Graph Based Enhancement of Clusters for Effective Semantic Classification of Twitter Text

Masterarbeit im Fach Informatik
Master's Thesis in Computer Science
von / by

Russa Biswas

Advisor

M.A. Thierry Declerck

begutachtet von / reviewers

Prof. Dr. Dietrich Klakow
Prof. Dr. Vera Demberg

UNIVERSITÄT
DES
SAARLANDES

Department of Computer Science
Saarland University
Saarbrücken,Germany
August, 2016

Declaration Statement

Hereby I confirm that this thesis is my own work and that I have documented all sources used. Saarbrücken, den 31 August 2016.

(Russa Biswas)

To my parents

Acknowledgements

I would like to take this opportunity to thank my parents for their unconditional love and untiring support.

I would like to thank my advisor M.A. Thierry Declerck, for making this thesis a fulfilling learning opportunity for me. I am grateful towards him for his excellent supervision during the course of the thesis helping me with his constructive criticisms and ideas. I am much obliged for his advices on scientific writing.

I am highly grateful to Prof. Dr. Dietrich Klakow for reviewing this thesis and also for the intuitions and knowledge shared during the course of lectures.

I am indebted to Prof. Dr. Vera Demberg for reviewing this thesis and also for her advices on scientific writing and evaluation method of my work.

A special note of thanks to my friend and colleague Mittul Singh for his support and the ideas for the evaluation methods of my work, scientific writing and also to proof read my thesis document. I would also like to thank my friend and colleague Pedro Mercado Lopez to find out time and proofread my thesis document.

I am thankful to all my friends and colleagues who gladly volunteered for doing the user study in a short notice.

Lastly, I would like to thank my family and close friends for their support during the difficult times of my thesis.

Abstract

In the era of social network, Twitter plays an important role in information exchange. Huge number of tweets are sent per minute which are based on the current affairs across the globe. Several approaches have been proposed by the researchers over the years to study the nature of large number of tweets. To have a better insight of the tweets, data journalists apply these algorithms to cluster the tweets based on various factors such as events, user profiles, variations in geographical location of the users etc. Clustering of tweets is same as text clustering which results into groups of words where each group or cluster represents a single topic. However, words from any of these clusters when considered individually might have a different interpretation from the one conveyed in the tweets. Therefore, there is a gap in the understanding of the words in the cluster when the nature of the original tweets is unknown. Also, a semantic classification of this cluster content might be erroneous.

Hence, analysis methods developed for the data journalists based on text clustering are incomplete without tools for connecting the clusters to the original tweets or any other standard reference material. This becomes the focus of the thesis. We aim to find ways of making cluster and its contents more connected to the original tweets and other standard structured information source DBpedia. We find a summary of tweets for each cluster and provide semantic enrichment of the clusters of the clusters by mapping onto the most similar DBpedia article(s).

Contents

List of Figures

List of Tables

Part I

Background

Chapter 1

Introduction

Online social media, Facebook[1], Twitter[2], Instagram[3] and many more has evolved as the prime source of communication since last 10 years. The easiest and cheapest mode of communication with friends and family across the world has brought this huge popularity of social media. It also provides a platform to voice our opinion and reach out to a large mass of people within seconds. Social media also plays a vital role in building professional network. It keeps us updated with the current events happening in any part of the world. It also promotes cheap advertising and promotional activities to a huge section of population across the globe. The popularity of social media in different countries is well explained by the average hours spent per day in social media and is well illustrated in the Figure 1.1 below.

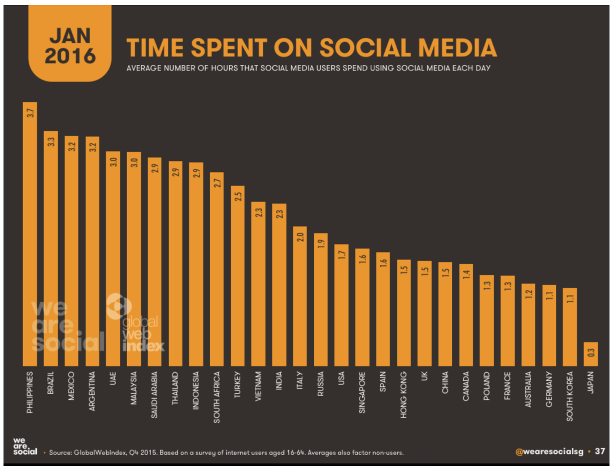


Figure 1.1: Average Hours spent on Social Media per day[84]

[1] www.facebook.com [2] www.twitter.com [3] www.instagram.com

This huge popularity of social media has given the researchers enormous topics to explore and analyse the data generated. In specific, twitter provides its data for research usage. In our study we have analysed twitter text. Twitter is a microblogging service allowing users to put up posts, popularly known as tweets, consisting of only 140 characters and remains a popular social networking site having 313 million monthly active users.[52]

Several researches have been carried out since Twitter came into existence in 2006[53]. To name a few, twitter user profiles [54, 55], sentiment analysis[56, 57], nature of tweets based on geographical variations[7], trending of breaking news[5, 61, 62], spread of info during natural calamities[58], riots [59], social events[60], sports [47] and many more are the main areas of interest for the researchers. Over time many interesting trends have been observed in the usage of twitter which has attracted the researchers to delve further into twitter secrets.

A study by American Press Institute[4] and Twitter in collaboration with research firm DB5 in 2015[4] reveals that nearly 9 in 10 Twitter users in the study (86%) say they use Twitter for news, and the vast majority of those (74%) do so daily.[4] Breaking news spreads like wildfire over the social media. More the popularity of Twitter amongst people, more the variation of tweets and more the information, views and both positive and negative opinions is circulated. During summer 2011, when riots broke out in London the news related to the riot was all over the Internet and twitter was no exception. People used the twitter platform to share the happenings in and around the city.

According to a study by one of the leading newspapers *The Guardian*[5], during the riots in London there were seven news which were popular among the Twitter users and went viral. These news includes both rumours and non-rumours. They are

- Rioters attacked London Zoo and set the animals free.
- Police beat a 16 year old girl.
- London Eye is set on fire.
- Army deployed in Bank
- Rioters attacked Birmingham Children Hospital.

[4] `www.americanpressinstitute.org` [5] `www.theguardian.com`

- Rioters broke into McDonald's and cooked their own food.
- Miss Selfridge shop was set on fire.

In this study our observations are focused mainly on the variety of these huge number of tweets which were sent during this riots in London in summer 2011. The tweet data corpus used in this study comprises of tweets of all these incidents. The same corpus is used for the study by Procter et al. [51]. It contains 2.6 million public tweets sent from 700,000 distinct twitter user accounts between 6th August and 17th August, 2011 during the riots in London. The tweet corpus was handed over to the newspaper *The Guardian* [51], under an agreement. Via the European FP7 project PHEME[6], I could have access to this very rich data set, which is the main resource used in this thesis. My thanks therefore for this to the University of Saarland, who is partner in this project.

The illustration Figure 1.2 shows how these news had spread rapidly. The graphs are generated by a study done by The Guardian newspaper[63] and shows us the number of tweets per hour over a period of 7th August to 12th August, 2011. The horizontal blue bars on each of the graphs indicate the time when a new tweet describing the respective incident had spread expeditiously.

[6] urlhttps://www.pheme.eu/

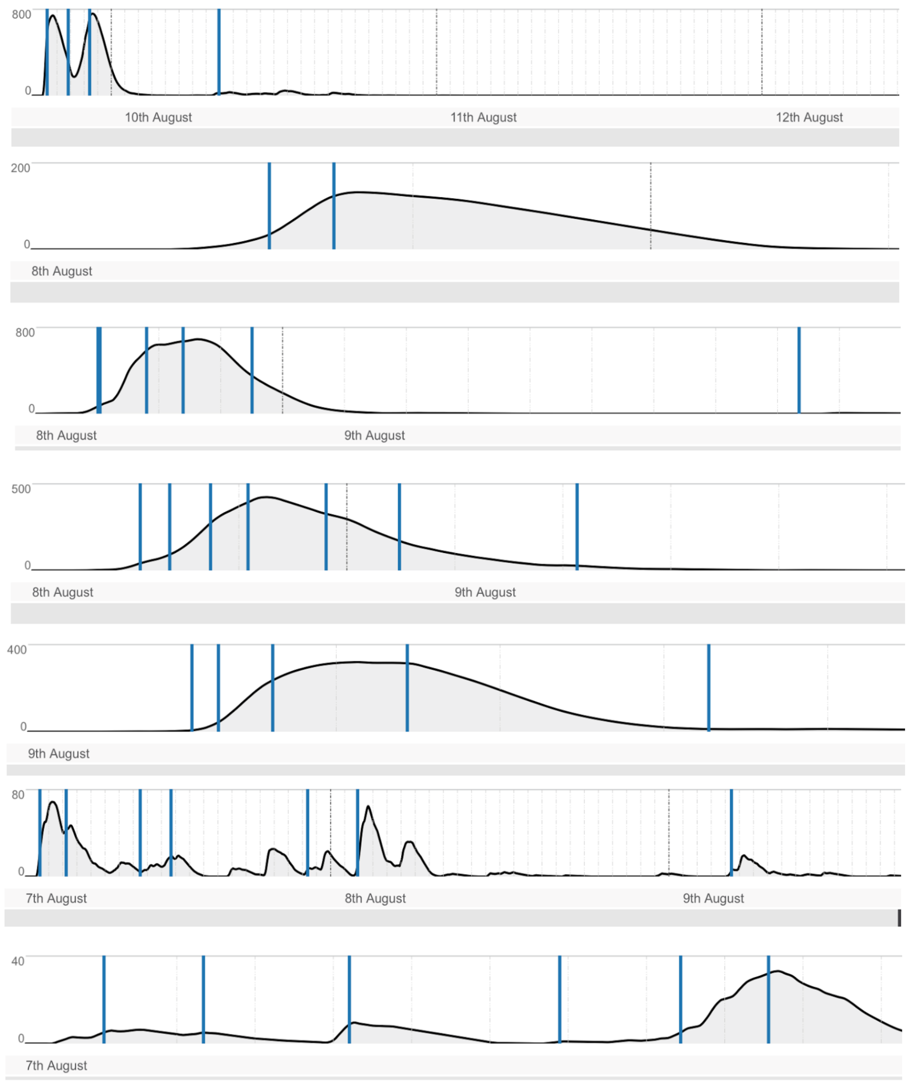


Figure 1.2: Illustration of graphs from top represent tweets/hour for the popularity of the events: Miss Selfridge set on fire, army deployed on bank, Birmingham children hospital is attacked by the rioters, rioters attacked London zoo and set the animals free, London eye is set on fire, police beat 16 year old girl and rioters broke into McDonald's and cooked food respectively.(Source: The Guardian)[63]

It is a challenging job for the data journalists to deal with millions of such tweets sent on a certain topic but comprising of many subtopics. Rather they prefer information in a concise format. The easiest way to deliver this information in a compact format to them is in form of clusters. This is because clusters are group of items which are similar to each other within a group and dissimilar between different groups. Hence, the data journalists can have information about the huge number of tweets in a compact format.

Clustering of twitter text is nothing but text clustering. However, the problem is text clustering results into clusters in which each cluster is a collection of words. The words from these clusters when considered individually might have a number of different interpretations. Some of these interpretations may be are different from the one conveyed in the original tweets. Hence, it is difficult to interpret the message conveyed by these clusters from the individual words present in the clusters without having detailed prior knowledge about the contents of the tweets. Hence, the problem of analyzing the information contained in huge number of tweets in a compact way still persists. The data journalists when provided with clusters formed from the tweets could not make sense out of it. Therefore, there is a necessity to find the correct interpretation of the words present in the cluster with respect to the original tweets.

Moreover, if the data journalists intend to have some detailed information about each cluster with the help of some reference knowledge base such as Wikipedia or DBpedia based on the individual words from the cluster, they are likely to have ambiguous or wrong information about the cluster. This is due to the same fact that words in the cluster when considered individually might be ambiguous and thus refer to various articles in these reference knowledge bases. Therefore a proper semantic enrichment of the cluster i.e. mapping of words from the cluster onto the most similar article of a reference knowledge base is necessary. These problems are addressed in this thesis.

Hence, to find a solution to these problems we aim to find the missing gap in the interpretation of individual words in the cluster with respect to the original tweets. Moreover, instead of providing the data journalists with clusters formed from the tweets, we intend to provide a summary for each of the clusters. The summary consists of a few tweets, which best represents the cluster, retrieved from the original corpus. Further, we provide a semantic classification of the clusters by mapping onto the most similar DBpedia article(s). This leads to a semantic enrichment of the clusters.

Overall, we make the following contributions to achieve this goal.

- Propose a novel mining technique to mine phrases from the words in a cluster.
- Propose a method to retrieve a few of the most relevant tweets based on the longest phrase, for each cluster.
- And then, describe a way to map the most frequent phrases onto the most similar DBpedia article(s).

The thesis document is structured as below.

In the next chapter, we present a review of related work done on clustering, phrase mining and semantic classification. The goal of the thesis and a solution approach is discussed Chapter 3. The details of the methods that have been developed and implemented to achieve the goal of this work is discussed in Part II. This section includes six chapters - chapter 4 to Chapter 9. The nature of the tweet corpus used for the study and list of preprocessing methods applied on the corpus is discussed in chapter 4. Chapter 5 includes the details of k-means++ clustering algorithm used to form clusters from the tweet corpus. Chapter 6 gives a method of post processing of the clusters formed from k-means++ algorithm. Then it goes on to provide a method to mine phrases. A word graph is generated from the mined phrases in Chapter 7 followed by graph partitioning. Chapter 8 includes methods developed to select tweets from the original corpus as summary for the clusters. The last chapter in this section Chapter 9 includes a semantic classification of twitter text.

Part III consists of the results and the conclusion of this study. Chapter 10 focuses on the evaluation metrics and the results. Lastly, Chapter 11 concludes this document by summarizing the tasks accomplished and the conclusions drawn from this study.

Chapter 2

Related Work

Recently, a significant amount of researches have focused on clustering of tweets to detect events [5, 64], find areas of interest of the users [65, 66], analyze variation of tweets based on geographical locations [67, 7] and many more. Also, semantic transformation of tweets has been an active area of research[11, 9]. This chapter describes about the previous works done on twitter data related to our study.

O'Connor et al. [6] has put forward an unsupervised approach of message summarization. They present *TweetMotif*, a faceted search interface for Twitter messages. Their application extracts a set of topics to summarize the messages with respect to user queries. The technique followed in their paper includes language modelling, syntactic filtering and ranking of the tweet messages in response to the user query. Therefore, this application summarizes tweets depending on a given query. However, we aim to we as already stated, we intend to select a small collection of tweets from the original corpus as summary for each of the cluster. Hence, it is not a suitable summarization approach for us.

Eisenstein et al. [7] presents methods to predict the geographic location of the user based on the message conveyed by the tweet he posted. Therefore, geographic tagging of the tweets are done. Dela Rosa. K., et al. (2011) Rosa automatically clusters tweets based on the topics. The approach is inspired by news aggregating services like Google News. Two approaches of topic classification of twitter text proposed by Lee. K., et al. [8] are Bag of Words based classification and network based classification. On the other hand, Sankaranarayanan. J et al. [5] builds a news processing system which captures all the tweets with respect to the latest breaking news, using a clustering approach. The system clusters the tweets based on news on a large scale. However, all these methods provide different approach of identifying similar tweets and form clusters. But, the summarization of tweets based on clusters is yet to be explored. Therefore, we propose an approach in which tweets are summarized for each cluster formed from a tweet corpus.

To achieve this goal, we mine phrases from the clusters followed by the construction of a word graph from the mined phrases. Phrase mining for named entities from blogs or social communities [68, 69] has always been an active area of research. But, we propose an approach in which phrases are mined from the words present within the cluster.

The phrase graph[45] approach has been used to track the memes[46] or the sports related events in twitter[48, 47]. The phrase graphs were constructed directly from the tweets or the status updates in social media. But, we put forward an approach in which word graphs are constructed from the phrases mined from clusters.

On the other hand, semantic interpretation of Natural Language processing based on Wikipedia has been explored [11]. Osborne. M., et al. [9] describes an approach of event detection in twitter with the help of Wikipedia. They present a method to improve the quality of the events detected from live streaming of tweets by parallel event reporting in Wikipedia in real time. Furthermore, Kapanipathi. P., et al. [10] provides a semantic enrichment of the twitter posts. Public knowledge base Wikipedia is used to identify areas of interest of the users in twitter. The inferred interests of the users are represented in form of Hierarchical Interest Graph formed by exploiting Wikipedia Category Graph (WCG). Another Wikipedia based approach of identifying topics of interest of twitter users based on their posts has been proposed by Michelson. M., et al. [65]. A *topic profile* is created which describes users' topics of interest. The words in the tweets are linked to Wikipedia pages which helps in identifying socially tagged categories. Inspired by the paper by Michelson [65], Genc. Y., et al. [13], provides a semantic transformation of the twitter messages by mapping each twitter message to the most similar Wikipedia page. Clearly, semantic enrichment of tweets has been an active area of research in recent past. However, semantic enrichment of the clusters formed from the tweets has not been explored. This gives a strong motivation for our work behind semantic enrichment of the clusters. Instead of mapping each twitter message to a knowledge base[13], we mine phrases from the cluster and map them to the most similar DBpedia articles. Therefore, the named entities present in the clusters with respect to the original tweets are mapped onto the most similar DBpedia article(s).

In the next chapter, we would define our thesis problem and a solution approach to the problem.

Chapter 3

Goal and Solution Approach

As stated earlier, there are two goals of this thesis.

First aim is to retrieve fewer number of tweets from the huge corpus of tweets for each cluster for the data journalists to make sense out of data. The second goal is to provide a semantic enrichment of the clusters formed from the tweets by mapping the cluster elements onto the most similar DBpedia article(s).

It is clear from the discussion so far that the two goals of this thesis is based on clusters. Therefore, the first and foremost step is to form clusters from the tweet corpus. Clustering of text data results into group of words, each group denoting some event or topic. The words present in the clusters is in form of unstructured text. Therefore, it is not possible to generate summaries for each of the clusters from this unstructured text. Moreover, mapping of each word from the cluster to the most similar DBpedia article might result into erroneous results as individual words might convey a meaning much different from the one conveyed by the tweets. Therefore, in order to generate a summary for the clusters and to have a proper semantic enrichment of the clusters, it is necessary to explore the relations between the words present within the cluster. Hence, to convert this unstructured text to structured meaningful units, we propose a method to mine phrases from the words present within a cluster with the help of the original tweet corpus.

Phrase mining results into a collection of structured meaningful units for each cluster. As the words present within a cluster are most similar to each other, it is likely to have a connection between the phrases mined from these words. To study the pairwise relation between the phrases, a word graph will be constructed from the phrases. To have a better understanding of the closely related phrases and to mine the most frequent phrases, graph partitioning is to be done.

Thereafter, the longest path problem is to be solved for the word graph formed from the phrases in order to have all the words from the cluster which are related to each other. Nonetheless this longest path is the longest phrase that could be mined from the cluster. In the following step, this longest phrase is used to retrieve fewer number of tweets from the huge tweet corpus and can be treated as the summary for the cluster. Therefore, a huge reduction in the number of tweets is achieved. The data journalists will be analyze the events based on much lesser number of relevant tweets.

However, the most frequent phrases mined from the graph partitioning might not be a phrase relevant to the context of the original tweets. Therefore, Wikipedia Categories, RDF predicates of DBpedia data set are exploited to find the most relevant frequent phrases. These relevant are then mapped to the most similar DBpedia article(s). All the words from the clusters which do not form any phrases are also mapped to the relevant DBpedia articles. Hence, the semantic enrichment of the clusters are achieved.

The figure below provides a pipeline of methods we propose in our work to achieve the goal of this thesis.

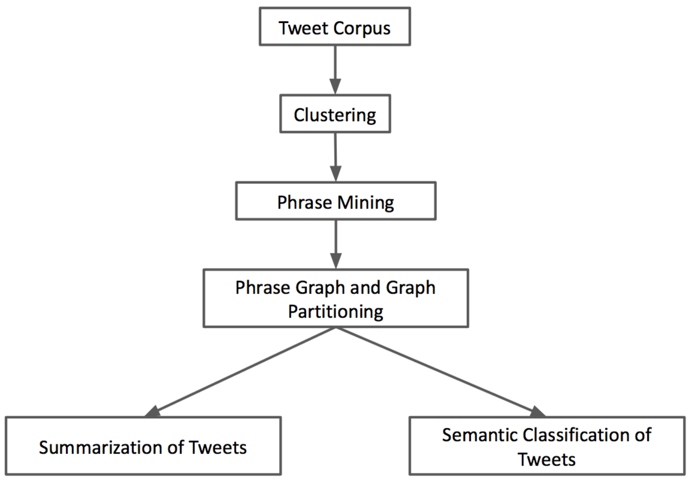

Tweet Corpus
Clustering
Phrase Mining
Phrase Graph and Graph Partitioning
Summarization of Tweets
Semantic Classification of Tweets

Figure 3.1: Pipeline of Methods to be followed

Part II

Methods

Chapter 4

Twitter Data and preprocessing

This chapter describes the tweet corpus used for the study followed by a detailed description of the preprocessing techniques applied on the corpus.

4.1 Twitter Data

The data used in this study was collected by the newspaper *The Guardian* as mentioned in Chapter 1 and made available to the European FP7 Project PHEME.

While analysing the corpus, it is noticed that the users often mentioned various URL links to Instagram pictures, YouTube videos, other tweets, newspaper articles etc. in their tweets. These URLs needs to be taken care of in the data preprocessing step as they might contain useful information.

Also, there are incomplete truncated hash-tagged words in the corpus. This provides incomplete information which is not desirable. Therefore, we find ways to tackle this issue during data preprocessing.

Apart from these two, no other significant properties are noticed in the corpus. The following section lists the data preprocessing techniques to be applied to this corpus so that it can be directly consumed by the methods described in the following chapters of *Methods* section

4.2 Data preprocessing

Data preprocessing is a method of analysing and cleaning the unstructured raw data to remove the redundant or irrelevant information present and making it accessible to various statistical modelling and machine learning techniques. It is the one of the necessary task of Natural Language Processing (NLP) , Information Retrieval (IR) and Text Mining, the complexity of which depends on the data sources used.

Data if not processed properly in this stage can lead to some misleading results [70] It also helps in easy storage and effective retrieval of the data from tweets. The pipeline of data preprocessing we followed with respect to our tweet corpus consists of the following steps:

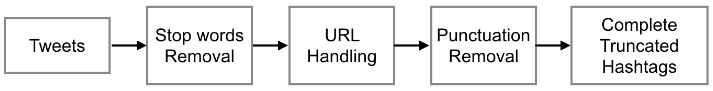


Figure 4.1: Tweet Pre-processing Steps

4.2.1 Stop Words Removal

In every language, there are a set of words which occur very frequently and are used to connect the other words together to form a sentence. These words play an important role in the grammatical correctness of the language. But they do not provide us with any meaningful information specially when taken out of the sentence. These set of neutral words are referred to as Stop words.

Since no valuable information about the content of the text is conveyed by these stop words and their frequency is high, we prefer to remove them before doing any text analysis. We have used NLTK[1]-based stopwords for English to identify and remove the stopwords. The tweet corpus used for this study comprises of 23.17% stopwords of the whole tweets.

4.2.2 URL Handling

Normally while handling twitter data, it is practised to remove the URLs in the data preprocessing step. However, these URLs might carry useful information. So, we do not remove the URLs. However, it is to be noted that an URL can be more than 140 characters, so twitter shortens the link to 10-12 characters.[5]

Rather, we create an associated array, key-value pair,to store them. We

[1] http://www.nltk.org/book/ch02.html

extract the URLs and assign an unique to key for each unique URL. Now,the URLs in the twitter text is replaced by the unique keys from the associated array. This associated array will help us to retrieve the URLs at a later stage.

4.2.3 Punctuation Removal

In any language, punctuation plays a vital role in conveying the context of a sentence. But while text processing, dealing with sentences delimited with punctuations such as $-$, $'$ etc. is often problematic.
For example, our corpus contains tweets which has hyphenated words like *Re-used* or same words without hyphen *reused*. Similarly, *children's* and *childrens* are present in the text, so we remove all the punctuations in the tweets along with the ones occurring within an alpha-numeric sequence and represent the tweets as bag of words. Therefore, *children's* becomes *childrens* and *re-used* becomes *reused* after punctuation removal.

Punctuation removal step should be done after URL handling. As URL contains punctuations like *., /, _, :* etc. so in the previous step we substituted all the URLs with the unique keys. If, it was done the other way round URLs would have been lost.

Generally, tweets would contain very few punctuation marks due to character limit. However, it is to be noted that *hashtag (#)* is very important when dealing with tweets. So it is wise to remove all the punctuation marks from the tweet collection except for the "Hashtags".

Another important aspect to be taken care of during this step is if a hash-tagged word is followed by more than one full stop (.) similar to the format (...), none of the dots are removed in this step because these dots might denote truncated hashtag words. This is discussed in more details in the next section Section 4.2.4

4.2.4 Completion of Truncated Hashtags

In order to express their thoughts within limited characters, users often end up truncating the hash-tagged words in tweets. However, an incomplete hash-tagged word followed by more than one dot (.), does not provide complete information about the content. Some of the truncated hash-tagged words found in the corpus are *#lond...*, *#london...*, *#totten....* Therefore, based on the available knowledge from the remaining tweets, we complete the missing information. However, it is noted that the data contains other complete hash-tagged words such as *#london*, *#londonriots*, *#londoneye*, *#tottenham*, *#tottenhamriots* etc. Clearly there is disambiguity now to complete the missing information. *#lond...* could be *#london* or *#londonriots* or *#londoneye*. Similarly, *#totten...*, could be *#tottenham* or *#tottenhamriots*. Completion of truncated hash-tagged words is important and we use Levenshtein distance to overcome it.

Levenshtein Distance, is a metric used to find the distance between two string sequences. Suppose, a and b are 2 strings of length $|a|$ and $|b|$ respectively then Levenshtein distance between these 2 strings is given by $lev_{a,b}(\mid a \mid, \mid b \mid)$. The mathematical formula for the same is given in Table 4.1

Levenshtein Distance
$lev_{a,b}(\mid i \mid, \mid j \mid) = \begin{cases} max(i,j) & , min(i,j) = 0 \\ min \begin{cases} lev_{a,b}(i-1,j)+1 \\ lev_{a,b}(i,j-1)+1 \\ lev_{a,b}(i-1,j-1)+1_{a_i \neq b_j} \end{cases} & , otherwise \end{cases}$

Table 4.1: Levenshtein Distance

In the mathematical equation of Levenshtein distance, $1_{a_i \neq b_j}$ is the indicator function equal to 0 when $a_i = b_j$ and equal to 1 otherwise. $lev_{a,b}(\mid i \mid, \mid j \mid)$ is the distance between the first i characters of a and the first j characters of b.[23]

The steps that has been followed to complete the truncated hash-tagged words in the corpus are mentioned below.

- Find all the truncated hash-tagged words from the tweets and save them in form of a list.
- Find all the complete hash-tagged words from the tweets and save them in form of a list.
- Form a frequency dictionary for the completed hash-tagged words, in which hash-tagged word is the key and frequency is the value.
- For each item in truncated hash-tagged list,find pairwise Levenshtein distance (refer to Table 4.1) with all the completed hash-tagged words.

- Replace the truncated hash-tagged word in the tweet with the complete hash-tagged word for which the Levenshtein distance measure is less.
- If there is a tie in the Levenshtein distance (refer to Table 4.1) for a truncated hash-tagged word w.r.t 2 or more complete hash-tagged word then replace with the one which has the highest frequency.
- Form a dictionary in which the truncated hash-tagged word is the key and the complete hash-tagged word which replaced the truncated one as value. This dictionary is used in future for reference and care is taken that no information is lost.

Continuing with the same example, if we follow the above steps we can see that Levenshtein distance of *#lond* and *#london* is much less than that of *#londonriots*, so we replace *#lond* with *#london.* In the corpus, 657 occurrences of truncated hash-tagged words are encountered among which 237 truncated hash-tagged words are unique. Out of total 487,780 number of hash-tagged words including both truncated and complete ones, the truncated ones apparently looks very less, but considering sparse twitter data, we cannot afford to loose any information. Hence, the above steps are necessary to interpret the missing information.

Chapter 5

Clustering

This chapter describes the implementation details of the clustering of the tweets.

5.1 Introduction

Text Clustering has been studied widely in the field of data science. Clustering is a method of finding coherent groups (clusters) of similar objects from unlabelled data. A more elaborate definition, for example, is stated in [71], "These clusters should reflect some mechanism at work in the domain from which instances or data points are drawn, a mechanism that causes some instances to bear a stronger resemblance to one another than they do to the remaining instances."[24]

A simple, formal, mathematical definition of clustering, as stated in [25] is the following: let $X \in R^{mxn}$, a set of data items representing a set of m points x_i inR^n. The goal is to partition X into K groups C_k such every data that belong to the same group are more "alike" than data in different groups. Each of the K groups is called a cluster. The result of the algorithm is an injective mapping $X \rightarrow C$ items X_i to clustersC_k[24].

Clustering can be broadly classified into

1. Hard Clustering - Each document or object is assigned to exactly one cluster or not.

2. Soft Clustering - Each document or object is distributed over all the clusters with a certain likelihood degree.[30]

However, it can be divided into the following categories

1. Strict Partitioning Clustering - Each document or object is assigned to exactly one cluster.

2. Strict Partitioning Clustering with Outliers - A document or an object may not belong to a specific cluster. These are termed as outliers.

3. Overlapping Clustering - Documents or objects may belong to more than one cluster. This type of clustering is also known as Multi-view Clustering or Alternate Clustering.

4. Hierarchical Clustering - As the name suggests, documents or objects are clustered in a hierarchical manner.

5. Subspace Clustering - It is a type of Overlapping Clustering but within an uniquely defined subspace clusters do not overlap.[72]

In this thesis, we want to achieve non overlapping Strict Partitioning Clustering of the tweets. The first kind of clustering in the above mentioned categories. So, we have used k-Means clustering, the simplest and best known unsupervised learning algorithm for document clustering. k-means is a strict partitioning clustering because each data point is assigned to a cluster with respect to a centroid.

k-Means is a NP hard optimization problem which finds only the local optimum.[74] Like any other NP hard optimization problem, k-means also tries to follow the common approach of finding approximate solutions. For a data set containing n objects, k-means clustering aims to partition these n objects into k clusters characterized by unique centroids or mean for each of the clusters based on the features of the objects. The value of similarity measure between the objects and the centroid is less within a particular cluster whereas inter-cluster distances are maximum.[73]

Each and every tweet in the corpus is a single document or object. In order to simplify the representation of documents or tweets, we represent each tweet as a *bag of words*. In the next section we will discuss more about representation of tweets in text clustering.

5.2 Tweet Representation

We use Vector Space Model (VSM) to represent the tweets. Vector Space Model[31] is a way of representing text documents as vectors of identifiers. As mentioned earlier, each tweet is an individual document and is represented in the form of *bag of words.* All the words present in the text is represented as a list of words, without taken into consideration the order of occurrence and grammar in the tweet. But, it does take into consideration multiplicity of the words in a tweet. Table 5.1 below provides an example for the same.

Tweet	*#tottenham eye witness tells BBC trouble erupted on the streets after police "set upon" a 16yr old girl who approached them*
Bag of Words	{*tottenham, eye, witness, tells, BBC, trouble, erupted, streets, police, set, upon, 16yr, old, girl, approached*}

Table 5.1: Bag of Words

It is to be noticed in the Table 5.1 that *Bag of Words* does not contain the stop words present in the tweet as they are removed from the corpus during data preprocessing (refer to Chapter 4 Section 4.2.1). Also, ordering of words in the original tweet and the grammar is not considered.

The tweets in form of *bag of words* are represented as vectors in the Vector Space Model, also known as term vector model. Mathematically, we know that a vector can be written as

$$v(T) = a_1\overrightarrow{v_{i1}} + a_2\overrightarrow{v_{i2}} + ... + a_n\overrightarrow{v_{in}} \tag{5.1}$$

where v(T) is the vector obtained from tweet T,
a_k are the weights, hence scalar,
and $\overrightarrow{v_{ik}}$ are the components or the elements of the vector[6]. In vector space model, terms are the axes in the space and the documents are vectors.

We have chosen the words in the tweets as terms, so the dimensionality of the vector is the number of unique terms present in the tweet corpus. Generally, in vector space model to find the relevant documents with respect to a given query similarity between each document and the query is measured, both represented in form of vectors in the space. However, we do not look for any query here. Instead, we are interested in finding the similarity between the tweets. We will see in a later section how this similarity is measured.

The tweets in the vector space can be represented mathematically as below:

$$d_i = \{w_{i,1}, w_{i,2}, w_{i,3}, ..., w_{i,t}\} \tag{5.2}$$

where d_i is the i-th tweet or document,
$w_{i,j}$ is the weight of the term j in tweet i.
If a term is present in the tweet, value of the term in the vector is non-zero. The weights of the components of the tweet vector $w_{i,j}$ also known as *term weights*, can be measured in a number of ways. The most popular amongst them are *Term Frequency*, *Term Frequency and Inverse Document Frequency.* Here, we have used the second weight measure which is the combination of *Term Frequency and Inverse Document Frequency.* We will discuss about it in more details in the next section.

5.2.1 TF-IDF

Term Frequency, as the name suggests is the frequency of a term in a document.[35] Sometimes, the term frequency is normalized by dividing it with document length. The length of the document is the number of unique terms present in the entire document. There might be documents of different lengths in the corpus, so there is a possibility of a term appearing more than once in a longer document than in a smaller document. Therefore, normalization helps in adjusting the term frequencies by eliminating the effect of gross influences of certain terms in the document.

In our work, despite of the fact tweets have a limit of 140 characters, some tweets may contain lesser words. But, clearly the possibility of having a significant difference in frequency of a certain term, present in more than one document is less. However, we want to have the frequencies measured on a notionally common scale, so we divide the frequency of a term by the total number of terms present in the tweet. Mathematically, *Term Frequency* $tf_{t,d}$ can be defined as below:

$$tf_{t,d} = \frac{\text{No.of times term(t) present in tweet d}}{\text{Total number of terms in tweet d}} \tag{5.3}$$

As mentioned above, document d referred in Equation 5.3 is a tweet.[35]

On the other hand, *Inverse Document Frequency* value tells us about the importance of a term across all the documents present in the corpus. It is the measurement of how much information is conveyed by a certain term in the corpus. Mathematically, *Inverse document frequency* $idf_{t,D}$ can be represented as follow:

$$idf_t = \log_e \frac{N}{|\{d \in D : t \in d\}|} \tag{5.4}$$

where N = $|D|$, total number of documents in the corpus and $|\{d \in D : t \in d\}|$ is the number of documents where term t appears. This means that $tf_{t,d}$ is not equal to 0. However, if the term is not present in the document, then we divide by $1 + |\{d \in D : t \in d\}|$ in order to avoid divide by zero error.[35]

It is also possible that a certain term might occur fewer times in the corpus but has a significant role in conveying information than a frequent term. If only *Term Frequency* method is used to assign the weights to the vector then we might loose out on the less occurring significant terms. *Inverse Document Frequency* method helps in scaling up the rare terms in the corpus and scaling down the frequent terms.

Therefore, the weight measure *Term frequency - Inverse Document Frequency* is given by:

$$tf_idf_{t,d,D} = tf_{t,d} * idf_{t,D} \tag{5.5}$$

Effect of common terms are reduced in this weight measure, because high value of $tf_idf_{t,d,D}$ is obtained if value of $tf_{t,d}$ is high and $idf_{t,D}$ is low. This can be explained with the help of basic mathematical logic. If a term is very popular among the tweets, i.e. appearing in many tweets then

$$\frac{N}{|\{d \in D : t \in d\}|} \approx 1$$

Therefore,

$$\log_e \frac{N}{|\{d \in D : t \in d\}|} \approx 0$$

which makes $tf - idf$ closer to zero, nullifying the strong effects of common terms. After, the tweets are projected onto the vector space, we find the similarity between the documents with the help of Euclidean distance, which is discussed in the next section.

5.2.2 Euclidean Distance

In our thesis, the metric used to find the distance between two vectors in the Vector Space Model is *Euclidean distance* [75]. In two dimension, the Euclidean distance can be mathematically expressed as below:

$$d(\vec{a}, \vec{b}) = \sqrt{(b_1 - a_1)^2 + (b_2 - a_2)^2} \tag{5.6}$$

where $\vec{a} = (a_1, a_2)$ and $\vec{b} = (b_1, b_2)$. Since, the dimension of the vectors formed from the tweets are n-dimensional. Equation (5.6) can be re-written as

$$d(\vec{a}, \vec{b}) = \sqrt{(a_1 - b_1)^2 + (a_2 - b_2)^2 + ... + (a_n - b_n)^2} \tag{5.7}$$

where $\vec{a} = (a_1, a_2, ..., a_n)$ and $\vec{b} = (b_1, b_2, ..., b_n)$

5.3 Clustering Algorithm

As stated earlier, the two goals of the thesis revolves around the improvement of the clusters by selecting fewer number of tweets as summary for cluster and also by semantic enrichment of the cluster. Therefore, generation of the clusters from the tweet corpus is the foremost task of this thesis. Needless to mention that the clusters formed are the input to the methods mentioned later chapters of the thesis. The k-means clustering for the tweet corpus is explained in the sections below.

5.3.1 Selecting k in k-means algorithm

Elbow Method [85], one of the most commonly used methods of determining k for k-means algorithm is used in our work. In this method, sum of squared errors are calculated for some values of k i.e. number of clusters. The sum of squared error is defined as the sum of squared distances from each data point to it corresponding centroid. Therefore, sum of squared distance between each tweet vector and its centroid is calculated. The sum of squared errors are plot against the clusters. The suitable value of k is selected from the graph where the curve decreases drastically. However, the curve displays a uniform decreasing pattern after that point. The Figure 5.1 below depicts the graph for Elbow method for our data. The number of clusters are plotted along x-axis and sum of squared errors in y-axis. It can be observed that the graph drops dramatically for $k = 7$ and the graph follows a definite pattern after that point. Also, as we are aware that seven different incidents are contained in the tweet corpus used. Therefore, $k = 7$ is chosen for the k-means algorithm.

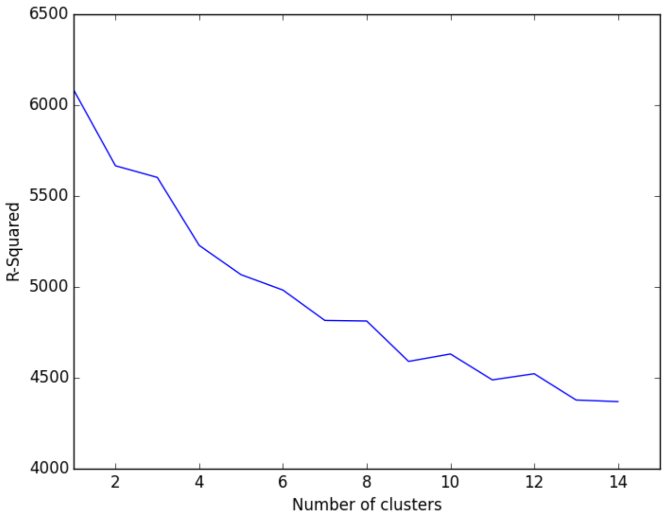


Figure 5.1: Elbow Method to determine k

5.3.2 k-means clustering for tweets

Lloyd's k-means clustering algorithm has been used for tweet clustering [73]. The algorithm separates n data points into k clusters. Here, we consider each document or tweet as a data point. The algorithm consists of the following steps:

1. Choose k random points as cluster centers also known as centroids. Let it be μ_i where i= 1,2,...,k.

   In our approach, k random vectors formed from the tweets are selected as centroids in the initial step.

2. Assign each data point to the closest centroid.

   Distance between the data points and the centroids are found using the formula

   $$\sum_{i=1}^{k} \sum_{x \in \mu_i} |x - \mu_i|^2 \tag{5.8}$$

   Assignment of the clusters is done by

   $$c_i = \{j : d(x_j, \mu_i) \leq d(x_j, \mu_l), l \neq i, j = 1, 2, .., n\} \tag{5.9}$$

   The algorithm calculates the distance between each tweet vector with

all the centroid vectors and assign the tweet vector to the centroid for which the distance is less.
Therefore, Equation (5.8) can be re-written representing the distance of the vectors to the cluster centers [39] as

$$\sum_{i=1}^{K} \sum_{x \in \mu_i} d(x, \mu_i) \tag{5.10}$$

where $d(x, \mu_i)$ is a generic distinct function given by

$$d(x, \mu_i) = (x - \mu_i)^T D (x - \mu_i) \tag{5.11}$$

where D is the distance matrix. This is the matrix formulation of expression 5.11. The assignment of the clusters is represented by a matrix U having dimension $K \times n$. The elements of U is given by $u_{ki} = 1$, if the ith data point belongs to cluster k [39]. Therefore, matrix U satisfies the condition

$$\sum_{k=1}^{K} u_{ki} = 1, \forall i = 1, ...n \tag{5.12}$$

$$\sum_{k=1}^{K} \sum_{i=1}^{n} u_{ki} = n$$

As before, the assignment of the clusters is done by

$$u_{ki} = \begin{cases} 1 & \texttt{if } d(x_i, \mu_k) \leq d(x_i, \mu_j), \texttt{for each } j \neq k \\ 0 & otherwise \end{cases} \tag{5.13}$$

3. Compute the new centroid of the clusters with respect to the newly assignment of the points

$$\mu_i = \frac{1}{|c_i|} \sum_{j \in c_i} x_j, \forall i \tag{5.14}$$

   where $|c_i|$= no. of elements in cluster c_i

   After all the tweet vectors are assigned to the nearest centroids, the centroid vectors are updated with the new assignment of tweet vectors to the corresponding centroid vectors.

4. Repeat steps 2 and 3 until there is no change in centroid assignment.

However, it is to be noted that Equation (5.8) is the sum of squared distances which is same as squared Euclidean distance mentioned in Equation (5.7). As square root is a monotone function hence it is minimum Euclidean distance.

Despite of the fact that choosing k before-hand might not seem to be feasible but experimentally we have seen Elbow method and Gap Statistic Method provided us with optimal suitable k. However, flat clustering method is chosen over hierarchical clustering method because run time for the later is $O(n^2)$ while for any flat clustering method it is $O(n)$. Therefore, for a huge collection of tweets hierarchical clustering will be very slow in execution.

5.3.3 k-means++ clustering for tweets

The termination condition of k-means clustering algorithm is that the assignment of data points to a cluster does not change. So, each step is designed to achieve the local optimum. We have already seen that in the first step of Lloyd's algorithm mentioned above, k random points are chosen as centroids. Therefore, this local optimum obtained in each step is solely dependent on the initial choice of points. The perfect choice of the initial centroids could lead to a local optimum much closer to the global optimum. To make a good choice of initial points a new algorithm is introduced by Arthur, D.; Vassilvitskii, S. [38] known as k-means++. The first step of selecting random points as cluster centers in Lloyd's k-means clustering is replaced by the following steps in k-means++ algorithm.

1. Choose the initial cluster center say μ_1 at random amongst all data points.

   In our approach, choose a random vector from the tweet vectors as the initial centroid.

2. For all the data points, compute D(x), the shortest distance from data point x to the nearest already chosen cluster center μ_1.

   This means distance is calculated between each tweet vector and the vector chosen as the initial centroid.

3. Choose the next center μ_i equal to an data point x' with a probability

$$p = \frac{D(x')^2}{\sum\limits_{x \in X} D(x)^2} \tag{5.15}$$

This weighting measure used is known as $D(x)^2$ *weighting.*

The next centroid to be chosen is the tweet vector with the probability given in Equation (5.15)

4. Step 2 and 3 are repeated until k initial points are selected.
5. Continue with the normal k-means algorithm already mentioned in the last section.

The data points mentioned in the above algorithm are the vectorized tweets. However, even though 3 more steps are introduced it does not have much effect in the run time of the algorithm. Clearly, the first step of choosing cluster centers takes more time compared to the random selection method of k-means. But, the rest of the algorithm converges quickly because of better choice of initial points. Thus the overall computation time is lowered.[38]

In our work, k-means++ algorithm when applied to the tweet corpus resulted into seven clusters, each containing a group of words. Table 5.2 provides an example of two of the given below are 2 of the seven clusters formed from the tweet corpus.

Cluster 0	*girl, police, 16, old, tottenham, year, started, riot, bbc, beat, battered, brixton, url_248, eyewitness, beating, news, riots, sparked*
Cluster 1	*zoo, london, londonriots, animals, broken, tiger, let, escaped, loose, large, reports, hearing, far, streets, true, tigers, camden*

Table 5.2: Clusters

Therefore, the clustering of the tweets resulted into group of words which are similar to each other within the group. But, these clusters are in form of unstructured text. Data journalists when provided with these clusters could not make much sense out of the data. Hence, unfit for analyzing the underlying nature of the tweets as the individual words from the cluster might have ambiguity in interpretation.

To solve this ambiguity of interpretation, processing of the clusters is required. Since the words within the clusters are similar to each other depicting some common event or topic, therefore in the next chapter we would try to transform the unstructured text into structured meaning units using phrase mining.

Chapter 6

Phrase Mining

In the last chapter, clustering of the tweet corpus using k-means++ clustering algorithm is discussed. However, a close look into the clusters (refer Table 5.2) reveals that there is room for improvement in the cluster output making it suitable for the phrase mining process to follow . The first part of this chapter describes a post processing technique of handling URLs to improve the quality of the cluster output. The second part of the chapter includes a detailed description of the methods developed and implemented for mining of phrases from the clusters.

6.1 Post Processing

Post processing is an important technique of refining the output of a process to make it accessible for the other methods to follow and also for better user understanding. In this section, post processing technique to refine the clusters is discussed.

First and foremost, the URLs which were removed by unique keys (refer to Chapter 4 Section 4.2.2) are reverted back.

6.1.1 URL handling

The clusters obtained by applying k-means++ clustering algorithm in the tweet corpus may contain some of the keys from the URL dictionary (refer to Section 4.2.2). The keys are being replaced with the corresponding values from the dictionaries. For example,

Cluster	*girl, police, 16, old, tottenham, year, started, riot, bbc, beat, battered, brixton,* ***url_248****, eyewitness, beating, news, riots, sparked*

Table 6.1: Cluster

In the cluster mentioned in Table 6.1, *url_248* is a key in the url dictionary created during pre-processing. In this step, it is replaced with the url corresponding this key from the dictionary.
After replacement the cluster looks like

Cluster	*girl, police, 16, old, tottenham, year, started, riot, bbc, beat, battered, brixton, http://t.co/RNSH0rI, eyewitness, beating, news, riots, sparked*

Table 6.2: Cluster after URL Handling

This method is repeated for all the clusters to replace the URL keys with the original URLs. The reason behind retention of URLs is that these are links to Youtube videos, newspaper articles, media sources, Instagram images, links to other tweets etc. which are used by the twitter user very frequently, hence providing important relevant information. Therefore, the cluster outputs now containing the actual links would help the data journalists to get hold of the other sources of information related to the tweets.

6.2 Mining of phrases

As already discussed, clustering of tweets resulted into groups of words called clusters where cluster represents a topic. However, when words in the cluster are considered individually it does not always make the same sense as conveyed by the original tweets. Thus, it becomes difficult to interpret the content of the original tweets a certain cluster is representing by looking at the individual words. Let us take an example to have better understanding of the problem. One of the seven clusters obtained from the k-means++ clustering algorithm is considered in Table 6.3

Cluster	*zoo, london, londonriots, animals, broken, tiger, let, escaped, loose, large, reports, hearing, far, streets, true, tigers, camden*

Table 6.3: Cluster subjected to Phrase Mining

In the cluster mentioned above in Table 6.3, we can see that the words *London*, *zoo*, *eye* etc. are present. As discussed earlier, our final goal is to have an effective semantic classification of the cluster output i.e. effective mapping of the words from the cluster onto DBpedia. So, the individual words *London*, *zoo*, *eye* when mapped onto DBpedia it might not refer to the exact information conveyed by the tweets. DBpedia provides separate dedicated page for each of the three words *London*, *zoo* and *eye* containing detailed information of the topic.
`http://dbpedia.org/page/London`, `http://dbpedia.org/page/Zoo` and `http://dbpedia.org/page/Eye` are the URLs of DBpedia articles for 'London', 'zoo' and 'eye' respectively.

However, there might be chances that in the original tweets 'London eye' or 'London zoo' are referred instead of these three individual words. Therefore, we intend to find phrases from the words within the cluster. Phrase mining technique helps in extracting salient features from the cluster converting the unstructured cluster into structured meaningful units. As a result, it would help us to find the underlying story of each cluster.

As we know, a phrase is a group of words appearing as a unit within a sentence, portraying a specific concept. The agenda here is to form phrases from words in the cluster with the help of the original tweets.

Bedathur. S., et al [68] proposed a technique of mining top-k interesting phrases from ad-hoc subsets of the corpus using indexing technique. However, we intend to mine phrases from the clusters with the help of the original corpus but the position of the phrase in the documents is beyond the interest of this work. Therefore, a method involving the formation bigrams from the original tweets and comparing with bigrams formed from the cluster is proposed.

Before, the steps to mine phrases are explained we would throw some light on the concept of bigrams for better understanding of the method developed. A bigram is a sequence of two adjacent elements in a string of tokens. Here, each tweet is considered as a string of tokens and each word in the tweet as elements. A combination of any two adjacent words in the tweet is a bigram. For example, let us consider the tweet

First hand account on BBC News: police beating a16 year old girl sparked the riot.

Here, any combination of two words such as *on BBC* or *BBC News* forms a bigram. So from each sentence of length *N*, i.e. number of words present in the sentence is *N*, we can have *N-1* bigrams. But not all bigrams are important for us as they do not provide relevant information. The most obvious way of finding relevant bigrams is

1. Find all possible bigrams for all the tweets in the tweet collection. For e.g. Bigrams formed from a single tweet *First hand account on BBC News: police beating a16 year old girl sparked the riot* are given in Table 6.4.

Tweet	*First hand account on BBC News: police beating a16 year old girl sparked the riot*
Bigrams	*First hand, hand account, account on, on BBC, BBC News, News police, police beating, beating a16, a16 year, year old, old girl, girl sparked, sparked the, the riot*

Table 6.4: Bigrams obtained from the tweet

Similarly all possible bigrams are obtained from all the tweets in the corpus.

2. Form all possible permutation of bigrams from the words present in the cluster in which order of the words in the phrases matter i.e. all possible pairing of words present in the cluster. Table 6.5 consists of an example to have a better understanding of this step. The same cluster example used previously in this chapter is considered.

3. Compare the bigrams formed in step 1 with the bigrams obtained in step 2. If they match, we conclude that these bigrams provides us with relevant information.

But this approach has some major drawbacks. Finding all possible bigrams and storing them in form of lists requires a lot of space and increases space complexity as well as the time complexity. Also, not all the bigrams

Cluster	**Random Bigrams**
zoo, london, londonriots, animals, broken, tiger, let, escaped, loose, large, reports, hearing, far, streets, true, tigers, camden	*london londonriots, londonriots london, london zoo, london eye, london reports, london animals, london tiger, london tigers, london broken, zoo london, eye zoo etc.*

Table 6.5: Random Bigrams from Cluster

formed in the second step are useful for us. To avoid these pitfalls we apply the following steps to mine phrases.

1. For each word in the cluster, find the list of words which occurs just before the word and the list of words which just after the word in the original tweets.
2. Compute bigrams out of this list of words in the form $\{previous\_word, cluster\_word\}$ and $\{cluster\_word, next\_word\}$.
   Here, the word before the cluster word in the original tweets is referred to as *previous_word* and the word after the cluster word in the original tweets is referred to as *next_word*.
3. Check if the same bigrams can be formed from the words in the clusters.
4. If such bigrams can be formed from the words in the cluster we call them phrases.
5. Repeat the above steps for all the clusters.

Let us put forward an example for better understanding. Re-using the same examples used before

The words *police*, *girl*, *16*, *old*, *year*, *beating* etc. present in the cluster does not explain the original content when considered individually. This is due to the fact we are yet to find the topics each cluster is depicting. So, we attempted to put them together. By following the steps as mentioned above we have formed bigram phrases like *16 year*, *year old*, *police beating* etc. Table 6.6 shows a few of such bigrams obtained from the cluster with the help of the original tweets.

Cluster	**Bigrams**
girl, police, 16, old, tottenham, year, started, riot, bbc, beat, battered, brixton, http://t.co/RNSH0rI, eyewitness, beating, news, riots, sparked	*(beating 16), (police beat), (girl battered), (old tottenham), (police battered), (girl sparked), (year old), (riots started), (battered 16), (sparked tottenham), (tottenham girl), (police tottenham), (beat 16), (riot police)* etc.
zoo, london, londonriots, animals, broken, tiger, let, escaped, loose, large, reports, hearing, far, streets, true, tigers, camden	*(tiger escaped), (londonriots london), (london zoo), (tigers roaming)* etc.

Table 6.6: Bigram Phrases mined from Clusters

Until now, phrases are mined from the clusters which would eventually help in summarization of the tweets for each cluster and also effective semantic classification of the cluster output. To obtain the summarization of the tweets a link between the words of the clusters needs to be established to have a better understanding of the underlying meaning conveyed in each cluster. To bridge this gap, a phrase mining method to form structured meaningful units from the words in the cluster is proposed in this chapter which brought a step closer to the goal by connecting the words together. In the next chapter, we would explore the relation between these phrases and discuss methods to connect them together to have summary of tweets.

Chapter 7

Word Graph and Graph Partitioning

7.1 Introduction

This chapter contains detailed description of the methods developed and implemented to establish the connection between the mined phrases (refer to Chapter 6 Section 6.2). As stated earlier, these phrases are bigrams formed from the clusters with the help of original tweet corpus. It is to be noted that a certain word from the cluster can be part of more than one phrase. Hence, the phrases having common words can be linked up together. It has been seen in the past that graphs are the most common choice to represent such a model and study pairwise relations between objects. Therefore in our study, graphs are used to analyze the pairwise relation between the phrases. However, in English language ordering of the words in a sentence is important. This is because minute change in the order of the words in the sentence might convey a meaning different from the original sentence. Therefore while studying the pairwise relation of the phrases, it is important to know the order of the words in the phrases. Hence, an efficient and feasible way of doing it is to construct a directed graph of words from these phrases.

There are a number of techniques to generate word graphs. We intend to build a word graph denoted by G = (V,E) representing the phrases, where V denotes the set of vertices or nodes of the graph and E denotes the set of edges between the vertices. As it is a directed graph so there exists directed edges or arrows between nodes. We will discuss about the different word graph construction techniques in the next section and exploit their properties to define the set of vertices and the set of edges to model the relationship between the mined phrases. However, since the graphs are to be generated from the phrases formed in the last section, so it is needless to say that the graphs to be generated will be finite graphs. Different set of phrases have been mined from different clusters, so there will be atleast one graph for each cluster.

7.2 Related Work on Word Graphs

In the past, researchers have come up with different methods to represent the words in the form of graphs. Sergey Kitae et al. [44] defines word-representable graph which studies the different patterns of letters within a word. But, we are interested in studying the relation between pairs of phrases. Pattern of words within the mined phrases is not the goal of this step. Therefore, word representable graph method could not be extended to our study.

Directed Acyclic Word graph is another most commonly used word graph to study the pairwise relation between words. Directed Acyclic Graph (DAG), as the name suggests is a directed graph that has no cycles. The precedence constraint i.e. the ordering of the occurrences of the words, for DAG is represented by a tuple of vertices (v_i, v_j), i.e. v_i precedes v_j so there exists a directed edge from v_i to v_j.[76] This property of DAWG satisfies the criteria of the ordering of phrases in our study.

Directed Acyclic Word Graph (DAWG), also known as Deterministic Acyclic Finite State Automation[45] is a data structure that represents set of strings. It inherits all the properties of Directed Acyclic Graph. It consists of a single source vertex and each vertex has at most one outgoing edge, hence the name Deterministic Acyclic Finite State Automation. Each edge is labelled by one possible letter or symbol. Therefore, the sequence of letters or symbols along the path of the graph from source vertex to any leaf vertex represents a string. The source vertex is considered to be the one which has no incoming edge to it. It is to be noted that DAWG nodes can have multiple parent nodes. The example given in the illustration below provides a much better understanding.

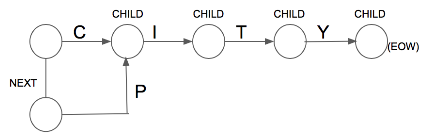


Figure 7.1: Directed Acyclic Word Graph

Here, we have a very small DAWG containing the words City and Pity. EOW represents End of Word, i.e. the goal state for a certain word. Now, if we want to search for the word City in the graph, first we have to find the letter C, then travel along the path of the graph until we reach EOW. So, here C followed by I, T and Y respectively gives us the word City. This DAWG contains two words with different beginning but with same ending. DAWGs with same beginning and different endings are also possible. A leaf node means end of a word. Multiple parts of different words can also be similar. However, one can encounter more states after one EOW is reached.[77]

The Directed Acyclic Word Graph considers the ordering of the letters in the words and each edge of the graph is denoted by a symbol. This helps in easy access to the attached information to each symbol. Therefore, this method can be extended to be applied in our study to construct a word graph from the phrases mined (refer to Chapter 6 Chapter 6).
For example, *London Eye*, *London Zoo* are two of the such phrases mined from the clusters. So, a modified version of Directed Acyclic Word Graphs can be constructed to accommodate the phrases by replacing the letters in the edges with words. The intuition of this modified directed word graph is illustrated in the Figure 7.2 below.

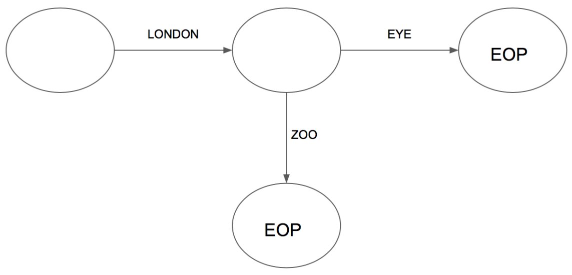


Figure 7.2: Intended Word Graph

The edges contains the words from the phrases. The edges are directed from *London* to *Eye* or to *Zoo* in accordance to the order of the words in the phrases. EOP in the graph denotes End of Phrase. A closer look into the intuition of this modified word graph allowed us to dig into another type of commonly used graph for set of strings, known as phrase graph.[45]

7.3 Phrase Graph

Phrase graph as explained by Nichols, J. et al. [47]

The phrase graph consists of a node for each word appearing in any status update, and an edge between each set of two words that are used adjacently in any status update.

The authors of this paper[47] found out ways to summarize sports events from the status updates in Twitter. They formed phrase graph from the longest sentence of each status update. The illustration below is an example from the paper which helps in understanding the structure of phrase graph for the status updates:
Landon Donovan scores!, Donovan scores a brilliant strike, Landon Donovan is brilliant.
Each word in the status is labelled as each node in the graph. The weight as-

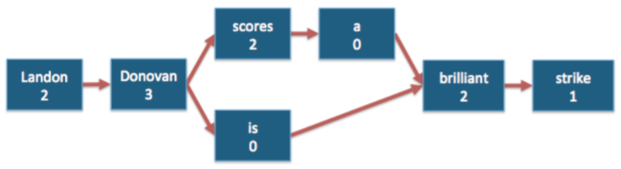


Figure 7.3: Phrase Graph (**Source:**Fig.3 Nichols, J., et.l,(2012) [47])

sociated with each node gives the frequency of the word in the corpus. While tweeting, retweeting or updating status related to a certain topic many words or phrases are used repeatedly by the users. This is due to the fact that the users are referring to the common elements in the status. Therefore, a lot duplicate words or phrases are found. Instead of putting same data in the graph over and over again, the authors removed the duplicate tokens with a score. Stop words are considered to have 0 frequency, hence *is* and *a* has weight 0 in the graph (see Figure 7.3). Alpha-numeric characters are also ignored. A directed edge between two nodes v_i and v_j implies the word represented by node v_i appears before the word represented by node v_j in the status. This phrase graph is similar to the word graph we intend to construct as shown in Figure 7.2. Therefore, it is important to know the steps to construct a phrase graph.

7.3.1 Construction of Phrase Graph

Sharifi, B., et al. [48] explains an easy method to build a phrase graph. The methods selects the twitter posts which contains common phrases on one side of the search phrases and then select the common phrases on the other side from the selected twitter posts. The phrase graph proposed in this paper is acyclic, therefore we can conclude that it is a tree. So, the terms graph and tree are used interchangeably in this section. The steps proposed by the authors to construct the phrase graph are as follows.

1. Graph considers a search topic or a phrase to be the root node.

2. The left hand side sub graph of the tree will contain the words or phrases appearing before the root node phrase in the input sentences whereas the right hand side will consist of the words appearing after the root node phrase in the input.

3. For each duplicate token encountered, in the set of input sentences containing the root node for this specific graph, the corresponding count for the token is incremented.

4. The position of the words are considered while calculating the frequency of the words. If a certain word appears in two sentences containing the same root node phrase, but once to the right of the phrase, another time to the left, then, the frequency of the word is counted one for the right tree and one for the left tree.

5. For each unique word encountered a new node is added to the left subtree or the right subtree depending on the position of the word in the input sentences.

It is clear that this approach merges the common words and the subsequences present in the set of input sentences depending on their position. Thus, it can be concluded that it acts similar to compressed tries. However, compressed tries does not work well with large amount of input sentences due to memory over burden. This algorithm as suggested by Sharifi [48] is easy to implement but it involves huge memory space. Moreover, this method determines the mostly overlapping phrase depending on the search topic.

On the other hand, Daciuk, J., et al. [45] has provided a minimal approach to construct the finite state automation for a set of strings. The method introduced in this paper adds new string one by one and minimization of the resulting automation is done on the fly. In case of sorted data, when a new word is to be added to the graph, only a sub part of the whole graph is traversed and altered. The new word might start with a symbol which is entirely or partially same as the first symbols of the existing words in the graph. Therefore, only the suffix needs to be altered. Similarly, if the new word starts with different symbol, and the tail part already exists in the graph links are to be created from the newly formed nodes to the already existing suffix. Therefore, each step of new word insertion includes one register search step and one register insertion step. The authors proposed the usage of hash tables to have a constant time complexity. The algorithm proposed is quite efficient having a time complexity of $O(l \log n)$, where $O(\log n)$ is the search complexity, n being the number of states in the minimized dictionary and l is the total number of letters in the input list. In case of unsorted data, some of the states are cloned if a confluence is encountered followed by alteration of nodes.

In our study, we intend to establish a connection between the phrases in the cluster with the help of the overlapping phrase words. Since the phrases are bigrams therefore it is unlikely to have a common prefix or suffix subsequence with respect to a search word. Therefore, the phrase graph construction method[48] cannot be solely applied in our study. On the other hand, Construction of Deterministic Acyclic Finite State Automaton or DAWG [45] would be advantageous in construction of phrase graph because it considers prefix, infix and suffix redundancy. As, a result much lesser states needs to be created and traversed compared to a compressed trie structure. Also, it has much lesser time complexity as well. Therefore, a combination of phrase graph and directed acyclic word graph would help in achieving the goal.

7.4 Directed Word Graph for Phrases

Our main interest to construct a phrase graph lies in finding out the missing links within the cluster words. In Section 6.2 of previous chapter, we have seen that bigrams are formed out of words in the cluster with the help of original tweets. Each cluster will contain the set of words closely related because these words describe the same incident. Therefore, it is likely to consider that the words in the cluster must have occurred several times in the original tweets. These words are nothing but the descriptors that describe the contents within the cluster. So, the intuition is that the phrases mined in Section 6.2 must have occurred multiple times in the tweets. Instead of finding the occurrences of individual words as suggested in the research work[48, 47, 45] discussed in last section (refer to Section 7.3.1), the occurrences of the bigrams formed from the cluster words are calculated. So a dictionary containing bigrams as keys and their corresponding frequencies as values is constructed. It also helps us to maintain the order of the occurrence of the words in the original tweets. For example, we consider *london riots* and *riots london* as different phrases because the ordering of words is important. We will use the frequency of the bigrams in the phrase graph as weights. Another improvisation is made to the phrase graph algorithm is that the weights are given to the edges of the graph and no longer associated with individual nodes. The following algorithm is used in our method to generate such a phrase graph.

Directed Word Graph for Phrases
Input : Set of phrases
Output : Word Graph formed from this set of Phrases

---

1. Choose a random phrase from the set of mined phrases.
2. As the phrases are bigrams, the two words present in this phrase forms the two initial nodes of the graph.
3. A directed edge is constructed from node u to node v if the word u appears before the word v in a phrase *(u,v)*
4. Each edge has a weight associated with it which represents the frequency of the phrase *(u,v)* in the original tweet corpus.
5. Check the list of phrases to find if there exists any phrase(s) containing either of the words from the initial phrase.

   (i) If yes, add a new node to the graph and a directed edge depending on the order of the words in the phrase as mentioned in Step 3. Further, recursively continue step 5 until no new nodes are to be added.

   (ii) If no, continue step 1-5 for all the remaining phrases.

Let us consider an example to understand the algorithm properly.

Cluster	*girl, police, 16, old, tottenham, year, started, riot, bbc, beat, battered, brixton, http://t.co/RNSH0rI, eyewitness, beating, news, riots, sparked, 16yr*
Few Phrases	*tottenham riot, tottenham girl, riot police, police beating, beating 16*

Table 7.1: Cluster and few Phrases mined from the cluster

Now, some of the phrases formed from this cluster (refer Section 6.2) are mentioned in Table 7.1.

To construct a phrase graph from these phrases an empty graph is considered to start with. Therefore, the first and foremost step to construct the word graph for phrases is to select a phrase randomly. Let it be *tottenham, riot*. Two nodes, one for *tottenham* and the other for *riot* is added to the empty graph. Now, *tottenham* appears before *riot* in the phrase *tottenham riot*, so a directed edge is constructed from *tottenham* to *riot*. The value at the edge is added to represent the frequency of the bigram *tottenham riot* in the original tweet corpus. The frequency of the phrase *tottenham riot* is 124 so the edge from , tottenham' to 'riot' has a weight value of 124. In the next step, the list of remaining phrases are checked to find if any of these two words in the graph are present in any of the other phrases. In this example, *tottenham girl* and *riot police* are the two overlapping phrases with the previous phrase *tottenham riot*. Therefore, two more nodes are added to the graph and a directed edge from *tottenham* to *girl* and another directed edge from *riot* to *police* are constructed. The same process is repeated for the new nodes added to the graph. Below is the illustration of an example of a directed word graph created from the cluster.

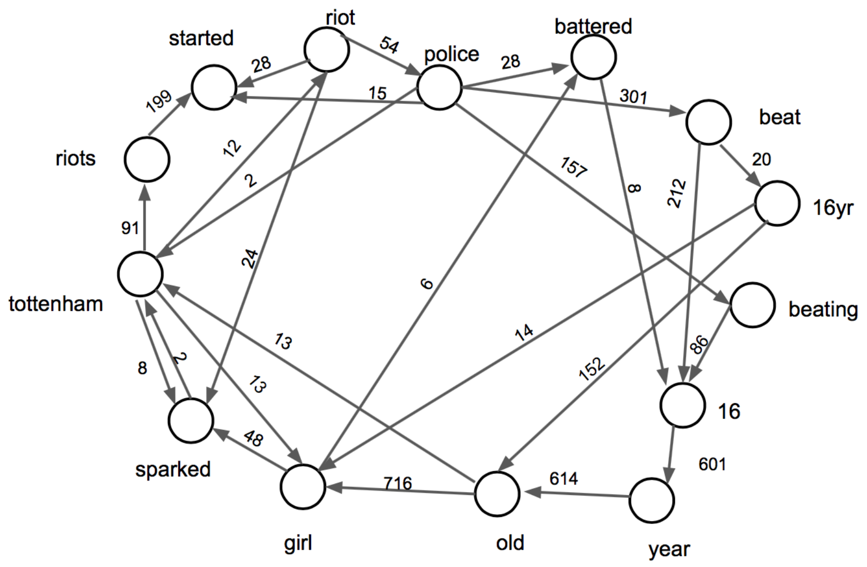


Figure 7.4: Directed Word Graph for Phrases

However, it is to be noted that overlapping of words within phrases forms the basis of the graph. Therefore, there is a possibility of cycles to exist in the graph hence it is non-acyclic. Hence, there is a necessity to partition the graph and remove the cycles to obtain the longest path from the graph which would eventually help in the selection of tweets for each cluster.

7.5 Graph Partitioning

Graph Partitioning is a NP hard problem and are solved mostly by heuristic methods. Some of the methods are based on local search strategies whereas other are based on global search. Kernighan-lin algorithm proposed by Fiducia, M. and Mattheyses [78] is one of the most widely used heuristic local search based graph partitioning algorithm. The algorithm attempts to find out two disjoint subsets of nodes A and B from the whole set of nodes V, of same size such that the sum of the edges of the nodes from A to the nodes in B are minimized. This is done by choosing some random initial partition of the set of vertices V. However, wrong choice of initial partition might end up in producing worse results.

Multiway Cut problem is another possible method of graph partitioning. As explained by Calinescu, G., et al. [49] multi-way cut problem for an undirected graph. The main aim is to delete a set of edges such that each terminal in the set T belongs to a separate component. The total weight of the edges to be deleted should not exceed a certain weight W. Since ordering of the words in the phrases are important hence the option of replacing directed edges with the undirected edges of the graph and perform multiway cut problem is discarded.

Another approach of phrase graph as explained by Leskovec, J., et al. [46] is a graph whose each node consists of a whole phrase. The authors provides a method for meme tracking with applications to the news cycle and find the popular memes and their changing patterns over a given period of time. The original quote(s)/meme(s) is(are) the root node then other nodes are added subsequently over time with slightest variation of the original post. Also, at some point of time it is found that one post is influenced by two or more memes/quotes. Clearly, there is a possibility of more than one incoming edges for a certain node. Figure 7.4 is an example taken from the same paper[46] to have a better understanding of the phrase graph. A small portion of the variants of Sarah Palin's quote is displayed in the graph. We can see two variants(nodes) are giving rise to a single variant(node) at the beginning of the graph, which in return gets split into two parts and it goes on in the similar way. So as mentioned previously, it can be noticed in the figure that more than one incoming edges (coming from nodes marked as node_1 and node_2) to the 3rd node(marked as node_3)

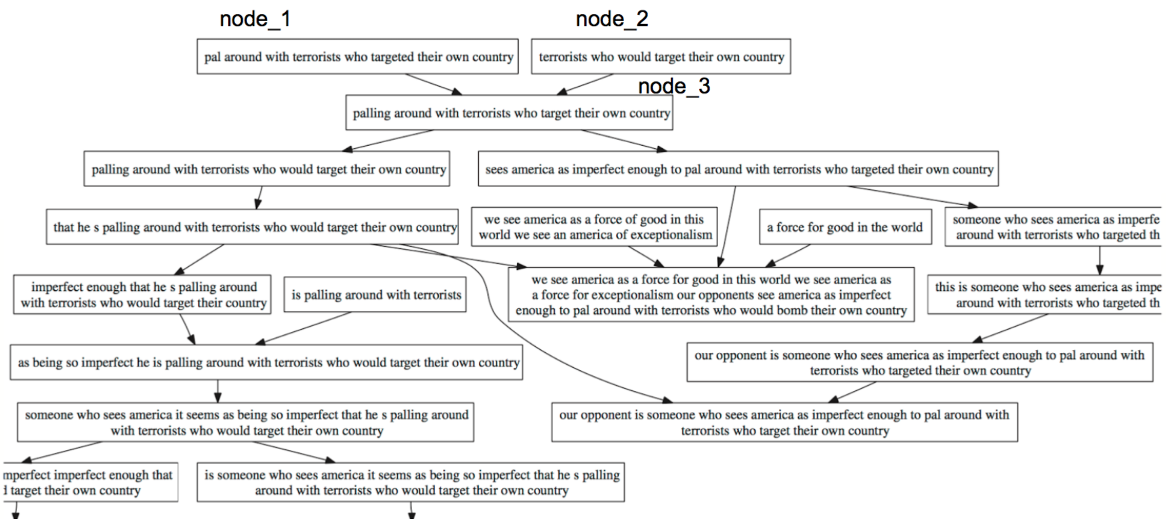

node_1
node_2
pal around with terrorists who targeted their own country
terrorists who would target their own country
node_3
palling around with terrorists who target their own country
palling around with terrorists who would target their own country
sees america as imperfect enough to pal around with terrorists who targeted their own country
that he s palling around with terrorists who would target their own country
we see america as a force of good in this world we see an america of exceptionalism
a force for good in the world
someone who sees america as imperfe around with terrorists who targeted th
imperfect enough that he s palling around with terrorists who would target their country
is palling around with terrorists
we see america as a force for good in this world we see america as a force for exceptionalism our opponents see america as imperfect enough to pal around with terrorists who would bomb their own country
this is someone who sees america as impe around with terrorists who targeted th
as being so imperfect he is palling around with terrorists who would target their own country
our opponent is someone who sees america as imperfect enough to pal around with terrorists who targeted their own country
someone who sees america it seems as being so imperfect that he s palling around with terrorists who would target their own country
our opponent is someone who sees america as imperfect enough to pal around with terrorists who target their own country
mperfect imperfect enough that l target their own country
is someone who sees america it seems as being so imperfect that he s palling around with terrorists who would target their own country

Figure 7.5: Phrase Graph **Source:** Figure 1 Leskovec, J., et al. [46]

Similar issue has also been encountered in our the phrase graph generated (see Figure 7.4). It is important to have a proper understanding of data regarding what is vital and what is incidental. The goal in this step is to find most frequent pairs of words i.e. bigrams. Similar to the phrase graph as illustrated in Figure 7.5 [46], the word graph for phrases constructed in our work also have more than one incoming edges to a given node. Therefore, we follow the heuristic graph partition technique suggested by the authors of this paper [46]. Partitioning of the graph G(V,E) is done by deleting the set of edges having minimum total weight, so that each resulting component is single rooted.
The following steps are developed and implemented in our work to partition the word graph formed from the phrases.

1. For each node find the set of all the minimum incoming edges.
2. Delete these set of edges

As a result, all the nodes in the graph will have indegree(v) = 1 , where v is any vertex from the vertex set V, i.e. only one incoming edge for each vertex/node. Deleting the edges having minimum weight implies we have the node which has the highest weight. Therefore, our graph represents only those phrases which are most frequent in the tweets. Figure 7.6 illustrates the steps involved in graph partition. The crosses marked in red in the Figure 7.6 are some of the nodes to be deleted from the graph. For example, node labelled *tottenham* has three incoming edges having weights 2, 2, and 13 respectively. The edge with a red cross marked on it in the Figure 7.6 is the edge with weight 2 coming from node *police*. This edge will be removed during graph partitioning. Another incoming edge having weight 2 to the node *tottenham* from the node *sparked* is also removed in this step. The edge with weight 13 is retained. Therefore, it can be concluded that the phrase *old tottenham* is the most frequent phrase in the tweets. Similarly it works for all the other nodes. Figure 7.7 gives the graph after partition is done.

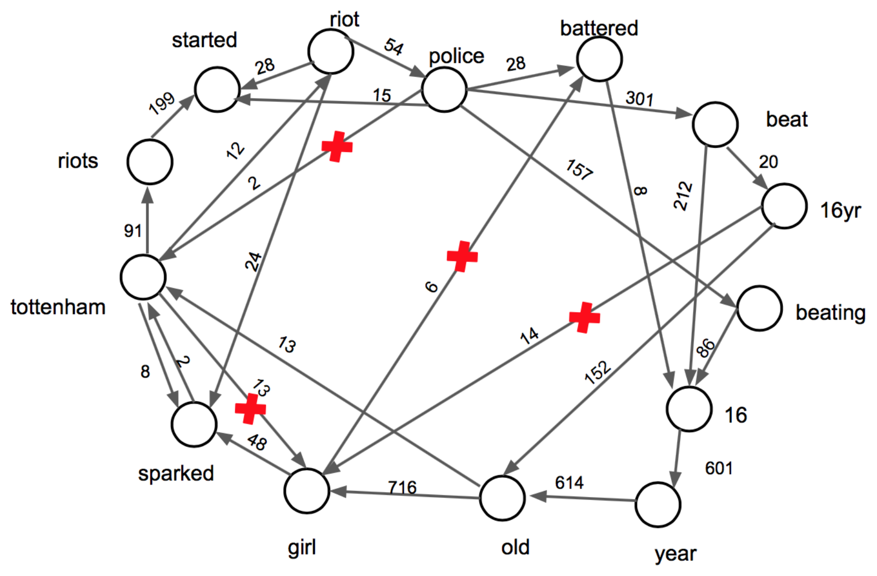


Figure 7.6: Steps of Graph Partition

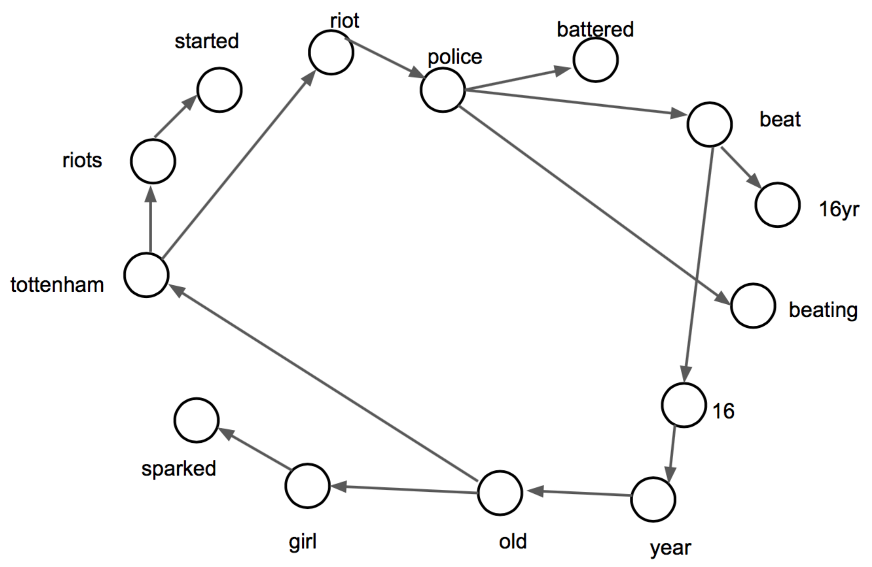


Figure 7.7: Graph after Graph Partitioning is performed

Therefore, a link between the phrases is established and the redundant phrases are removed by partitioning the graph formed from the phrases.

7.5.1 Removal of Cycles from Graph

The agenda of graph partitioning is to find the longest path from each graph which would eventually help in selection of tweets for each cluster. The detailed discussion about the methods involving the selection of tweets is done in the next chapter. However, it is to be noticed that the graph might still contain cycles due to overlapping words among the phrases. Therefore, in order to obtain the longest path from each graph the cycles are to be removed. The following steps are performed to remove the cycles from the graph.

1. Find all the cycles present in the graph.
2. For each cycle, find the list of nodes and the edges present in the cycle.
3. Remove the edge from the cycle which has the minimum weight.

The illustration given in Figure 7.7 provides a proper example. After the partitioning of the graph using the heuristic method [46], the graph contains a cycle consisting of the nodes *tottenham, riot, police, beat, 16, year, old, tottenham.* The minimum value at edge is 12 (refer to Figure 7.4) which is the frequency of the bigram phrase *tottenham riot.* Removal of this edge from the graph in Figure 7.7 generates a graph without any cycle. The illustration below is the graph obtained after the removal of cycle.

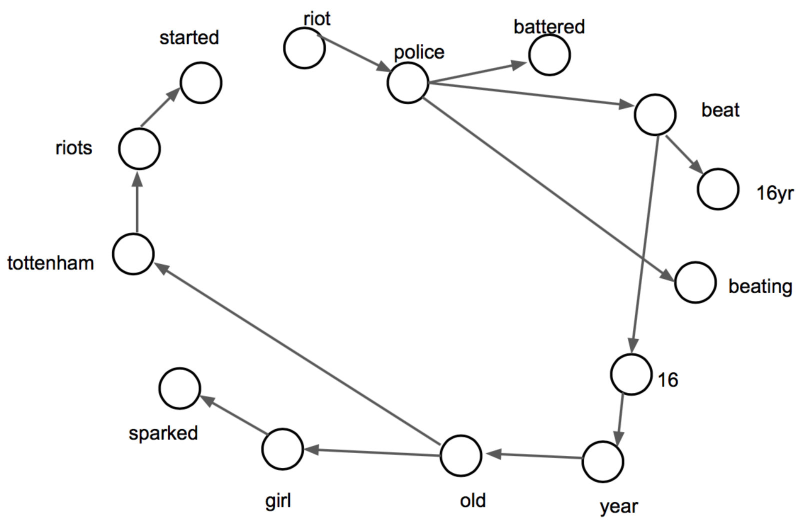


Figure 7.8: Directed Acyclic Word Graph for Phrases

Therefore, the word graph after partitioning and removal of cycles can now be termed as Directed Acyclic Word Graph. In the next chapter, we would focus on selection of the tweets for each cluster which would offer a summary of data set that was the basis of building the cluster.

Chapter 8

Selection of Tweets for Clusters

In this chapter, we would discuss the methods implemented to select few tweets from the original corpus for each of the clusters. These selected tweets for each of the clusters offers a summary of the data set that was the basis for building the cluster.

In the previous chapter we have discussed the methods to construct the directed word graph for phrases. In this chapter, the properties of the word graph are exploited to automatically select few tweets to summarize the clusters. In the first section of this chapter, a suitable method to find the longest path from the word graph formed from phrases is discussed. The second section includes a detailed discussion of the method developed and implemented to select tweets from the original corpus based on the longest path(s) obtained from phrase graph(s).

8.1 Introduction

Until now we have seen that bigram phrases are mined from the clusters. To study the pairwise connection between these phrases, a directed word graph is generated from the phrases. Thereafter, partitioning of the directed word graph formed from the phrases is done to obtain the set of phrases which are closely related. To explore this set of closely related phrases, longest path is to be determined from this directed word graph.

As the longest path of a graph contains the maximum number of nodes from the graph so it is used to select the relevant tweets as summary from the original tweet corpus.We will have a detailed discussion of these steps in the sections below.

8.2 Longest Path Problem

As already stated, the first step of the selection of tweets involves finding longest path of the directed word graph formed from the phrases. The nodes in the word graph of phrases are the words from the cluster and the directed edge from node u to v exists if u appears before v in the phrase uv. Therefore, the longer the path means larger number of words from the cluster are involved. Hence, the longest path calculated from phrase graph would give us the longest possible linkage of words in the cluster. There is a possibility that the longest phrase may contain all the words from the cluster, if every word present in the cluster is a part of a mined phrase. Clearly, there exists exactly one path from the given node to one of the leaf nodes in the graph, which has got maximum number of intermediate nodes along their path.

One of the most common ways to solve the longest path problem for a weighted directed graph with no cycles is by Flyod's Algorithm [79, 80]. Floyd's algorithm of shortest path problem between any two vertices of a graph is converted to the longest path problem by multiplying the edge weights with -1. The final result obtained is again multiplied by a negation to get the actual result. The direction of the edges are also reversed to get the actual longest path. The main drawback of this algorithm is that it compares all possible paths between each pair of vertices. Hence it is solved in polynomial time. The worst case complexity is $O(n^3)$ and space is $O(n^2)$, where n is the number of vertices. Since this approach involves polynomial time, therefore an efficient way to solve the longest path for the word graph formed from the phrases is by using Topological Sorting [50]. This method involving linear time is discussed in details in the next section.

8.2.1 Directed Acyclic Graph Approach

It is to be noted that partitioning of the directed word graph formed from the phrases has resulted into a number of connected components (refer Section 7.5.1). All these connected components are directed acyclic graphs. Therefore, this approach to find the longest path problem is applied to each of these connected components. As a result, one longest path is generated for each of the connected components of a phrase graph.

Longest Path problem for directed acyclic graph can be solved in linear time using dynamic programming[50, 81]. The method takes a directed acyclic graph as input and returns the longest path as output.

Solving the longest path problem is a two-step process.[50, 81]

1. A linear ordering of the nodes i.e topological sorting is done.
2. Process all the nodes in topological order to find the longest path.

Topological Sorting is performed on each of the connected components of a phrase graph. It is the linear ordering of the vertices of a graph such that for every directed edge $v_i vj$, from vertex v_i to vertex v_j, vertex v_i appears before vertex v_j in the sort ordering. Topological Ordering is not possible with graphs having directed cycles. This is because the sorting technique provides an order of processing of each vertex before its successor. Each DAG will have atleast one topologically ordering. As mentioned by Skiena, S. S. [81], depth first search (DFS) is used to sort the vertices topologically. DFS is applied to the directed acyclic graph and reverse order of the vertices being processed are returned as output of topological sorting. It is also solved in linear time $O(|V| + |E|)$, where V is the number of vertices and E is the number of edges in graph G = (V,E).

For example, topological sorting when applied to one the connected components of a phrase graph given in the illustration Figure 8.1 below results in the output provided in Table 8.1

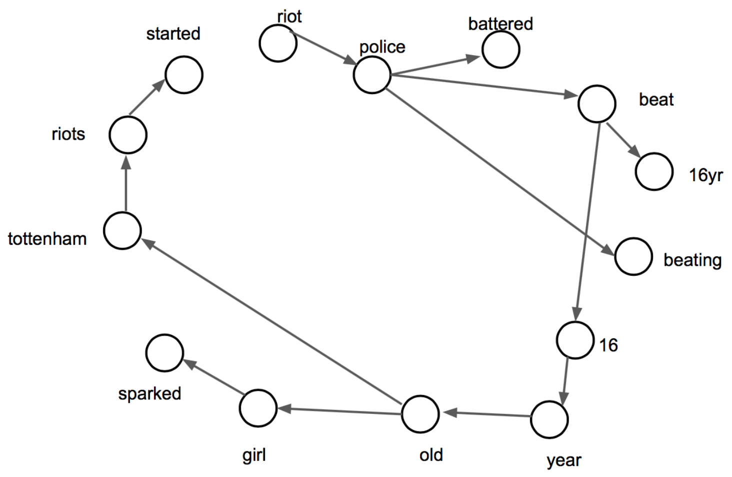


Figure 8.1: Phrase Graph

Topological Sorting	*riot, police, beat, 16, year, old, tottenham, riots, started, girl, sparked, 16yr, battered, beating*

Table 8.1: Topological Sorting

The second step of longest path problem involves processing of the topologically ordered vertices. The procedure for the same is as follows.

1. For each edge associated with each vertex in the topological order of the graph, update the distance of its adjacent vertex using the current node.

2. Return the set of vertices which forms the longest path.

The topologically sorted vertices mentioned in Table 8.1 when processed using the above mentioned steps generates the longest path given in Table 8.2

Topological Sorting	*riot, police, beat, 16, year, old, tottenham, riots, started, girl, sparked, 16yr, battered, beating*
Longest Path	*riot, police, beat, 16, year, old, tottenham, riots, started*

Table 8.2: Longest Path of the connected component subgraph in Figure 8.1

Topological sorting starts with the node with no incoming edge. The indegree of the node *riot* in the graph in Figure 8.1 is zero, i.e. it has no incoming edge. Therefore, the sorting starts from this node as no other node in the graph has indegree equal to zero. Therefore, there exists only one topological ordering of the graph. If there exists more than one node with indegree zero, then more than one topological sorting exists.

From the graph, it is easily verified that the set of vertices mentioned in the longest path of the graph consists of most of the nodes from the graph. Clearly, the total cost of path is also highest for this set of vertices.

The main objective behind finding the longest path from the phrase graph is to find the ordered list of words which is supposed to have occurred sequentially in the original tweets. This is due to the reason that the phrases are mined with the help of the original tweets (refer Section 6.2) and the phrase graph is generated considering the common words between phrases (refer Section 7.3.1). The longest path(s) obtained from each of the connected components of the phrase graph(s) eventually helps in selecting the tweets from the original corpus for summarization of the clusters. Also, the longest paths help in mining the most frequent phrases to be mapped onto the DBpedia articles for semantic enrichment of the clusters. The implementation details for semantic classification is discussed in the next chapter. However, the next section of this chapter includes description of the method to select the tweets from the corpus with the help of the longest path.

8.3 Tweet Selection

Summary of a topic refers to the synopsis of the topic or a brief restatement of the main points covered in the topic. As we know, that our model aims to provide the data journalists with the data with all the relevant information in a much compact way to make the most use of it. Therefore, to have a much smaller number of original tweets are selected to summarize the clusters is our goal in this step.

To accomplish this goal, we followed a simple approach to select some tweets which contains most of the words from the longest path of the connected components of the phrase graphs. It has been observed that the length of some of the longest paths are very small. Only those tweets are selected from the original corpus corresponding to these small longest paths if all the words present in the longest path is present in the tweet.

However, for a comparatively longer length of longest path a threshold value is defined depending on the length of the longest path obtained. The threshold is defined as floor value of the number of words in the longest phrase divided by two. If the number of words common to both the longest phrase and a certain tweet is greater than or equal to half the length of the longest path, we filter that tweet for a possible summary tweet.

The threshold is defined depending on the length of the longest phrase and not any other specific value because the length of the longest phrase will be different for different connected components of the phrase graphs. Clearly, the number of common words between the longest phrase and the tweets will be depending on this length.

Let us take the example of the same longest path Table 8.2 to find the the summary tweets. Five tweets corresponding to each longest path formed from the connected components of the phrase graph are selected from the original corpus as summary tweets.
Total number of words present in the path = 9

Therefore, threshold will be 4, the floor value.
The following tweet contains the eight words in common to the longest path (refer Table 8.2) so , considered as one of the potential summary tweet.

Tweet
RT @DeclanJMN: first hand account on BBC News: police beating a 16 year old girl sparked the riot. #tottenham"

Similarly, four other tweets are selected. Subsequently, the same process is to be followed for all the longest phrases obtained from different clusters.

Therefore, in this chapter solution to one of the research problem of this thesis is achieved. Until now, we have formed clusters from the tweet corpus, followed by phrase mining from the cluster words. In the next step we have formed a directed word graph from the mined phrases. Subsequently, graph partitioning helps us to find the most longest phrase which eventually provides the base for selection of tweets to summarize the clusters. However, the second research problem yet to be solved is to have a semantic enrichment of the clusters. The phrases mined from the clusters and the words from the clusters are to be mapped onto the corresponding most similar DBpedia article. We will discuss minute details about this mapping in the next chapter.

Chapter 9

Semantic Classification of Twitter Text

This chapter describes the methods developed and implemented for semantic classification of clusters by linking the phrases formed from the clusters (refer to Chapter 6) and words from the clusters to the abstract(s) retrieved from the corresponding to the most similar DBpedia article(s).

The chapter is constructed as follows: An overview of RDF structure and DBpedia is described in Section 9.1. Section 2 contains a definition of the aim of this step in our work. The implementation details and the challenges faced are discussed in later sections.

9.1 Introduction

To understand the methods discussed in this chapter a brief introduction about DBpedia is provided for better understanding.

DBpedia is community effort to extract structured knowledge from Wikipedia and makes the information available freely on the Web, using Semantic Web standards and Linked Data practices [14, 15]. Wikipedia, despite of being the most widely used free encyclopedia does not provide extensive query and search capabilities. Therefore, DBpedia project focused on the task of converting the Wikipedia information to a structured format with better querying capabilities. Wikipedia content is extracted and represented as a large multi-domain Resource Domain Framework better known as RDF dataset, such that Semantic Web techniques can be easily applied on it [Auer].

RDF is a model for processing metadata, data about data, i.e. data describing web resources. It enables effective operations among the applications which exchange machine understandable information on the Web. RDF allows data merging irrespective of the underlying schema or domain structure. The structure is domain neutral but capable of describing any domain knowledge. Moreover, it can adapt to any changes in the data schemes over time without affecting the consumers consuming the same data.[16, 17]

An end-to-end system that automatically extracts RDF triples from unstructured text has been proposed by Peter, E., et al. [18] The authors of this paper have used ontology mapping to map extracted triples to the DBpedia namespace.

The English version of the DBpedia knowledge base currently describes 6.2M things of which 4.6M have abstracts, 955K have geo coordinates, and 1.54M have depictions. In total, 5M resources are classified in a consistent ontology, which comprises 1.6M persons, 800K places (including 500K populated places), 480K works (including 133K music albums, 102K films, and 20K video games), 267K organizations (including 66K companies and 52K educational institutions), 293K species, and 5K diseases. The total number of resources in English DBpedia is 16.4M which, besides the 6.2M resources, include 1.3M skos concepts (categories), 7.1M redirect pages, 254K disambiguation pages, and 1.6M intermediate nodes. Localised versions of DBpedia is available in 125 languages[19, 20]

To access the DBpedia datasets, SPARQL endpoints are provided. A proper SPARQL query to the endpoint at `http://dbpedia.org/sparql` will return the exact information needed. This end point also supports extensions of SPARQL query language [15]. Therefore, querying proper RDF property from the DBpedia dataset would return appropriate results.

As explained by Bizer, C., Lehmann, J., et al. [22] the DBpedia knowledge extractor comprises of 11 extractors which process the Wikipedia content. The extractors are namely *Labels, Abstracts, Interlanguage Links, Images, Redirects, Disambiguation, External Links, Pagelinks, Homepages, Categories and Geo-coordinates.* These extractors contains the wikipedia details in form of predicates in the DBpedia RDF structure. Further we will see that this SPARQL endpoint will help us to extract the abstract of the most relevant DBpedia article.

9.2 Aim

Until now, we have generated clusters from twitter text, followed by mining of phrases from the words within the cluster. Thereafter, a phrase graph is generated from these mined phrases. Finally the graph is partitioned into components and the longest path problem is solved for each subgraph. The longest path plays two important roles in our study.

- It helps in summarization of the tweets as already seen in the last chapter. (refer Chapter 8 Section 8.3)
- It also helps in generating the most frequent phrases from each of the clusters which leads to the effective semantic classification.

This is because, the longest path of a phrase graph consists of vertices which are connected by edges having highest weights. Therefore, the vertices in the graph are the most frequently occurred words in the tweets. These form the most frequent phrases which eventually plays the key role in semantic classification of twitter text.

However, all the frequent phrases obtained from the phrase graph might not be relevant. We exploit the Wikipedia categories to determine the most relevant frequent phrases. We intend to associate the abstract of the most similar DBpedia article to each of these relevant phrases. Moreover, there might be some words in the cluster which do not form any frequent phrases. Yet, retrieval of the abstract of the most similar DBpedia article to these individual words in the cluster is important because it will provide a detailed information.

The intuition behind associating the most similar DBpedia article for these phrases as well as for words from the cluster, which are not part of any frequent phrases, is to have a semantic enrichment of the cluster with respect to the original tweets. For example, one of the clusters contains the words *tottenham* and *riots*. DBpedia abstract of the most similar article related to Tottenham infers that

Tottenham is an area in the London Borough of Haringey, in north London, England. It is situated 8.2 miles (13.2 km) north-north-east of Charing Cross.

Whereas, the corresponding most similar DBpedia article for *Riots* says

A riot is a form of civil disorder commonly characterized by a group lashing out in a violent public disturbance against authority, property or people.

However, *Tottenham Riots* i.e. the riots which broke out in Tottenham during summer 2011, is being referred in the original tweets. Hence, mapping of proper phrases such as, *Tottenham Riots*, formed from the words in the cluster to DBpedia article(s) would lead to correct semantic enrichment of the clusters as well as the tweets.

The remaining part of this chapter includes a detailed discussion of the methods for the extraction of abstract from the similar DBpedia articles for phrases and words and the challenges faced to achieve this goal.

9.3 Frequent Phrase Mining

Before delving into the details of DBpedia mapping for the words and the phrases from the clusters, frequent phrases are to be mined from the longest path. The longest path derived from the phrase graph includes the edges having maximum weight. As already stated earlier, the weight on an edge joining any two vertices u and v in the phrase graph depicts the popularity of the phrase u,v in the original tweets. Therefore, every bigram i.e. adjacent pair of vertices uv extracted from the longest path is a frequent phrase.

For example, longest path obtained from one of the phrase graphs is
riot, police, beat, 16, year,old, tottenham, riots, started
Bigrams formed from these longest path are the most frequent phrases of the corresponding cluster. The set of frequent phrases formed from this longest path is given in Table 9.1 below

Longest Path	*riot, police, beat, 16, year,old, tottenham, riots, started*
Frequent Phrases	*riot police, police beat, beat 16, 16 year, year old, old tottenham, tottenham riots, riots started*

Table 9.1: Frequent Phrases from Longest Path

In Section 9.5.3 a detailed explanation of method for identifying the set of most relevant frequent phrases from this set of frequent phrases is discussed. Moreover, the set of frequent phrases obtained from different clusters helps in finding out the individual words from the clusters which are to be mapped onto the DBpedia articles for effective semantic classification.

9.4 DBpedia mapping for words

It is to be noted that only a few words from each of the clusters are present in the longest paths derived from the corresponding phrase graphs. Therefore, there exists words in the cluster which does not form a frequent phrase. But, these words when mapped onto the most similar DBpedia article might provide a detailed information about the cluster. Hence, the words from the clusters which are not part of any phrases are considered as the set of words to be mapped onto their most similar DBpedia article(s). The most serious challenge in mapping of word(s) onto the most similar DBpedia article(s) is ambiguity of words.

9.4.1 Word Disambiguation

The tweet corpus used in our study consists of tweets written only in English. Hence our study includes methods to deal with ambiguity in English words. In English language, there are certain words which are homonyms. Homonyms are the words which has the same spelling, pronunciation but different meanings. Also, there might be some words which might refer to a movie or a song album or a place but the word has a different meaning when used in a sentence.

For example, the word *Riot*. This word is present in most of the clusters formed from the tweet corpus. Most commonly we refer the word as a civil disorder. However, a couple of famous movies are made with the name *Riot*. Also, there are songs, movie albums, music bands, video games with the same name as well. This creates an ambiguity.

All these ambiguous information corresponding to any word or topic is maintained by Wikipedia in a separate *Wikipedia Disambiguation* page. These pages contain the different meanings of homonyms as well as information about the other sources in which the word has been used in a different aspect. Similar disambiguation pages also exists in DBpedia for these ambiguous topics containing structured information from the corresponding Wikipedia articles.

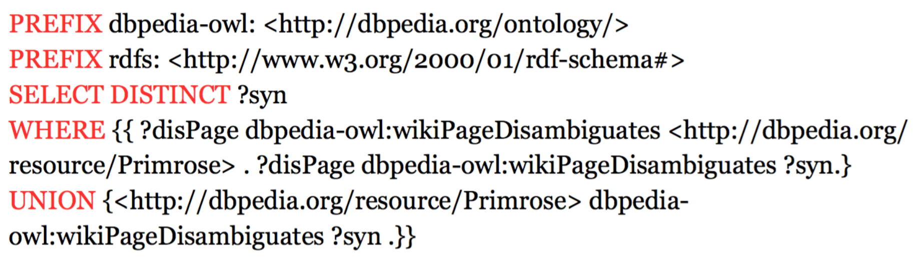
Bizer. C., Lehmann, J., et al. [22] mentions DBpedia RDF structure has a predicate dbpedia:wikiPagedisambiguates. In DBpedia, the disambiguation links of Wikipedia are represented using the predicate *dbpedia:wikiPagedisambiguates*.

We use SPARQL query to extract the disambiguation links for each of the words in a cluster. A standard SPARQL query consists of
Prefix declarations, for abbreviating URIs
Dataset definition, stating which RDF graph(s) to query
Result clause, which identifies what information is to be returned from the query
Query pattern, which specifies what to query for depending on the underlying dataset
Query modifiers, which helps in slicing, ordering, and otherwise rearranging the query results. An example of the structure of a standard SPARQL query is illustrated in Figure 9.1 below.[82]

```
# prefix declarations
PREFIX foo: <http://example.com/resources/>
...
# dataset definition
FROM ...
# result clause
SELECT ...
# query pattern
WHERE {
    ...
}
# query modifiers
ORDER BY ...
```

Figure 9.1: Structure of a SPARQL query

For example, the following query to the DBpedia dataset from SPARQL endpoint results in the list of all disambiguation links for the word *Primrose*, present in one of the clusters and not a part of any frequent phrases mined from that cluster. The query finds out all the URLs of the disambiguation articles , with the help of the DBpedia URL `http://dbpedia.org/resource/Primrose` from the predicate *wikiPageDisambiguates*.



```
PREFIX dbpedia-owl: <http://dbpedia.org/ontology/>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
SELECT DISTINCT ?syn
WHERE {{ ?disPage dbpedia-owl:wikiPageDisambiguates <http://dbpedia.org/
resource/Primrose> . ?disPage dbpedia-owl:wikiPageDisambiguates ?syn.}
UNION {<http://dbpedia.org/resource/Primrose> dbpedia-
owl:wikiPageDisambiguates ?syn .}}
```

Now, the challenge is to find the most suitable DBpedia page for these words from the list of disambiguation links in accordance with the clusters.

In the course of our study to find a suitable method for abstract extraction from DBpedia, it is found that every DBpedia article has a specific URL format given by

`http://dbpedia.org/page/XXX`

where XXX is the topic which is described in the corresponding page. For example, the DBpedia article for Germany has the url `http://dbpedia.org/page/Germany`.

It has also been noticed that the URLs of the DBpedia articles contain certain specific keywords hinting to the actual content of that corresponding article. This helps to disambiguate between homonyms and also different areas of usage of the same word.

For example, URLs for the DBpedia articles referring to *Riot* movies released in different years have the corresponding years and the word *film* mentioned in the URLs, `http://dbpedia.org/page/Riot_(1969_film)`. On the other hand, the URL for the DBpedia article corresponding to the word *Riot*, which means a civil disorder is `http://dbpedia.org/page/Riot`.

This characteristic feature of the URLs of DBpedia articles is exploited in our method to find the most similar DBpedia article corresponding to a word. All the words in the cluster refers to the same topic. Therefore it is likely that the words in the cluster hint at the description of the topic conveyed by the cluster. Therefore, in the next step we strip the last part of all disambiguation URLs obtained from the querying the DBpedia dataset at SPARQL endpoint. For example, the URL follows the format `http://dbpedia.org/page/XXX`, where XXX part of the URL consists the information about the corresponding article. Therefore this XXX part of the URL is stripped off. In the two URLs `http://dbpedia.org/page/Riot_(1969_film)` and `http://dbpedia.org/page/Riot` corresponding to the two articles related to *Riot*, *Riot_(1969_film)* and *Riot* are the *XXX* part respectively Hence, we have a list consisting of the stripped part of DBpedia URLs corresponding to each of the ambiguous words.

9.4.2 Comparison Approach

One of the significant approaches proposed in the past by Genc. Y., et al. [13] of finding most similar Wikipedia pages corresponding to tweets is done by analysing and comparing the categories associated between pages. The underlying graph structure is being exploited for the purpose.

We used a different technique of comparison in our study. Firstly, a vocabulary is formed from the unique words present in the clusters. Thereafter the stripped disambiguation URLs corresponding to ambiguous words are compared with the vocabulary formed. If there are some words in the cluster which are common with a part of the stripped URL we assign a score for this URL. If this score is above a threshold these potential URLs are shortlisted for the corresponding word from the cluster.

Finally, abstract of the DBpedia articles corresponding to the shortlisted URLs for a particular word in the cluster are extracted. The vocabulary formed from the cluster is now compared with the extracted abstract from DBpedia. Eventually, we assign the abstract of DBpedia article to a certain word in the cluster for which the number of common words and the frequency of the occurrences of the common words between the abstract and the vocabulary is highest.

9.5 DBpedia mapping for Phrases

First and foremost task is to find the most relevant frequent phrases. Next, we find the most similar DBpedia page for each of these phrases by exploiting Wikipedia articles. The reason behind this is DBpedia is the structured content of Wikipedia and also there might not be a relevant page in DBpedia for a phrase whereas there is one in Wikipedia. In order to find the relevant phrases, it is important to consider the occurrence of time period of the events.

9.5.1 Date and Year Retrieval

Day to day happenings in and around the world comprises of a huge section of the tweets. Moreover, the data used in this study is related to riots. World History has witnessed riots at different periods of time. Therefore, it is evident that date and year will play an important role in our analysis. In our experiments we restrict our time search space and use the extracted date and year from the original tweets.

9.5.2 Hashtagged Words

Another important aspect to be noted here is that there are some hashtagged words present in the clusters. Normally, the twitter users clubs in more than one word to form a hashtagged word. For example, *#londonriots*, *#londonzoobreakin* etc. An Apache2 licensed module for English word segmentation[1] is used in our study to segment the hashtagged words present in the cluster. The reason behind this segmentation is that the hashtagged words are the most frequent words in the corpus so they undoubtedly forms the relevant frequent phrases.

9.5.3 Relevant Frequent Phrases

Mining of the frequent phrases from the longest path has already been discussed earlier in this chapter (refer to Section 9.3). However, it is to be considered that some of these phrases when considered individually and mapped to the corresponding most similar DBpedia article might not reflect relevant information with respect to the original tweets. Apart from this, some of these phrases might not convey any proper meaning when considered individually or may not refer to any named entity. Therefore, no article dedicated to these phrases in the reference knowledge base i.e. Wikipedia or DBpedia is to be found. Eliminating these two types of phrases from the set of frequent phrases would result into a set of most relevant frequent phrases.

We exploit Wikipedia categories to find the relevant phrases. Wikipedia categories refer to groups of articles on the same topic. These categories further have other categories listed as sub categories. Therefore, the categories of the Wikipedia pages are linked in form of a graph structure. Genc. Y.,

[1] PyPI WordSegment 0.6.2 `https://pypi.python.org/pypi/wordsegment`

et al. [13] introduced a method to find the most relevant Wikipedia page corresponding to a tweet by capturing this network graph structure upto five levels followed by semantic distance computation. In our method instead of capturing the subcategories via the graph structure associated with a Wikipedia page upto a certain level, list of all the categories associated to the Wikipedia page corresponding to all the phrases are extracted. In the next step, these lists of categories for the phrases are compared against each other to find the similar phrases. The phrases having common categories are considered to be similar.

Wikipedia defines some of it categories as Maintenance Categories. These maintenance categories groups articles in which source of information is missing such as, external links referred in these articles are dead, articles that need external additional references or the article is a stub one i.e. article is too small to provide any rudimentary information. Therefore, we ignore these maintenance category articles while comparing the category lists. The steps can be summed up as follows:

1. Find the bigrams i.e. most frequent phrases from the longest path. (refer Section 9.3)

2. For each of the frequent phrase, check if there exists a valid Wikipedia page.

   (i) If a valid Wikipedia page does not exists, discard the phrase.

   (ii) If a valid Wikipedia page exists, extract the list of all the categories corresponding to the page.

   (iii) If there is Wikipedia disambiguation page for the phrase, we mark the phrase as a disambiguated valid frequent phrase.

3. Discard if any Wikipedia Maintenance Categories are present in the list of categories of all phrases.

4. Compare the list of categories obtained for each phrase with the list of categories of other phrases.

   (i) If there are common categories between the Wikipedia pages, we represent the corresponding phrases in pairs.
   For example, the phrase *London Zoo* has Wikipedia categories in common with *London Eye* and *Manchester Riots*, the pairs of phrases with common categories formed in this step are *(London Zoo, London Eye)* and *(London Zoo, Manchester Riots)*

(ii) If there are no common categories between the phrases, a score calculation is done to check if the phrase is relevant.

Until now, using the above mentioned steps, a set of pairs of phrases having common Wikipedia page categories and a set of phrases having no categories in common are obtained. In the next step, we form a graph with the phrases having common Wikipedia page categories to analyse the closely related pairs of phrases.

9.5.3.1 Graph

Each node of the graph represents a phrase. There is a directed edge from phrase u to phrase v, if the pair *(u,v)* have shared common Wikipedia page categories. The weakly connected components of a directed graph are found. This is because if all the edges of a directed weakly connected graph are replaced by undirected edges it will result into a connected graph. Therefore, all the weakly connected components from this graph will result into sets of phrases which are connected to each other. The set of connected components of length greater than two are considered as the most relevant frequent sets. This is because more number of phrases have common Wikipedia page categories and therefore they are likely to one another. A score calculation is done to check the relevance of the phrases included in the remaining weakly connected components of the graph of length two.

9.5.3.2 Score Calculation

An approach of finding the associated Wikipedia page is proposed by Genc. Y., et al. [13] in which a word set is generated based on the text of the tweet. A score is assigned to each potential Wikipedia page corresponding to a tweet based on the number of occurrences of the words in the word set. In our work, to mine the relevant phrases from the the list remaining phrases in the weakly connected graph (refer to Section 9.5.3.1) and the phrases having no categories in common we have used a similar approach. A bag of words is generated from the words in the cluster and is considered as reference vocabulary. Now, if the number of common words in the vocabulary and the Wikipedia page of a phrase is greater than a threshold we consider the corresponding phrase as a relevant frequent phrase for Semantic Classification. Also, while computing the number of common words, we also consider occurrence of the date and year retrieved from the tweets (refer to Section 9.5.1) in the Wikipedia page of the same phrase.

Therefore, a list of relevant frequent phrases are obtained from the longest path which leads to the effective semantic classification of the clusters formed from the tweets.

9.5.4 Phrase Disambiguation

The mined relevant frequent phrases might refer to more than one event or topic. In order to find the correct DBpedia page with respect to the corpus, SPARQL query is used to extract disambiguation of the phrases from DBpedia. If no such disambiguation exists we extract the corresponding abstract of that article and assign it to the corresponding phrase.

However, if there is a disambiguation, we take the help of Wikipedia articles. We check for the corresponding Wikipedia article of disambiguation and extract the list of all the ambiguous Wikipedia pages corresponding to that phrase. Further, we extract the abstract from all these listed Wikipedia disambiguation pages. Subsequently, we compute a same score calculation as mentioned in Section 9.5.3.2 of finding the common words between the vocabulary and each Wikipedia page. We assign the page having the most number of common words as the most relevant page corresponding to the phrase. This is due to the fact that DBpedia does not always provide the disambiguation pages for ambiguous phrases unlike for ambiguous words. But, Wikipedia provides with a list of disambiguation articles for ambiguous phrases. However, the date and year plays a vital role in disambiguating between events happened at the same place in different time period.

9.6 Link between different clusters

Clusters formed from the tweets are group of words depicting a certain incident or topic. However, there might be some common links between the clusters. As already mentioned, our study is based on twitter data related to London riots in the year 2011. Therefore, there is a possibility of overlapping topics between the clusters.

We follow a simple step to see if there is any missing link between the clusters. In the previous section we have seen that abstract from the most similar DBpedia article is extracted and assigned to phrases. Longest Common Subsequence method [21] is applied in pairs between the abstract of phrases from all the clusters. If a common subsequence is noticed, we conclude that the clusters have a underlying connection. The following steps are followed for the same.

Steps to establish links between clusters

---

1. For each phrase p formed from cluster C_i:

   (a) Perform Longest Common Subsequence(LCS) Algorithm with the abstract of all the phrases formed from cluster C_j.

2. If a subsequence is encountered, it is concluded that clusters C_i and C_j is related.

It is to be noted that stop words are removed from the DBpedia abstracts of the corresponding phrases before LCS is performed. The Longest Common Subsequence(LCS) problem is discussed in the section below.

9.6.1 Longest Common Subsequence

Longest Common Subsequence problem, as the name suggests, finds the common subsequence between two string of sequences. Unlike substring problem, common subsequence need not occupy consecutive positions in the sequence. A mathematical study for this approach is done by Chvatal.V., et al. [21]. Dynamic Programming approach is used to solve this problem. This is because the problem comprises of optimal substructure and overlapping sub problems. The problem can be broken down into multiple simpler sub problems until the solution becomes trivial. Again the solution of lower level sub problems is used by the higher level sub problems. A recursion approach is used to solve the problem.
An example is given below to illustrate how the longest common subsequence problem works.
Sequence_1 = $ABCDGPHRO$
Sequence_2 = $ADECFIH$
Longest Common Subsequence = ADH

For example, in our study *Manchestor Riots* is a relevant frequent phrases from one of the clusters whereas *UK Riots* is a relevant phrase from another cluster. Longest Common Subsequence Algorithm is applied to the extracted abstract for both the phrases and it has been noticed that the abstract for both the phrases are same. Therefore, it is concluded that the two clusters which consists of *Manchester Riots* and *UK Riots* respectively have an underlying connection.

Therefore, the semantic classification of the phrases and the words from the cluster is done by assigning the abstract of the most similar DBpedia article to them. Further, an approach is proposed to link between the clusters with the help of the abstracts from DBpedia.

In the next chapter we evaluate the pipeline of these techniques using human judgements.

Part III

Results and Conclusion

Chapter 10

Experiments and Evaluation

In this chapter, we discuss the experimental setup and evaluation metrics used to evaluate our work.

As we have seen, a pipeline of methods helped in achieving the two goals of this thesis. The first goal is to select a small number of tweets as summary for each cluster from the original tweet corpus of 2.6 million tweets. The second goal is to provide semantic enrichment of the cluster by associating the most similar DBpedia pages to the words from the cluster and also to the most frequent relevant phrases mined from the clusters. The methods developed and implemented to achieve these two goals have been discussed in previous chapters (refer Chapter 4 to Chapter 9). In this chapter, the qualitative and quantitative results obtained from the evaluation metrics are analyzed to evaluate the methods used to achieve these two goals.

10.1 Evaluation of Summarization of Tweets

As discussed earlier, the tweet corpus used in our study comprises of the news related to seven most popular events which went viral on Twitter during the London Riots, 2011. These events are

- Rioters attacked London Zoo and set the animals free.
- Police beat a 16 year old girl.
- London Eye is set on fire.
- Army deployed in Bank
- Rioters attacked Birmingham Children Hospital.

- Rioters broke into McDonald's and cooked their own food.
- Miss Selfridge shop was set on fire.

As there are seven underlying events, it has been observed that seven different clusters are formed from this tweet corpus. In the previous chapters, we have discussed methods to select only five tweets from the original tweet corpus for each of the cluster which summarizes the cluster. However, no standard method is available to judge the relevance of this small collection of retrieved summary tweets. One commonly used method used by the researchers for summary evaluation is to check the summary against some predefined parameters such as content, fluency and grammatical correctness. This method of evaluation is known as Intrinsic Evaluation[48, 83]. However, in our evaluation method fluency and grammatical correctness parameters are discarded because the most relevant five tweets are selected from the corpus depending on the longest phrase formed from each cluster (refer to Chapter 8). In intrinsic method, manual summaries are generated from the original corpus and are compared against the automated summaries [83].

10.2 Manual Selection of Summaries

The data set used in our study comprises of annotated tweets labelled for each of the seven events. In our work, manual summaries are generated by selecting 10 tweets from this annotated tweet corpus for each of the seven events mentioned above and compared against the five automatically selected tweets by our method. Since tweets are selected from the corpus as automated summaries therefore the manual summaries are also generated from the tweets from the corpus.

The tweets that have been retweeted by a large number of users are clearly the most popular tweets. Therefore, while generating the manual summaries from the corpus for each of the seven events, these popular tweets are included in the summary. The intuition behind selecting these popular tweets is that it describes the event properly and contains the most relevant information about the event. Hence, firstly these popular tweets corresponding for each of the seven events are included in their respective manual summaries. The remaining tweets for the manual summary are selected randomly from the same annotated corpus corresponding to the event. These two summaries are then compared against each other.

10.2.1 Comparison Study

Each of manually selected tweets summaries are compared against each group of automatically selected five tweets. The metric used for the comparison is the ratio of the words common to both the manually selected tweets and automatically selected tweets to the number of words present in the manually selected summary.

$$Comparison(tweet1, tweet2) = \frac{\texttt{\#common words}}{\texttt{\#words in tweet2}} \tag{10.1}$$

where tweet1 is the tweet from automatically selected set of tweets i.e. from automated summaries and tweet2 is the tweet from the manually selected set of tweets i.e. from manual summaries. The ratio is likely to give a good coverage of the comparison of the words in the automatically selected tweets to the manually selected tweets. Figures below illustrates the comparison study of the automated summary and the manual summary for each of the seven events.

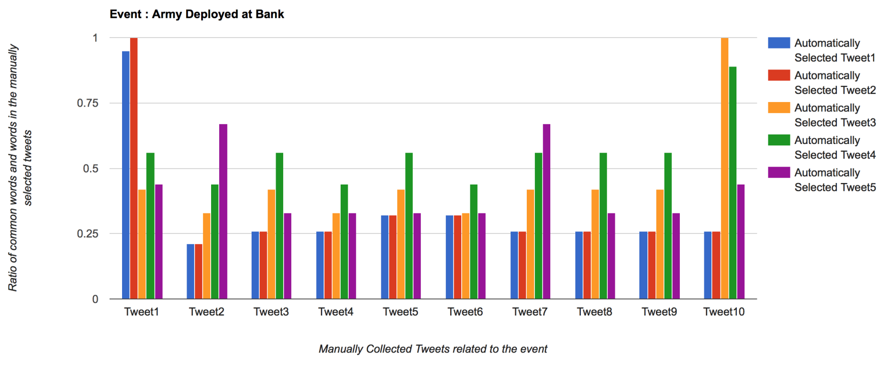


Figure 10.1: Army Deployed at Bank

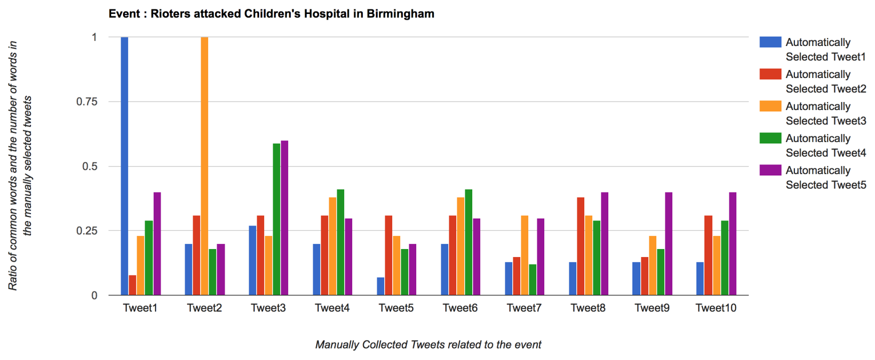


Figure 10.2: Rioters attacked Birmingham Children's Hospital

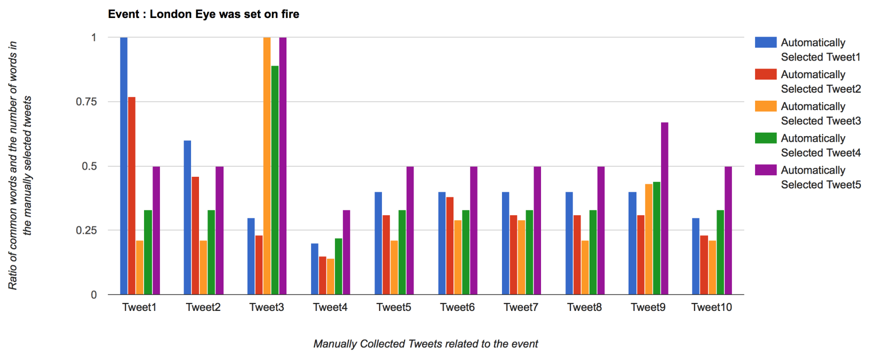


Figure 10.3: London Eye was set on fire

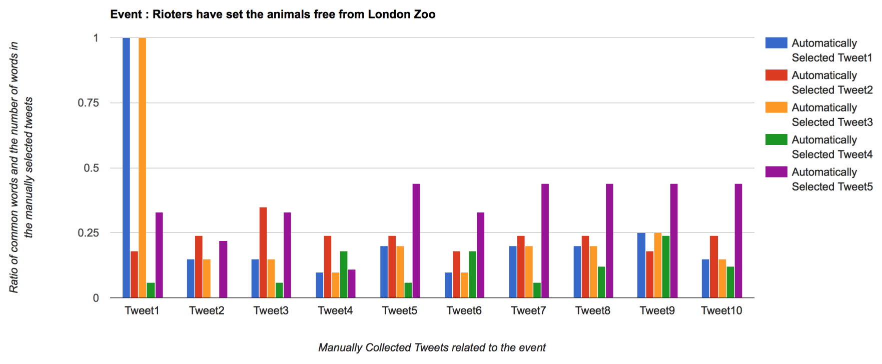


Figure 10.4: Rioters attacked London Zoo

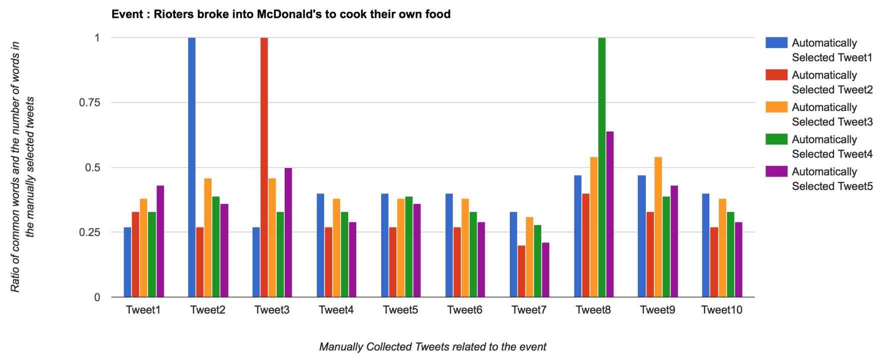


Figure 10.5: Rioters broke into McDonald's and cooked their own food

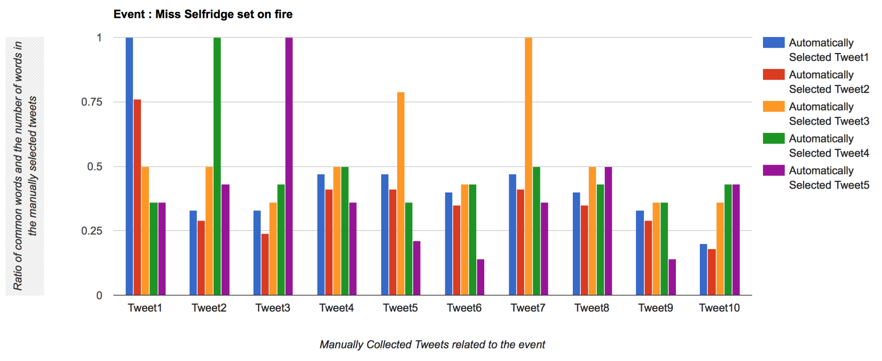


Figure 10.6: Miss Selfridge shop was set on fire

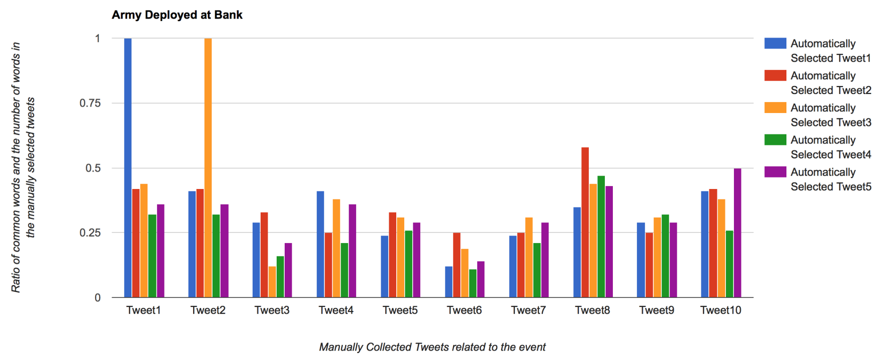


Figure 10.7: Police beat a 16 year old girl

Therefore, a comparative study of the 10 manually selected tweets for each of the events is done with the corresponding five automatically generated tweets for each of the events. The manually selected summary tweets are different for each of the seven events. Also the automatically selected summary tweets depicted in each of the graphs corresponds to the tweets related to the events mentioned in each of the graph. The comparison ratio is plotted along the Y-axis. It is to be noticed from the graphs that the comparison ratio is equal to 1 or closer to 1 in many cases. This concludes that the automatically selected summary of tweets has a good coverage of the manually selected tweet summary hence comprises of the popular tweets from the corpus. Therefore, it can be inferred that the automatically selected tweets are relevant to the events. Also it is to be noticed there are cases where the coverage is less for a particular automatically selected tweet with respect to a certain manually selected tweet for a particular event. But the coverage of the same tweet is considerably better with another manually selected tweet. Also, it is noticed that the coverage of most of the tweets specific to an event with respect to the manually selected tweets for the same event is almost similar.

We have also carried out the comparison of the automatically generated tweets of one event with manually generated summary tweets for other events. It is clearly noticed that the coverage of words between them is significantly less. The coverage of words in this case, only includes certain keywords common to all the events because these seven events are part of London riots, 2011.

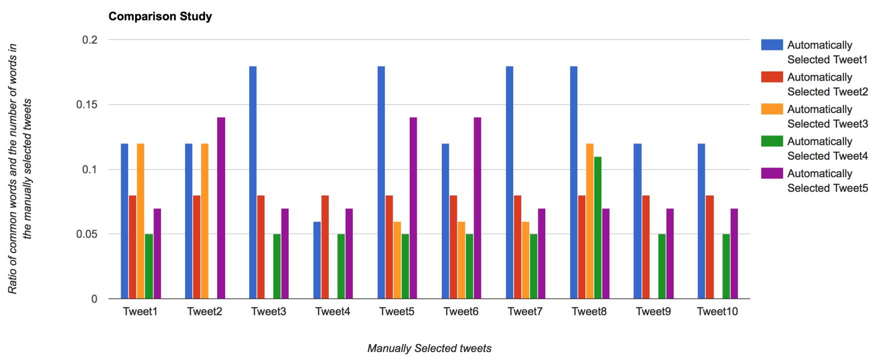


Figure 10.8: Comparison Study of Manual Summary for the event *Army deployed at the bank* with the Automated Summary for the event *Police beat a 16 year girl*

It is to be noted that in the above illustration the coverage of the words in the manual summary with respect to the automated summary is very less. The highest comparison ratio calculated using Equation (10.1) is 0.18, which is much less compared to the ratio when a set of automatically selected tweets are compared to the set of manually selected tweets of the same event.

Therefore, it can be concluded that tweets selected automatically perfectly summarizes the clusters.

10.3 Manual Metric for Content Evaluation

A second evaluation method of content of the automatically selected tweets is also performed. This metric is known as Manual Metric used during DUC 2002[48, 83]. Volunteers are asked to evaluate the content of the summaries with respect to the reference event. Google Forms is used to conduct the study. 10 volunteers are asked to evaluate the automatically selected tweets with respect to the events on a scale of 1 - 5. Scale 1 denotes Strong Disagreement, scale 2 is for disagreement, partial agreement and disagreement is denoted by scale 3, scale 4 and scale 5 denotes the agreement and the best match respectively. The graphs below illustrates that the volunteers agreed to most of the automatically generated tweets with respect to the events. The x-axis denotes the evaluation scale whereas the y-axis denotes the percentage of user response.

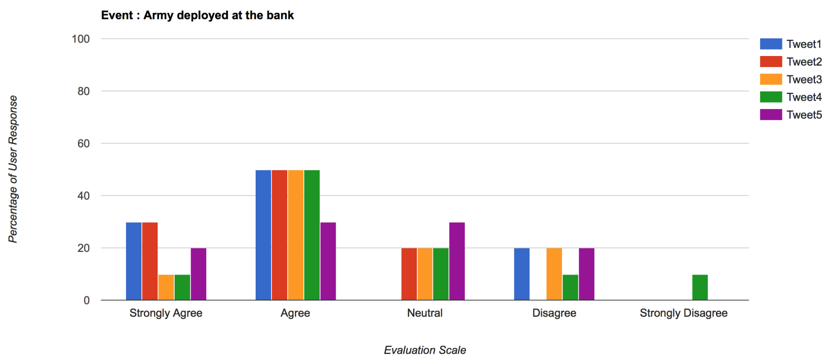


Figure 10.9: Army Deployed at Bank

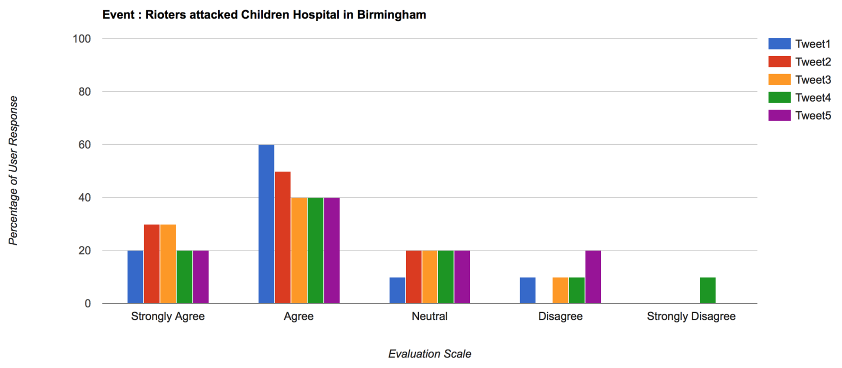


Figure 10.10: Rioters attacked Birmingham Children's Hospital

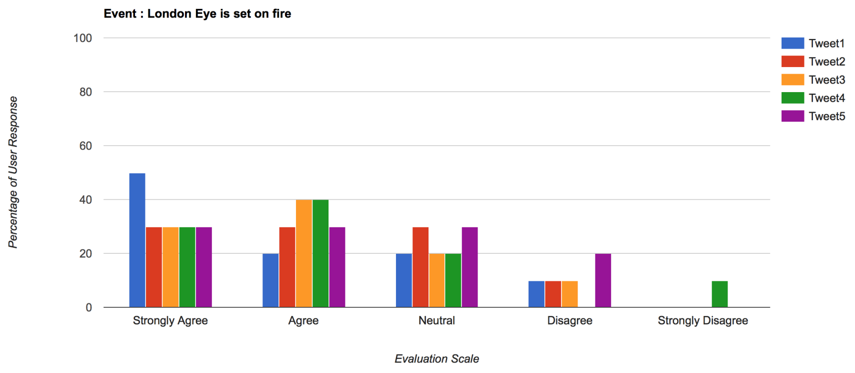


Figure 10.11: London Eye was set on fire

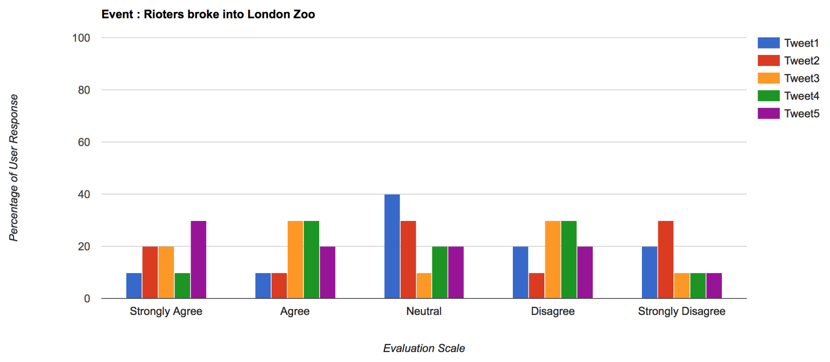


Figure 10.12: Rioters attacked London Zoo

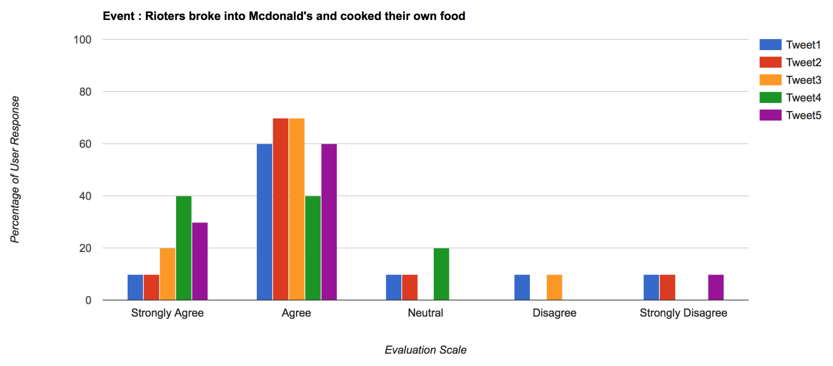


Figure 10.13: Rioters broke into McDonald's and cooked their own food

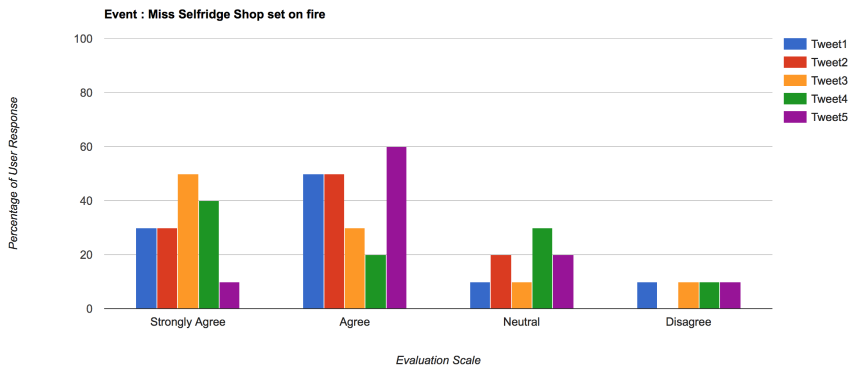


Figure 10.14: Miss Selfridge shop was set on fire

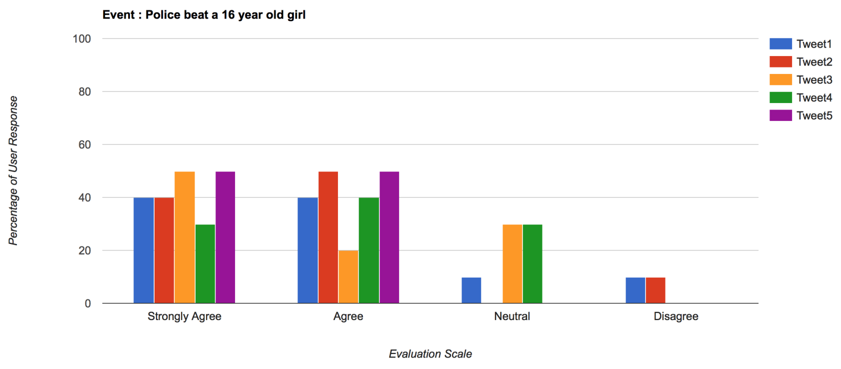


Figure 10.15: Police beat a 16 year old girl

Clearly, the higher percentage of user response corresponding to the scale *Strongly Agree* and *Agree* in the evaluation scale of tweets depicts that automatically selected tweets are relevant to the events. It is also to be noted in the graphs below that *Strongly Disagree* parameter of evaluation scale is missing in most of the graphs. This is due to the fact that none of the volunteers have marked the automatically selected summary tweets corresponding to that event as a strong disagreement.

However, it is to be noted that mining of phrases from the cluster plays a vital role in selecting the tweets as summary. We use a manual metric method followed by a score calculation to evaluate the importance of the phrase mining in our work. The setup for this evaluation method comprises of the following steps.

1. From each cluster, three sets of words are selected randomly.
2. Each set of words are considered as the longest phrase and five tweets are selected for each such set of words from the original corpus using the method discussed in Chapter 8 Section 8.3
3. The relevance of these retrieved tweets with respect to the clusters are judged by a group of five volunteers on a scale of 1 to 5. Scale 5 represents strong agreement and scale 1 represents strong disagreement as before.
4. An average acceptance score is determined for each cluster.
5. A similar acceptance score is calculated for the tweets selected automatically in our method.
6. For each cluster, the average acceptance score calculated for the tweets selected from the random words is compared with the automatically selected tweets. The higher the average acceptance score, the better the tweets selected.

The average acceptance score is defined as the sum of the response values corresponding to the selected tweets for each cluster divided by total number of user responses. For example, five tweets are selected for the set of words selected randomly from a cluster. A group of five volunteers are asked to mark the relevance of the tweet in a scale of 1 - 5 with respect to a specific event. The user responses are given in the table below.

	Tweet 1	**Tweet 2**	**Tweet 3**	**Tweet 4**	**Tweet 5**
User 1	1	1	1	2	2
User 2	2	1	1	2	2
User 3	2	1	1	2	2
User 4	4	5	3	4	3
User 5	4	5	3	4	3

Table 10.1: User Response to a set of tweets from randomly selected words corresponding to the event *London Eye is set on fire*

The corresponding acceptance score is given by
Average Acceptance Score $= \frac{\texttt{sum of the responses}}{\texttt{total number of responses}} = \frac{61}{25} = 2.44$

This method of selecting tweets from some random words from the cluster serves as the baseline to evaluate our work. However, the average acceptance score for our method corresponding to the same event is 4.68. The table below shows a comparison of the average acceptance score between the baseline method and our method of phrase mining.

Events	**Baseline Method**	**Phrase Mining Method**
Rioters attacked London Zoo	3.8	4.4
Rioters attacked Children's Hospital	3.9	4.3
London eye is set on fire	2.44	4.68
Police beat a 16 year old girl	4.56	4.6
Miss Selfridge shop set on fire	4.36	4.68

Table 10.2: Comparison study of the Average Acceptance Score between the Baseline method and our method

However, it has been noticed that tweets from all the seven events are not retrieved using random words. No tweet has been retrieved by any of the three sets of random words for the events *Army deployed at the bank* and *Rioters broke into McDonald's and cooked their own food.* Thus, the tweets selected corresponding to two events are not considered for comparison. However, only one tweet for the event *Rioters attacked London Zoo* and two tweets for the event *Rioters attacked Birmingham Children's Hospital* are retrieved. Five tweets per event are retrieved for each of the remaining events. The number of words in each set of random collection of words for each cluster is equal to the average length of the longest phrases obtained from these clusters.

The difference in the acceptance score between the baseline method and our method for the events *Police beat a 16 year old girl* and *Miss Selfridge shop set on fire* is very small. This is because we have mined bigram phrases from the clusters, therefore while selecting words randomly from the cluster there is a possibility of selecting words which are part of most frequent bigram phrases. However, the difference being significant for the other events and the fact that no tweets or lesser number of tweets being selected concludes that mining of phrases is important for our method.

Hence, we conclude that the method proposed in this work for selecting a few tweets from a huge corpus of tweets for each cluster is efficient and provides all the relevant information. Hence, the data analysis for the data journalists becomes easier because for each event only five tweets containing the most relevant information is provided to them.

10.4 Evaluation of Semantic Enrichment

In the previous chapters, it has been discussed that the most frequent relevant phrases mined from the clusters are mapped to the corresponding most similar DBpedia article for semantic enrichment of the clusters. As already mentioned earlier, clustering of tweets results into groups of words which are similar to each other. However, these words present in the clusters are individually mapped to the corresponding most similar DBpedia article may convey a different meaning than the one conveyed in the original tweets. Therefore, a necessity to mine phrases from the clusters arises. Bigram phrases are mined from the words in the cluster with the help of the original tweets. Not all these bigram phrases formed are frequent and relevant for DBpedia mapping which would lead to semantic enrichment of the clusters. In order to obtain the most frequent phrases, word graph is constructed from these bigram phrases. Partitioning of this word graph formed from the phrases provides us with the most frequent phrases. However, to mine the most relevant frequent phrases Wikipedia categories are exploited. The reduction in the number of phrases in different stages of our method to detect the set of most relevant frequent phrases is well depicted in the Figure 10.16

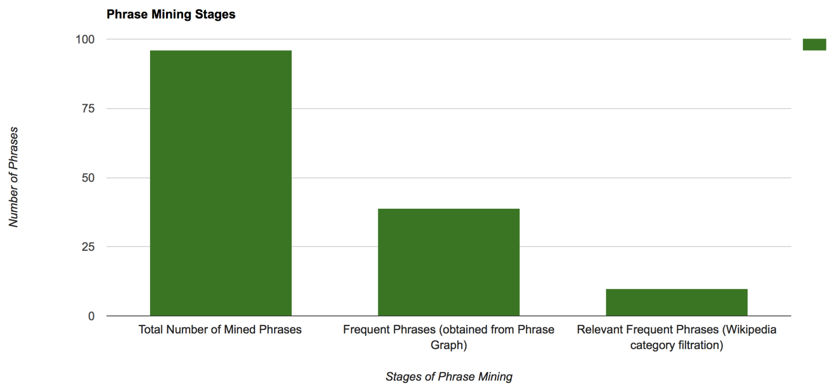


Figure 10.16: Reduction in the number of phrases

Wikipedia categories are exploited to detect the most relevant frequent phrases for the clusters whose mapping to the corresponding most similar DBpedia article would provide a semantic enrichment of the cluster. The pie chart in Figure 10.17 illustrates the fraction of relevant frequent phrases marked in blue concluding the necessity of mining of the relevant phrases.

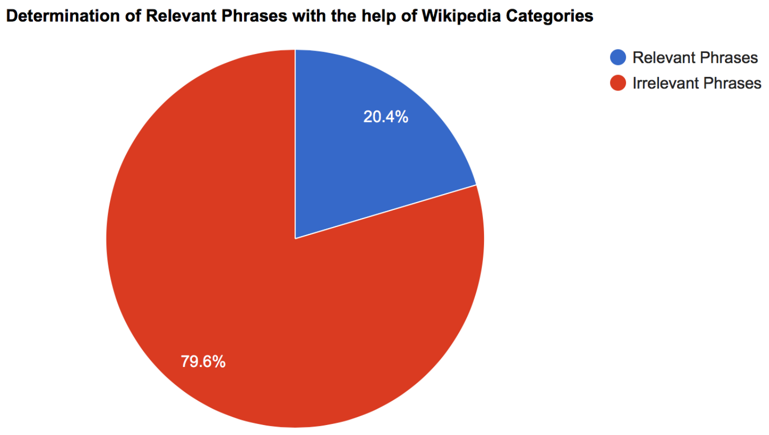


Figure 10.17: Pie Chart showing the fraction of Relevant and Irrelevant Frequent Phrases

The same approach of manual metric is again used to evaluate the semantic enrichment of the clusters. To conduct this evaluation procedure, individual words from the clusters and the frequent relevant phrases are mapped onto the corresponding most similar DBpedia article. This set of mapping of words and phrases to DBpedia articles were provided to the same group of 10 volunteers. The volunteers were asked to evaluate the relevance of DBpedia mapping of the words and phrases with respect to the seven incidents mentioned in the original tweets. The illustration in Figure 10.18 below clearly depicts that the relevant phrases are mapped to the most similar DBpedia articles with respect to the original tweets compared to the individual words.

It is clear from the graph that Miss when mapped separately to the most similar DBpedia article is irrelevant as Miss Selfridge shop is being mentioned in the original tweets. Similarly, it follows for the other words and the corresponding phrases.

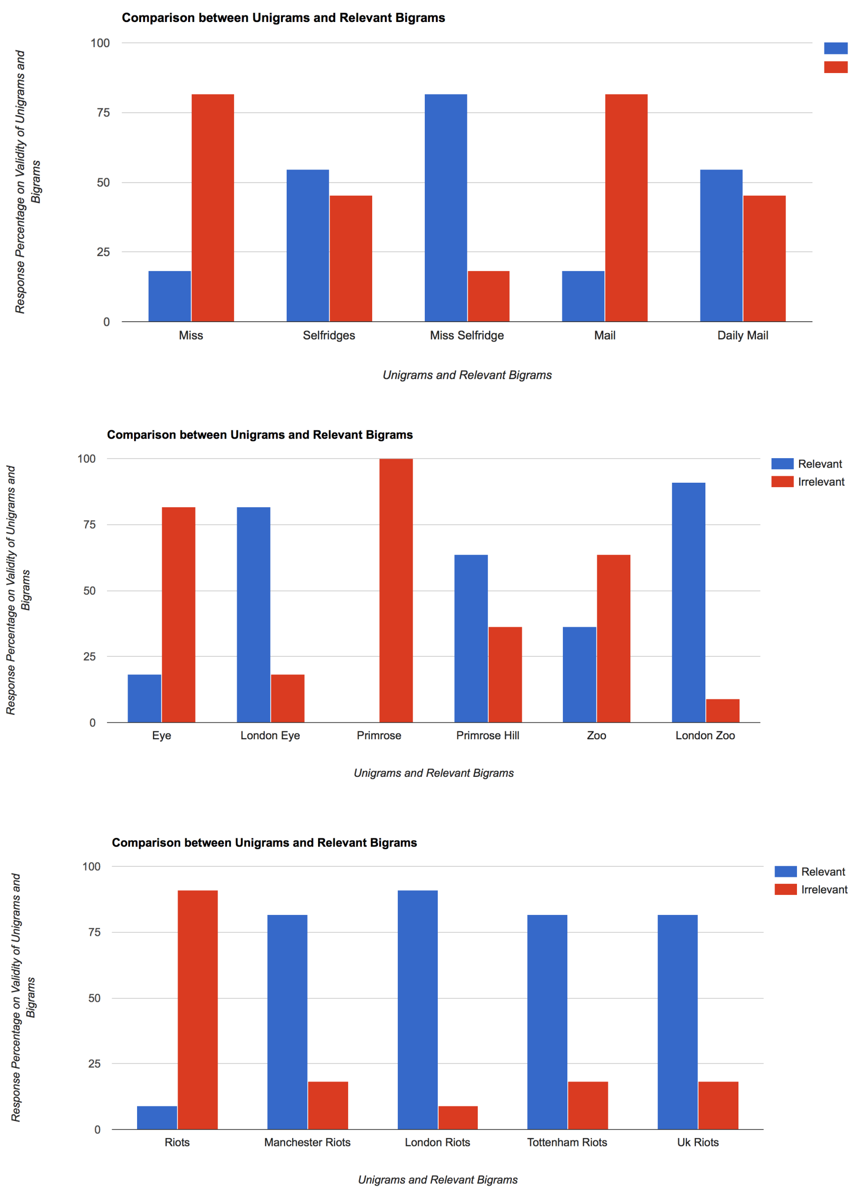


Figure 10.18: Comparative study of relevance of Individual Words from the Cluster and the Frequent Phrases formed from the Cluster

As discussed in Chapter 9, the words from the cluster which are not part of any phrases are also mapped to the most similar DBpedia article with respect to the original tweets. The mapping of the DBpedia articles to unigrams includes handling of the ambiguity. As already stated we the URL feature of DBpedia articles is exploited followed by a score calculation with respect to the original tweets is done to identify the most similar DBpedia article(s). The table below shows the reduction in the number of ambiguous DBpedia articles for a set of words after the URL handling of the ambiguous articles is performed (refer to Chapter 9 Section 9.4.1 for the method).

Words	**Number of Ambiguous URLs**	**Number of URLs after URL Disambiguation**
god	45	1
street	70	2
person	6	1
bank	36	3
protect	6	1
army	2	1
news	49	2
eyewitness	52	1
guy	22	2
brixton	20	1
video	88	1
tiger	99	3
pile	20	1
primrose	27	1
food	8	2
head	56	3
selfridges	13	1
store	6	1
hill	28	3

Table 10.3: Reduction in the number of ambiguous URLs after URL Disambiguation is performed

Therefore, it is evident that URL disambiguation step has helped in reducing the number of ambiguous URLs in large scale. However, a manual metric evaluation has also been carried out to evaluate the mapping of the most similar DBpedia articles to a set of unigrams from the clusters with respect to the events mentioned in the original tweets. Figure 10.19 below provides an illustration for the same. It shows that most of the unigrams are considered to be irrelevant with respect to the original tweets which is correct. The ones which are marked relevant are the words which are being mapped to the most similar DBpedia article.

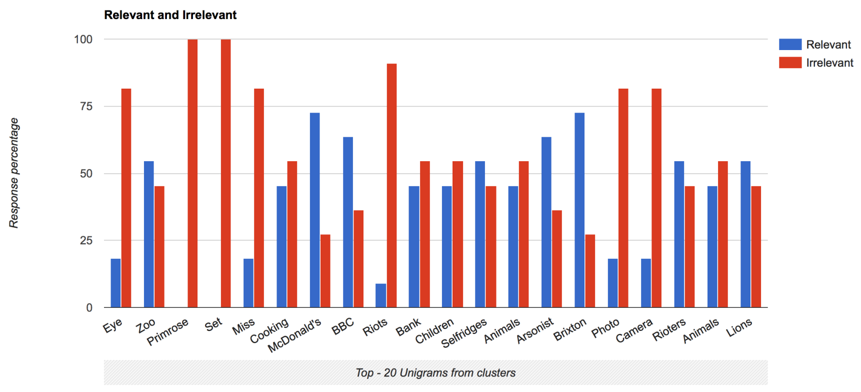


Figure 10.19: Comparative study of relevance of Individual Words from the Cluster

In this section, we have evaluated different phases of our method and the final output by manual metric evaluation and the performance is analysed. Apart from manual metric method, the selection of tweets as summary for the clusters has also been evaluated using intrinsic method by comparing the automatically selected tweets with a set of manually selected tweets. The coverage of the words in the automatically selected tweets with respect to the manually selected tweets is treated as the metric measure. In the next chapter, we would summarize our work and discuss the future aspects.

Chapter 11

Conclusion

11.1 Summary

It is a challenging job for the data journalists to analyse the nature of millions of tweets sent on a certain topic. However, a concise way of representing the huge number of tweets is to represent them in form of clusters. But, text clustering results into clusters which consists of group of words. Therefore, analysis of these clusters or semantic classification of the clusters is difficult without any prior knowledge about the subtopics present in the tweets.

In this thesis, we have developed and implemented a graph based approach which has successfully helped in effective semantic classification of the clusters. We started our study with a collection of tweets from 2011 London riots. We have implemented k-means++ clustering on this tweet corpus. Each of the clusters formed from this corpus is a collection of words. We developed and implemented a phrase mining technique to mine phrases from the words in the clusters. Subsequently, a graph based approach is implemented to find the longest phrase and the frequent phrases for each of the clusters. Summarization of tweets is generated in the next step, for each cluster to provide the data journalists with sufficient relevant information. Instead of dealing with millions of tweets, they can now count on the summary of tweets for the same information. Summary is available for each of the clusters, therefore, the data journalists can have a clear picture of the subtopics present in the tweets under the main topic. The frequent phrases formed from each cluster mapped onto the most similar DBpedia article(s) provides a semantic enrichment of the clusters.

Our overall contribution in this thesis can be summarized as:

- Mining of phrases from the clusters with the help of the original tweets.

  Phrases are mined by comparing the bigrams formed from original tweets with the bigrams formed from the clusters.

- Form a word graph $G = (V, E)$ from the mined phrases.

  If a word u appears before word v in any of the phrases then there is a directed edge from node u to node v. The value at the edges represent the frequency of the phrase uv in the tweets.

- Graph partitioning is done to have the most frequent phrases.

  Partitioning of the graph G(V,E) is done by deleting the set of edges having minimum weight, such that indegree of each vertex is 1. If a cycle exists in the graph after partition, the cycle is removed from the graph to generate a Directed Acyclic Graph.

- Longest Path Problem for Directed Acyclic Graph is solved to obtain the longest phrase and the set of frequent phrases for each cluster.

- Summarization of tweets is done with the help of the longest phrase.

  Five most relevant tweets for each of the clusters are extracted from the original corpus with the help of the longest phrase.

- Frequent phrases mined from the graph and the words from the cluster which could not form any phrase are mapped onto the most similar DBpedia article(s).

  Disambiguation of the phrases are done by exploiting Wikipedia articles and also the URL of the DBpedia articles. Thereafter, the abstract of the corresponding most similar DBpedia article(s) is extracted for each of these phrases and words from the cluster. Thus a semantic enrichment of the words and the phrases from the cluster is achieved.

Therefore, we can conclude that each cluster has a small collection of tweets associated with it which perfectly describes the content of the cluster. Also, each phrases formed from the cluster is associated with the abstract of the most similar DBpedia article which provides a vivid explanation of the phrases. Further, the rest of the words from the cluster which are not part of any phrases are also mapped onto the most similar DBpedia article as in for the phrases.

11.2 Outlook

The methods developed and implemented in this thesis can be easily extended for the tweets of different languages. NLTK comes with a corpora for many languages[1], hence stopwords can be easily identified. Also DBpedia is a multilingual knowledge base. Therefore, the phrases and words which are now mapped only to the corresponding abstracts of the most similar English DBpedia articles can also be easily mapped to the DBpedia articles of different languages.

All the methods can be applied to a corpus containing Facebook status updates, messages or any other social media updates or messages which is in form of text. Moreover, the steps followed after clustering of the tweets can be applied to any the text clusters if one wishes to have a summary for each of the clusters as well as some semantic enrichment of the content in the clusters.

It can also be a very useful method to analyse the clusters formed from some historical data corpus, where the enrichment of the clusters can give us a better insight of the data. Also a short summary for each of the clusters will give a clear picture of the information conveyed by the cluster.

[1] http://www.nltk.org/book/ch02.html

Bibliography

[1] Letierce, Julie and Passant, Alexandre and Breslin, John and Decker, Stefan (2010) Understanding how Twitter is used to spread scientific messages. *In: Proceedings of the WebSci10: Extending the Frontiers of Society On-Line, April 26-27th, 2010, Raleigh, NC: US.*

[2] Zhao, D. and Rosson, M.B., 2009, May. How and why people Twitter: the role that micro-blogging plays in informal communication at work. *In Proceedings of the ACM 2009 international conference on Supporting group work (pp. 243-252). ACM.*

[3] Kwak, H., Lee, C., Park, H. and Moon, S., 2010, April. What is Twitter, a social network or a news media?. *In Proceedings of the 19th international conference on World wide web (pp. 591-600). ACM.*

[4] Twitter and the News: How people use the social network to learn about the world. *Survey report published by American Press Institute 2015* Accessed on July, 2016

[5] Sankaranarayanan, J., Samet, H., Teitler, B.E., Lieberman, M.D. and Sperling, J., 2009, November. Twitterstand: news in tweets. *In Proceedings of the 17th ACM SIGSPATIAL international conference on advances in geographic information systems (pp. 42-51). ACM.*

[6] O'Connor, B., Krieger, M. and Ahn, D., 2010, May. TweetMotif: Exploratory Search and Topic Summarization for Twitter. *In ICWSM.*

[7] Eisenstein, J., O'Connor, B., Smith, N.A., and Xing, E.P. 2010. A Latent Variable Model for Geographic Lexical Variation.*In Proceedings of Conference on Empirical Methods in Natural Language Processing*

[8] Lee, K., Palsetia, D., Narayanan, R., Patwary, M.M.A., Agrawal, A. and Choudhary, A., 2011, December. Twitter trending topic classification. *In 2011 IEEE 11th International Conference on Data Mining Workshops (pp. 251-258). IEEE.*

[9] Osborne, M., Petrovic, S., McCreadie, R., Macdonald, C. and Ounis, I., 2012, August. Bieber no more: First story detection using Twitter and Wikipedia. *In SIGIR 2012 Workshop on Time-aware Information Access.*

[10] Kapanipathi, P., Jain, P., Venkataramani, C. and Sheth, A., 2014, May. User interests identification on twitter using a hierarchical knowledge base. *In European Semantic Web Conference (pp. 99-113). Springer International Publishing.*

[11] Gabrilovich, E. and Markovitch, S., 2009. Wikipedia-based semantic interpretation for natural language processing. *Journal of Artificial Intelligence Research, 34, pp.443-498.*

[12] Rosa, K.D., Shah, R., Lin, B., Gershman, A. and Frederking, R., 2011. Topical clustering of tweets. *Proceedings of the ACM SIGIR: SWSM.*

[13] Genc, Y., Sakamoto, Y. and Nickerson, J.V., 2011, July. Discovering context: classifying tweets through a semantic transform based on wikipedia. *In International Conference on Foundations of Augmented Cognition (pp. 484-492). Springer Berlin Heidelberg.*

[14] Lehmann, J., Isele, R., Jakob, M., Jentzsch, A., Kontokostas, D., Mendes, P.N., Hellmann, S., Morsey, M., van Kleef, P., Auer, S. and Bizer, C., 2015. DBpedia–a large-scale, multilingual knowledge base extracted from Wikipedia. *Semantic Web, 6(2), pp.167-195*

[15] Auer, S., Bizer, C., Kobilarov, G., Lehmann, J., Cyganiak, R. and Ives, Z., 2007. Dbpedia: A nucleus for a web of open data. *In The semantic web (pp. 722-735). Springer Berlin Heidelberg.*

[16] Resource Description Framework (RDF): Concepts and Abstract Syntax W3C Recommendation 10 February 2004 http://www.w3.org/TR/PR-rdf-syntax, Accessed on March,2016

[17] Resource Description Framework (RDF): Concepts and Abstract Syntax W3C Recommendation 10 February 2004 http://www.w3.org/TR/2004/REC-rdf-concepts-20040210, Accessed on March,2016

[18] Exner, P. and Nugues, P., 2012. Entity extraction: From unstructured text to DBpedia RDF triples. *In The Web of Linked Entities Workshop (WoLE 2012) (pp. 58-69). CEUR.*

[19] DBpedia version 2015-10 http://wiki.dbpedia.org/dbpedia-dataset-version-2015-10 Accessed on July, 2016

[20] DBpedia 2015-10 Dataset Statistics http://wiki.dbpedia.org/services-resources/datasets/dataset-2015-10/dataset-2015-10-statistics Accessed on July, 2016

[21] Chvatal, Václáv, and David Sankoff. "Longest common subsequences of two random sequences." *Journal of Applied Probability (1975): 306-315.*

[22] Bizer, C., Lehmann, J., Kobilarov, G., Auer, S., Becker, C., Cyganiak, R. and Hellmann, S., 2009. DBpedia-A crystallization point for the Web of Data. *Web Semantics: science, services and agents on the world wide web, 7(3), pp.154-165.*

[23] Levenshtein, V.I., 1966, February. Binary codes capable of correcting deletions, insertions and reversals. *In Soviet physics doklady (Vol. 10, p. 707)*

[24] Fung, G., 2001.A comprehensive overview of basic clustering algorithms.

[25] Graepel. T., Statistical physics of clustering algortihms.,1998 *Technical Re- port 171822, FB Physik, Institut fur Theoretische Physic*

[26] I. Witten and E. Frank. Data Mining

[27] Kaufman L., and Rousseeuw P. J., Finding Groups in Data : an Introduction to Cluster Analysis. *John Wiley & Sons, 1990.*

[28] R.C.Dubes and A.K.Jain. Algorithms for Clustering Data. Prentice Hall, 1988.

[29] Vector Space Model from Slideshare
http://www.slideshare.net/dalal404/document-similarity-with-vector-space-model

[30] Christopher D. Manning, Hinrich Schütze, and Prabhakar Raghavan, (2008) Introduction to Information Retrieval

[31] Salton, G., Wong, A. and Yang, C.S., 1975. A vector space model for automatic indexing. *Communications of the ACM, 18(11),* pp.613-620.

[32] Kanungo, T., Mount, D.M., Netanyahu, N.S., Piatko, C.D., Silverman, R. and Wu, A.Y., 2002. An efficient k-means clustering algorithm: Analysis and implementation. *IEEE transactions on pattern analysis and machine intelligence, 24(7),* pp.881-892.

[33] Lee, D., Baek, S. and Sung, K., 1997. Modified k-means algorithm for vector quantizer design. *IEEE Signal Processing Letters, 4(1),* pp.2-4.

[34] Hartigan, J.A. and Wong, M.A., 1979. Algorithm AS 136: A k-means clustering algorithm. *Journal of the Royal Statistical Society. Series C (Applied Statistics), 28(1),*pp.100-108.

[35] Rajaraman, A., Ullman, J. D. (2011). "Data Mining". Mining of Massive Datasets (PDF). pp. 1–17

[36] Luhn, H.P., 1957. A statistical approach to mechanized encoding and searching of literary information. *IBM Journal of research and development, 1(4), pp.309-317.*

[37] G. B. Arfken , H. J. Weber (2000). Mathematical Methods for Physicists (5th ed.)

[38] Arthur, D. and Vassilvitskii, S., 2007, January. k-means++: The advantages of careful seeding.*In Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms (pp. 1027-1035). Society for Industrial and Applied Mathematics.*

[39] Bontampi, G. and Taieb, S.B. (2013) Statistical foundations of machine learning.Section 13.1.2 OTexts: Melbourne, Australia. http://otexts.org/fpp/. Accessed on November, 2015.

[40] Idé, T., 2006, September. Why does subsequence time-series clustering produce sine waves?. *In European Conference on Principles of Data Mining and Knowledge Discovery (pp. 211-222). Springer Berlin Heidelberg.*

[41] Collins, M.J., 1996, June. A new statistical parser based on bigram lexical dependencies. *In Proceedings of the 34th annual meeting on Association for Computational Linguistics (pp. 184-191).* Association for Computational Linguistics.

[42] Tamilselvi, R., Sivasakthi, B. and Kavitha, R., 2015. AN EFFICIENT PREPROCESSING AND POSTPROCESSING TECHNIQUES IN DATA MINING.

[43] S. Kitaev and S. Seif: Combinatorics for asymptotic bounds on the free spectrum of Perkins semigroup, in preparation.

[44] Graphs and Words by Sergey Kitaev and Vadim Lozin,*Springer*

[45] Daciuk, J., Mihov, S., Watson, B.W. and Watson, R.E., 2000. Incremental construction of minimal acyclic finite-state automata. *Computational linguistics, 26(1), pp.3-16.*

[46] Leskovec, J., Backstrom, L. and Kleinberg, J., 2009, June. Meme-tracking and the dynamics of the news cycle. *In Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining (pp. 497-506). ACM.*

[47] Nichols, J., Mahmud, J. and Drews, C., 2012, February. Summarizing sporting events using twitter. *In Proceedings of the 2012 ACM international conference on Intelligent User Interfaces (pp. 189-198). ACM.*

[48] Sharifi, B., Hutton, M.A. and Kalita, J.K., 2010, August. Experiments in microblog summarization. *In Social Computing (SocialCom), 2010 IEEE Second International Conference on (pp. 49-56). IEEE.*

[49] Calinescu, G., Karloff, H. and Rabani, Y., 1998, May. An improved approximation algorithm for multiway cut. *In Proceedings of the thirtieth annual ACM symposium on Theory of computing (pp. 48-52). ACM.*

[50] Sedgewick, Robert; Wayne, Kevin Daniel (2011), Algorithms (4th ed.), Addison-Wesley Professional, pp. 661–666

[51] Procter, R., Vis, F. and Voss, A., 2013. Reading the riots on Twitter: methodological innovation for the analysis of big data. *International journal of social research methodology, 16(3),* pp.197-214.

[52] Twitter usage/Company Facts Numbers approximate as of June 30, 2016 `https://about.twitter.com/company` Accessed on July, 2016

[53] Dorsey, Jack (March 21, 2006). "just setting up my twttr". Twitter. Retrieved February 4, 2011.

[54] Abel, F., Gao, Q., Houben, G.J. and Tao, K., 2011, May. Semantic enrichment of twitter posts for user profile construction on the social web. *In Extended Semantic Web Conference (pp. 375-389). Springer Berlin Heidelberg.*

[55] Hecht, B., Hong, L., Suh, B. and Chi, E.H., 2011, May. Tweets from Justin Bieber's heart: the dynamics of the location field in user profiles. *In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (pp. 237-246). ACM.*

[56] Agarwal, A., Xie, B., Vovsha, I., Rambow, O. and Passonneau, R., 2011, June. Sentiment analysis of twitter data. *In Proceedings of the workshop on languages in social media (pp. 30-38). Association for Computational Linguistics.*

[57] Jansen, B.J., Zhang, M., Sobel, K. and Chowdury, A., 2009. Twitter power: Tweets as electronic word of mouth. Journal of the American society for information science and technology, 60(11), pp.2169-2188.

[58] Sakaki, T., Okazaki, M. and Matsuo, Y., 2010, April. Earthquake shakes Twitter users: real-time event detection by social sensors. *In Proceedings of the 19th international conference on World wide web (pp. 851-860). ACM.*

[59] Denef, S., Bayerl, P.S. and Kaptein, N.A., 2013, April. Social media and the police: tweeting practices of british police forces during the August 2011 riots. *In proceedings of the SIGCHI conference on human factors in computing systems (pp. 3471-3480). ACM.*

[60] Lee, R. and Sumiya, K., 2010, November. Measuring geographical regularities of crowd behaviors for Twitter-based geo-social event detection. *In Proceedings of the 2nd ACM SIGSPATIAL international workshop on location based social networks (pp. 1-10). ACM.*

[61] Hu, M., Liu, S., Wei, F., Wu, Y., Stasko, J. and Ma, K.L., 2012, May. Breaking news on twitter.*In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (pp. 2751-2754). ACM.*

[62] Phuvipadawat, S. and Murata, T., 2010, August. Breaking news detection and tracking in Twitter. In Web Intelligence and Intelligent Agent Technology (WI-IAT), 2010 IEEE/WIC/ACM International Conference on (Vol. 3, pp. 120-123). IEEE.

[63] Reading the Riots, an interactive page by the newspaper The Guardian, Accessed on June, 2016

[64] Weng, J. and Lee, B.S., 2011. Event Detection in Twitter. *ICWSM, 11, pp.401-408.*

[65] Michelson, M. and Macskassy, S.A., 2010, October. Discovering users' topics of interest on twitter: a first look. *In Proceedings of the fourth workshop on Analytics for noisy unstructured text data (pp. 73-80). ACM.*

[66] Xu, Z., Ru, L., Xiang, L. and Yang, Q., 2011, August. Discovering user interest on twitter with a modified author-topic model. *In Web Intelligence and Intelligent Agent Technology (WI-IAT), 2011 IEEE/WIC/ACM International Conference on (Vol. 1, pp. 422-429). IEEE.*

[67] Hong, L., Ahmed, A., Gurumurthy, S., Smola, A.J. and Tsioutsiouliklis, K., 2012, April. Discovering geographical topics in the twitter stream. *In Proceedings of the 21st international conference on World Wide Web (pp. 769-778). ACM.*

[68] Bedathur, S., Berberich, K., Dittrich, J., Mamoulis, N. and Weikum, G., 2010. Interesting-phrase mining for ad-hoc text analytics. *Proceedings of the VLDB Endowment, 3(1-2), pp.1348-1357.*

[69] Richman, A.E. and Schone, P., 2008, June. Mining Wiki Resources for Multilingual Named Entity Recognition.*In ACL* (pp. 1-9).

[70] Salvador García, Julián Luengo, Francisco Herrera Data Preprocessing in Data Mining pp. 1-17 *ISBN: 978-3-319-10246-7 (Print) 978-3-319-10247-4 (Online)*

[71] Sembiring, R.W., Zain, J.M. and Embong, A., 2011. A comparative agglomerative hierarchical clustering method to cluster implemented course. *arXiv preprint arXiv:1101.4270.*

[72] Jeffrey Strickland. Predictive Analytics using R *Lulu.com (January 16, 2015)* pp.147

[73] Lloyd, S., 1982. Least squares quantization in PCM. *IEEE transactions on information theory, 28(2),* pp.129-137.

[74] Mahajan, M., Nimbhorkar, P. and Varadarajan, K., 2009, February. The planar k-means problem is NP-hard. *In International Workshop on Algorithms and Computation (pp. 274-285). Springer Berlin Heidelberg.*

[75] Deza, M.M. and Deza, E., 2009. Encyclopedia of distances. In Encyclopedia of Distances (pp. 1-583). *Springer Berlin Heidelberg.*

[76] Thulasiraman, K.; Swamy, M. N. S. (1992), ”5.7 Acyclic Directed Graphs”, Graphs: Theory and Algorithms, *John Wiley and Son, p. 118, ISBN 978-0-471-51356-8.*

[77] Directed Acyclic Word Graphs in `http://www.wutka.com/dawg.html` Accessed on May, 2016

[78] Fiduccia, C.M. and Mattheyses, R.M., 1982, June. A linear-time heuristic for improving network partitions. *In Design Automation, 1982. 19th Conference on (pp. 175-181). IEEE*

[79] Floyd, R.W., 1962. Algorithm 97: shortest path. *Communications of the ACM, 5(6), p.345.*

[80] Cormen, T.H., 2009. Introduction to algorithms. *MIT press.*

[81] Skiena, S. S. The Algorithm Design Manual (Springer-Verlag, 1998).

[82] SPARQL by W3C `http://www.w3.org/2009/Talks/0615-qbe/` Accessed on May, 2016

[83] Liu, L. and Özsu, M.T., 2009. Encyclopedia of database systems (Vol. 6). Berlin, Heidelberg, Germany: Springer.

[84] Digital, Social & Mobile in 2015 Published January, 2016 by We are social

[85] Ketchen, D.J. and Shook, C.L., 1996. The application of cluster analysis in strategic management research: an analysis and critique. *Strategic management journal, 17(6),* pp.441-458.