         double clusterTopicDeviation = meanVal/clusterDocs.size();
277         tp.setDeviation(clusterTopicDeviation);
278     }
279

```

Figure 44: Finding mean and deviation of topics in cluster

```

200     Set<DocumentPoints> clusterDocs = clusterVsDocs.get(cno);
201
202     for(DocumentPoints dp : clusterDocs) {
203         String content = "";
204
205         // write each row to a file (docId      clusterNo      label      probability)
206         content = collectionNumber+"-" + dp.getDocId() + "\t" + cno + "\t" +clusterVsLabels.get(cno) + "\t";
207
208         // calculate respective score for each document by adding the scores of those topics
209         double docProbability = 0.0;
210         Map<String, Double> topicScoreDoc = dp.getTopicVsScore();
211         for(String topic : clusterTopics) {
212             if(topicScoreDoc.containsKey(topic)) {
213                 docProbability = docProbability + topicScoreDoc.get(topic);
214             }
215         }
216
217         content = content + docProbability + "\n";
218
219         bw.write(content);

```

Figure 45: Calculate probability score for each document

Loading final output in HBase:

Following are the steps to load the data in the HBase schema:

- a. The file should be in a tab separated format. The sequence of entries in the file should be as mentioned in the schema diagram, Table 11.
- b. Assume your local collection data to be imported is named “clustered_tweets.txt”. We assume that this file is present on your VM. Now you move it to the Hadoop cluster, using following command.

- i. **hadoop fs -put clustered_tweets.txt***
 - ii. **hadoop fs -ls***
- c. Now you have to go to HBase shell and add your column family to the specified table (ideal-cs5604s16). After successfully completing this task, exit from the HBase shell.
 - i. **hbase shell***
 - ii. **alter 'ideal-cs5604s16','clustered-tweets'***
 - iii. **describe 'ideal-cs5604s16'***
 - iv. **exit***
- d. After that, run importTsv MapReduce job to import your collection into the corresponding column family.
hbase org.apache.hadoop.hbase.mapreduce.ImportTsv -
Dimporttsv.columns=HBASE_ROW_KEY,clustered-tweets:cluster-no,clustered-
tweets:cluster-label,clustered-tweets:doc-probability ideal-cs5604s16
clustered_tweets.txt

For further details, you can refer to the file 'HBaseImportTutorial' uploaded on Canvas by the Solr team.

8. User manual

8.1 Running the Scala-Spark program on shell

Currently the dataset is accessed by the Spark driver program from the HDFS. The datasets are expected to be at the location `/user/cs5604s16_sn/input/ [a]`. The filtered tweets provided by the collection management team are used for tweet clustering and the raw Twitter data we received initially is used for be used for clustering of users.

The Scala code can be run directly in the Spark-Scala shell by first launching the Spark-Shell

```
[cs5604s16_sn@node1 ~]$ spark-shell --driver-memory 2G --executor-memory 2G
```

Here `--driver-memory` parameter represents the memory needed for the driver program and `--executor-memory` represents the memory of the executor core. Higher number of clusters needs higher memory. We found that using the current configuration, clustering can be done on the dataset upto 10 clusters. Using higher memory led to task scheduling problems on Spark, which we have already informed the Research Assistants.

Load the program in the spark-shell

```
Scala> : load <path to file>
```

Spark-shell will indicate the Scala object has been successfully loaded.

Execute the function `main` with the necessary input parameters. Refer file inventory for more description. The below example runs `word2vec.scala` using 6 clusters on `z541` collection.

```
Scala> word2vec.main(6,"541")
```

Calling the `main` function executes the code. The argument for the `main` function is the number of clusters to be used for clustering and the collection number

All the scala objects expect the input files to be present at `~/input/tweets/` for tweets and `~/input/webpages` for webpages on HDFS

The output of the code includes the number of members in each cluster and the Within Set Sum of Squares (WSSE) for the clustering displayed on the clusters. The clustering results are stored at HDFS location `/user/cs5604s16_sn/output/` as part files.

Note that the input and output file locations can be changed by modifying the file paths in the scala objects.

Here each entry indicates the tweetID and the clusterID it belongs to.

8.2 Evaluation of clustering result with LDA output

Software/Tools requirement: Java 1.7 and above

Code structure and description:

Before deploying and running the evaluation code, it is recommended to read the 'readme' file shared with the code base. The instructions and details will be mentioned in this manual as well.

The code is a simple standalone java program (1.7 and above), using a parsing library called 'opencsv' for parsing input files for evaluation.

The project consists of 3 files:

Files	Description
src/edu/vt/ir/Main.java	Contains the core logic for deriving results
src/edu/vt/ir/objects/DocumentPoints.java	POJO which contains metadata for a document
src/edu/vt/ir/objects/TopicPoint.java	POJO containing measures for a topic and its metadata

Table 12: Files for evaluation

The project has following important folders that will be needed to run the evaluation:

Folders	Description
clusterOutput	Collection wise clustering output files. (Tweets: 700, 541, 694, 668, 602, 686 Webpages: 686)
ldaOutput	Collection wise LDA output files from topic analysis team. (Tweets: 700, 541, 694, 668, 602, 686 Webpages: 686)
firstRunOutput	Preliminary stage output for a collection, showing the topics to which the cluster correlates.
finalOutput	Final detailed output for a collection. These are tab separated files which are uploaded into the HBase schema.

Table 13: Folder structure for evaluation task

Samples of files in each folder from Table 13, for tweet collection 541 are shown below:

```
(541-552940477469970432, 2)
(541-552940478292049920, 2)
(541-552940484339834880, 3)
(541-552940484574736384, 2)
(541-552940486789697536, 2)
(541-552940488068968449, 2)
(541-552940489826373634, 0)
(541-552940490350661632, 2)
(541-552940490942078976, 2)
(541-552940491810308096, 0)
(541-552940491898384385, 3)
(541-552940500861616133, 2)
(541-552940507991531520, 2)
```

Figure 50: Clustering output for tweets-541

```
541-554539937958473728 naacp,michellemalkin,bombing,naacpbombing,media
0.8676785,0.062945314,0.031863865,0.023184044,0.014328244
541-563233442352336897 michellemalkin,bombing,naacpbombing,media,naacp
0.84447104,0.059827946,0.042748537,0.03249106,0.020461395
541-555164614192033793 naacp,michellemalkin,bombing,media,naacpbombing
0.28545412,0.22317381,0.21378015,0.13973966,0.13785224
541-568926662626299904 bombing,michellemalkin,naacpbombing,media,naacp
0.9024262,0.039140016,0.036312073,0.012420325,0.009701429
541-565491025398231040 michellemalkin,bombing,media,naacpbombing,naacp
0.6455951,0.1792411,0.13786434,0.025684815,0.011614666
541-553596859512791040 naacpbombing,bombing,naacp,michellemalkin,media
0.79115105,0.08607191,0.083971344,0.029752972,0.0090526985
```

Figure 51: LDA output for tweets-541

```
naacpbombing      541
naacpbombing,suspect,sketch,msnbc,paris,
0.13029103464568,0.03049340723161587,0.025262233841054034,0.01843253118501338
,0.017804625106426723,0.016640413003371598

media      541
media,black,naacpbombing,coverage,zellieimani
0.08185242732490461,0.05897674249060026,0.05382613399917965,0.035863113248256
95,0.030825512075630706
```

Figure 52: LDA output for tweets-541, with top 5 topic words and their probabilities

```

cluster no: 0
Topics: michellemalkin media
cluster no: 1
Topics: naacp bombing
cluster no: 2
Topics: naacpbombing michellemalkin
cluster no: 3
Topics: bombing media
cluster no: 4
Topics: naacp naacpbombing
cluster no: 5
Topics: michellemalkin bombing

```

Figure 53: Intermediate evaluation output for tweets-541 mentioning top 2 topics resembling the cluster

```

541-554782118656491520 0 Naacpbombing news in media 0.9502571
541-554115795975168000 0 Naacpbombing news in media 0.67650213
541-553440263805607936 0 Naacpbombing news in media 0.77383299587
541-568851596236304385 1 NAACP stories 0.921410103
541-568895168985890816 1 NAACP stories 0.921346928
541-553102937723895808 1 NAACP stories 0.55167194

```

Figure 54: Final evaluation output for tweets-541

The Figure 54, follows the output format in Table 14 that is uploaded into HBase schema.

HBase schema design:

Document ID	Cluster no.	Cluster label	Probability
String (Tweet-id/URL)	Integer	String	Float

Table 14: HBase schema design

Executing the code:

Debugging is made easier by putting in log statements, which are commented otherwise.

The run is made per collection. (tweets and webpages collections separately)

3 inputs are required for this file to run successfully: -

- V. Cluster output file location
- VI. LDA output file location
- VII. Output file location
- VIII. Collection number

The following snapshot shows the changes to be made in main.java

```
35 // Runs are made per collection
36 static String collectionNumber = "541";
37 static String partialOutput = "firstRunOutput/partial-tweet-evaluation-"+collectionNumber+".txt";
38 static String finalOutput = "finalOutput/TweetSchemaOutput"+collectionNumber+".txt";
39
40 static String clusterOutputFile = "clusterOutput/full-"+collectionNumber;
41 static String ldaOutputFile = "ldaOutput/doctopic_"+collectionNumber+".txt";
42
```

Figure 55: Input parameters

Input files and output files for collections which were shared for evaluation are present in the code at locations specified in Table 13

Two output files will be created.

- c. partial-tweet-evaluation-<collection_number>.txt
- d. TweetSchemaOutput<collection_number>.txt

Cluster labeling is done with heuristics, however, code has a section where labels are generated using LDA topic words as well.

Run the main.java file in steps as described below.

For a new collection, run the main.java once only till step 4. Then with the top topic words generate a label from LDA topic words or using heuristics. Add these labels in the code. Now, run the entire code from step 1 to step 6 again.

These are the 6 steps to execute the code:

- I. Read clustering output
- II. Combine it with LDA output by clusters
- III. Create frequency matrix
- IV. Find relevant topics for documents in a cluster
- V. Get labels from cluster by topic from the table OR heuristics
- VI. Calculate probabilities for documents

Loading final output in HBase:

Following are the steps to load the data in the HBase schema:

- e. The file should be in a tab separated format. The sequence of entries in the file should be as mentioned in the schema diagram, Table 14.

- f. Assume your local collection data to be imported is named “clustered_tweets.txt”. We assume that this file is present on your VM. Now you move it to the Hadoop cluster, using following command.
 - iii. ***hadoop fs -put clustered_tweets.txt***
 - iv. ***hadoop fs -ls***
- g. Now you have to go to HBase shell and add your column family to the specified table (ideal-cs5604s16). After successfully completing this task, exit from the HBase shell.
 - v. ***hbase shell***
 - vi. ***alter 'ideal-cs5604s16','clustered-tweets'***
 - vii. ***describe 'ideal-cs5604s16'***
 - viii. ***exit***
- h. After that, run importTsv MapReduce job to import your collection into the corresponding column family.

hbase org.apache.hadoop.hbase.mapreduce.ImportTsv -
Dimporttsv.columns=HBASE_ROW_KEY,clustered-tweets:cluster-no,clustered-
tweets:cluster-label,clustered-tweets:doc-probability ideal-cs5604s16
clustered_tweets.txt

For further details, you can refer to the file ‘HBaseImportTutorial’ uploaded on Canvas by the Solr team.

9. Future work

The following work can be taken up in later stages:

- Use probabilistic models for clustering
- Clustering evaluation
 - Evaluate clusters with internal and external criteria
 - Implement Silhouette scoring in Spark-Scala
 - Establish ground truth for comparison of evaluation results (probabilities)
- Labeling of clusters
 - Label extraction from clusters words
 - Compare cluster labeling methods
- Clustering as a start point
 - Feed clustering results to LDA/classification/collaborative filtering for more accurate results.
- Feature Vectors
 - Principal Component Analysis (PCA) for dimensionality reduction
 - Exploring consideration of timestamp in the feature vectors of tweets

There are several problems we faced in finishing the social network:

- Crawler speed limit: crawling the user metadata is fine, but crawling the follower's list is hard since some public accounts have millions of followers. This will extend the crawling time to unacceptable. Right now, in our solution we only crawl a maximum of 50,000 followers. We may need a better idea to solve this problem. The number of tweets are also far more than the number of accounts. So the tweets crawler also faces this problem.
- Tweets tracking: in retweet process, the tweet ID of the original will not be kept and sometimes the content will be altered. It's hard to identify the entire process of retweet in our program right now. We need to find a solution for this problem.
- Visualization in D3: an interactive visualization tool is the objective of the visualization work, and we want it to be web based. We don't have a good example on this work so we need to figure this out in the future.

Appendices

Appendix A - Abbreviations

RDD- Relational Distributed Databases

TF-IDF- Term Frequency-Inverse Document Frequency

EM – Expectation Maximization

WSSE – Within Set Sum of Squares

Appendix B - File Inventory

The entire code repository, report and final presentation for the project will be available at VTechWorks.

- Sourcecode.zip – All the necessary code and readme files are present here. Please refer user manual and developer manual for the individual files. This includes code for clustering and comparison with the topic analysis team
- ClusteringReport.doc/ClusteringReport.pdf – Our report for the project
- ClusteringPresentation.ppt/.pdf -Our presentation
- CollectionStats.xls – Contains all the data collected which was used for plotting various graphs in the report.

Acknowledgement

We want to thank Dr. Fox for giving us the opportunity to work on such exciting project. We sincerely thank the GRAs supporting the course, Mohamed Magdy Farag and Sunshin Lee for advising us. We thank various teams like collection management team, Solr team and topic analysis team for their support and help. We express our gratitude to NSF for grant IIS – 1319578 and Integrated Digital Event Archiving and Library (IDEAL).

Last but not least, we thank the whole class for discussing the problems and learning about information retrieval together.

References

1. Christopher D. Manning, Prabhakar Raghavan and Hinrich Schütze, *Introduction to Information Retrieval*, Cambridge University Press. Online Edition 2009.
2. Spark documentation, <http://spark.apache.org/documentation.html>
3. Machine Learning library (MLlib) documentation, <http://spark.apache.org/docs/latest/mllib-guide.html>
4. What happens in a Twitter minute, <http://www.adweek.com/socialtimes/happens-Twitter-minute-infographic/207992>
5. David Ediger, Karl Jiang, Jason Riedy, David A. Bader, Courtney Corley, Rob Farber, William N. Reynolds, *Massive Social Network Analysis: Mining Twitter for Social Good*, 2010 39th International Conference on Parallel Processing
6. Pew Research Centre, <http://www.pewresearch.org/fact-tank/2013/10/28/Twitter-served-as-a-lifeline-of-information-during-hurricane-sandy/>
7. Sproutsocial, <http://sproutsocial.com/insights/new-social-media-demographics/>
8. Barnettcox, <http://www.barnettcox.com/nielsen-Twitter-tv-ratings>
9. S. Wasserman and J. Galaskiewicz, *Advances in social network analysis: Research in the social and behavioral sciences* vol. 171: Sage Publications, 1994.
10. B. A. Huberman, D. M. Romero, and F. Wu, *Social networks that matter: Twitter under the microscope*, available at SSRN 1313405, 2008.
11. S. E. Schaeffer, *Graph Clustering*, Computer Science Review, vol. 1, pp. 27-64, 2007.
12. Cluster analysis, https://en.wikipedia.org/wiki/Cluster_analysis (Access date - 18 Mar, 2016)
13. Andy Konwinski, Holden Karau, Matei Zaharia and Patrick Wendell, *Learning Spark*, Copyright © 2015 Databricks, Published by O'Reilly Media, Inc.
14. Kevin Dela Rosa, Rushin Shah, Bo Lin, Anatole Gershman, Robert Frederking, *Topical Clustering of Tweets*, Language Technologies Institute, Carnegie Mellon University, 5000 Forbes Ave. Pittsburgh, PA, USA
15. Pengtao Xie, Eric P.Xing, *Integrating Document Clustering and Topic Modeling*, State Key Laboratory on Intelligent Technology and Systems Tsinghua National Lab for Information Science and Technology Department of Computer Science and Technology Tsinghua University, Beijing 100084, China; Machine Learning Department Carnegie Mellon University Pittsburgh, PA 15213, USA