Bursty Event Discovery from Online News Outlets

by

SEYED POORIA MADANI KOCHAK

Bachelor of Computer Science with Honours, University of Prince Edward Island, 2012

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

Master of Computer Science

In the Graduate Academic Unit of Faculty of Computer Science

This thesis is accepted by the

Dean of Graduate Studies

THE UNIVERSITY OF NEW BRUNSWICK

April, 2015

Abstract

On this thesis, we have developed a set of methods along with a framework for discovery of bursty events and their relationship from streams of online news articles. Bursty event discovery can be done using the discovered bursty terms which are significantly smaller in size compared to the original feature-set. Moreover, the discovered bursty events are compared in order to discover any potential relational link between any of two. It is the assumption of this work that bursty events and their relationship in time can provide useful information to firms and individuals who their decision making process is significantly affected by news events. The system performed at 64% level of accuracy on a real world dataset. The results show a great promise as do the implicit measures that our proposed framework and methods can be utilized towards real world applications.

Dedication

I dedicate my dissertation work to my amazing wife, Shabnam Mousavi, whose words of encouragement and push for tenacity ring in my ears.

Acknowledgments

I wish to express my sincere thanks to my supervisor, Dr. Ali A. Ghorbani, for his guidance and support thought the course of this thesis. I take this opportunity to express gratitude to all of the faculty members for their help and support. I also thank my parents for the unceasing encouragement, support and attention. I am also grateful to my partner who supported me through this venture. I also place on record, my sense of gratitude to one and all, who directly or indirectly, have let their hand in this venture.

Table of Contents

List of Tables

List of Figures

List of Abbreviations

NED	New Event Discovery
RED	Retro Event Detection
TTM	Temporal Text Mining
POS	Part of Speech
NLP	Natural Language Processing
PCA	Principle Component Analysis
LSA	Latent Semantic Analysis
TDT	Topic Detection and Tracking
FSD	First Story Detection
TIE	Temporal Indexing Engine
DFT	Discrete Fourier transformation

Nomenclature

Symbol	Meaning	Section
D	Denotes document collection	2.1
T	Denotes global dictionary for terms	2.1
tf	Term frequency count	2.1
idf	Inverse document frequency	2.1
df	Document frequency count	2.5
$sim_t^c(d,q)$	Cosine similarity between document D and Q	2.5
mintiest	Minimum points per cluster	2.10
$D(\theta_1\|\|\theta_2)$	Computing similarity between two mixture model	2.11
f_i	i-th feature in the feature-set	3.3
w_j	j-th window	3.3
$n_{i,j}$	Number of occurrence of feature i in window j	3.3
P_g	Binomial probability of occurrence of feature i in window j	3.3
$s(i)$	Silhouettes Score	4.4

Chapter 1

Introduction

1.1 Background and Motivation

With advances in technology and the world wide web, access to vital information became easier and faster. As a result, analyzing such information and acting on is more important than any time in the history of civilizations. Today, a typical investor has to read a large volume of news articles and base his decision on the new articles by connecting the dots and predicting future of the stock market. President of a big food corporation has to be aware of the ongoing news in the world as different events can affect the supply chain of the company and result in a saviour market lost. Consequently, with today's availability of information, only those who analyze these information quicker are able to see the big picture and succeed in the competitions. Event extraction and tracking, a subfield of Topic Modelling and Topic Tracking (TDT) in temporal text mining, studies models of event detection and extraction from a stream of unstructured text data. Events play an important role in our daily life and every organization utilizes news events differently. For example, scientist who wants to study all the earthquakes that occurred in the United States since 1980 can use an event detection system towards finding such events and their related information from historical news articles (a.k.a. retrospective event extraction). Or it may be important for hedge funds in Wall Street to detect events as they occur that may affect their investments.

Event detection and extraction can be divided into two subfields known as *Retrospective Event Detection (RED)* and New (novel) Event Detection (NED). In NED, the nature of events can be unknown and it is important to detect events as they raise in the stream. In contrast, in RED, the nature of events that one is trying to extract can be known in advance and such knowledge can be incorporated in the extraction process.

Bursty events are special type of events that their sudden occurrence may have strong consequences on many daily tasks, therefore it is important for different entities (i.e. firms) to be aware as they take place. Given the nature of bursty events, the whole stream of news articles needs to be monitored and studied in order to discover such events, which itself is proven to be a challenging task.

In this thesis we are proposing a framework for detecting and extracting bursty events by utilizing time in our calculations and finding *relationship* between the discovered bursty events in a course of time. Due to the fact that we are considering the role of time in text documents, our research is considered as *Temporal Text Mining* (TTM). By considering time windows in our framework, we expect to be able to discover *causal* relationship between the events and enable the users to draw a big picture of the ongoing issues on news media and be able to analyze the news articles more efficiently.

1.2 Thesis Objectives

The objective of this thesis work is to create a framework for discovering bursty events and their relationships from a stream of news article. Due to the fact that events take place in course of time, this thesis research will take into account the temporal nature of events and provide a set of methods in realm of temporal text mining. Thus, temporal document indexing architecture has been studied extensively in this research work.

Moreover, after discovering bursty events using our discussed novel approach, this research work will use the graph theory for constructing relation graph of related events. By representing events as nodes and their relations as edges, we argue that human agents can see the bigger picture of the discovered issues and can draw their conclusion and analysis more informed.

In summary, our contributions are providing a general framework for temporal document indexing, developing a novel procedure in discovering bursty events, discovering link between the bursty events in course of time, and visualizing them as a graph.

1.3 Thesis Organization

Chapter 2 discusses the essential background techniques and methods that are used in the field of text mining and can help with the understanding of this research work. Moreover, Chapter 2 covers the related works that are previously done on extracting bursty events.

Chapter 3 presents the overall architecture of the framework and our proposed solutions. It discusses the challenges for utilizing the framework on the real world data and proposes models to over come such difficulties. Moreover, in a great length, it explains the algorithms that are used for processing, indexing, extracting, and linking bursty events.

Chapter 4 goes over the implementation of the proposed framework. It discusses the nature the test-dataset and the results that are obtained. It demonstrates the performance of the burst detection algorithm and provides some examples of the discovered bursty events.

Chapter 5 provides an overview of possible future works that can be done in order to expand or improve our proposed methods. It discusses the challenges that we faced with during the tests and implementation of our work and makes comments which can enhance the performance of the proposed framework.

Chapter 2

Literature Review

2.1 Text Encoding

For mining and extracting useful information from a large collection of text documents, it is necessary to pre-process and store the documents in a suitable data structure which can be used by the data mining and machine learning algorithms. Even though natural language processing techniques can reveal more about the words and their syntactical meaning in a text document, most text mining approaches assume that text documents are simply a set of words (bag of words representation).

2.1.1 Preprocessing

The goal of preprocessing is to build a *global dictionary* for a document collection in order to transform documents into the *bag of words representation* form. Dictionary of a document collection is the list of all the words have been used in the collection of documents. As a formal description, let $D = \{d_1, d_2, \ldots, d_n\}$ denote a text document collection of size n and $T = \{t_1, t_2, \ldots, t_m\}$ denote collection of terms that are used in D, then T is called the *global dictionary.* In order to be able to represent each document in the bag of words representation, it is important to have the frequency of each term $t \in T$ in document $d \in D$. As a result, we define the notion of *term-frequency* for a term t in a document d as $tf(d,t)$. One can define a term-vector for each $t \in T$ and for each $d \in D$ as $\vec{t_d} = (tf(d,t_1), \ldots, tf(d,t_m))$.

Another goal in the pre-processing phase is to remove the words that are not important for the purpose of analysis in order to reduce the size of the *global dictionary.* The *global dictionary* holds the word-terms which are used in the most of the machine learning and data mining algorithms, and since the size of the feature space is very critical to most of the machine learning algorithms, the reduction of size can be very beneficial. This problem is known as curse dimensionality. Moreover, words that bear little or no content information, like articles and conjunctions are considered to be noise and are removed.

Stop word filtering is the process of removing words that have no contextual information. Stop word filtration not only removes words with less contextual information, it also removes words that are most frequent. Words that are most frequent can be regarded as no particular statistical relevance and can be removed from the global dictionary.

Lemmatization methods are another approach for reducing feature space. These methods use known word form and map verbs to their infinite tense and nouns to their singular form. Knowing the correct word form demands *part of speech* *taging* (POS) of each word in a text document, thus creating POS can be very time consuming and error prone. Consequently, this method is not widely used in the field of text mining.

Stemming methods accomplish the same goal as Lemmatization. However, stemming does not use NLP techniques for reducing words to their basic form, rather uses a rule based approach. Authors define rules to reduce words to their basic forms; for example, one can define a rule that "-ing" should gets dropped from end of each word.

All these three pre-processing steps helps with building a global dictionary free of noise and unnecessary words in order to get accurate and reliable results from text mining algorithms. However, after these three steps it is very common that the size of the global dictionary remain large and cause the curse of dimensionality problem for text mining algorithms . Therefore, other algorithms are used to select terms from the global dictionary that are more appropriate for text mining purposes and can represent most of the documents in the document collection. Such algorithms are known to be feature selection algorithms and bellow is the list of techniques that can be used to extract important features from dictionary in order to reduce the size of feature space:

- Principle Component Analysis (PCA)
- Multifactor dimensionality reduction
- Kernel PCA
- Latent semantic analysis (LSA)

- Independent component analysis
- Entropy-based selection.

Most of the named items such as PCA and LSA are computationally expensive to perform and are not feasible for document collections that are updated routinely. However, entropy-based selection is a straight forward and a fast method that can handle routinely updated document collections without substantial performance loss.

$$W(t) = 1 + \frac{1}{log_2|D|} \sum_{d \in D} P(d,t) log_2 P(d,t) \tag{2.1}$$

$$P(d,t) = \frac{tf(d,t)}{\sum tf(d_i,t)} \tag{2.2}$$

This method assumes that the words which have occurred less in a document can separate and describe the documents better. Therefore, less frequent word have high entropy. This is a greedy method that select only subset of the global dictionary given the configured threshold. Equation 1 shows the formula for calculating entropy for each term.

2.1.2 Linguistic Preprocessing

Even though linguistic preprocessing is not a necessary step in text mining, in some cases it can enhance the available information about terms in the global dictionary. Bellow is the brief list of techniques that gets used:

Part of speech tagger(POS-tagger)

Finds out role(noun, verb, etc.) of each term in a sentence and tag the term accordingly. It is worth to mention that machine learning algorithms can be used to learn the tagging process if a training data set is available.

Text chunking

Group the adjacent words together in order to form a new term. For example, "United States of America" should be considered as a unique term instead of set of delimitated terms.

Parsing

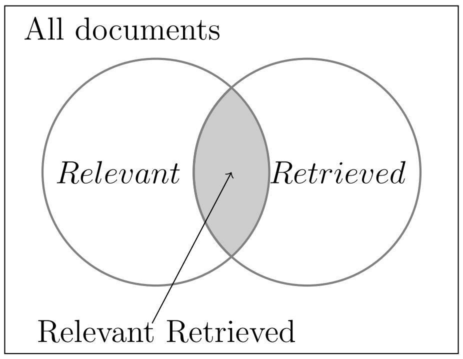
Build a parse tree of sentences in a text document. Using the constructed parse tree it is possible to discover relationship between terms and solve ambiguities that may occur during the processing.

2.2 Precision and Recall

The main goal of an information retrieval system is to find and retrieve *related documents* which is more than a simple *pattern matching* [12] . In order to evaluate the performance of an information retrieval system, one must pay close attention to the values returned by the two measures known as *Precision and Recall.* As shown in Figure 2.1, from a set of available documents in an information retrieval systems only a subset of the documents are considered relevant to a particular issued query and the goal is to return all the relevant documents. However, in most cases the retrieved documents are not all relevant which can be considered the deficiency of the retrieval system.

As given by Formula 2.3, *Precision* measures the ratio of the relevant documents

Figure 2.1: Result Set: Relevant Retrieved, Relevant, and Retrieved



that are retrieved to the total number of the document retrieved. However, having a high *Precision* does not always mean that the system is performing at its best. For example, a system that returns ten documents where nine of which are relevant has $Precision = 0.9$ but there might be hundred relevant documents that were available to the system and the system failed to capture and return all of them to the user. Consequently, another measure, as given by Formula 2.4, demonstrates how well the system is doing in terms of the ratio of the *relevant retrieved* documents to the total number of the documents available in the system that are believed to be relevant.

$$Precision = \frac{\text{Relevant Retrieved}}{\text{Retrieved}} \tag{2.3}$$

$$Recall = \frac{\text{Relevant Retrieved}}{\text{Relevant}} \tag{2.4}$$

Optimal case is when Precision is equal to Recall which almost never happens because the actual number of relevant documents are not always known and many estimation methods are being used to calculate this number. Therefore, to obtain a desired level of Recall more documents need to be retrieved which results in huge drop in the precision. Thus, it is important to come up with an acceptable value for both Precision and Recall. Normally modern standard document retrieval systems have average precision of between 0.2 and 0.3 [12].

2.3 Topic Detection and Tracking

The objective of Topic Detection and Tracking(TDT) is to develop an evaluation paradigm that addresses event organization of broadcast news. In other words, TDT research tries to address challenges in extracting events from news articles and grouping the related events together. As discussed by James Allan [2], TDT research initially sponsored by Defence Advanced Research Projects Agency (DARPA) of the United States in 1990s and was meant to perform TDT not only online news articles but also on selected television channels and radio channels as well. TDT defines topics to be a set of strongly related news articles and as a result there are two categories of topics that can be defined, namely *event-based topics* and *subject-based topics.* In subject-based topics, focus is on what topics or ongoing issues is all about rather than what has triggered the event. Our work fall in *subject-based* topic modelling, however by creating links between events we are expecting to eventually find casual relationship between different events and come up with explanation what has triggered sequence of events in a chain(event-based topics). Due to the temporal nature of topics in TDT systems, most of the discovered events(a.k.a topics) are temporally anchored and further, the events evolve over time to include clearly related events that may not be grouped to representing same subject. This research work, by considering the temporal nature of the news events, tries to discover links over time between discovered events in order to reveal the flow and evolvement of events through course of time.

As defined in the literature [2], Topic Detection and Topic Modelling focuses on five sub-tasks:

- **Story Segmentation:** segmenting broadcasted news show into different news stories.
- **First Story Detection:** detecting novel event/news from news streams.
- **Cluster Detection:** grouping stories as they arrive and can be considered the main part of event/topic detection.
- **Tracking:** sequence of actions for monitoring the stream of news stories to find more related stories on a topic that was identified.
- **Story Link Detection:** deciding whether two randomly selected stories are related.

Story segmentation is mostly used in video or audio broadcasts as news anchor present many news events at once. Therefore, it is important for TDT system to be able to breakdown the whole news program's script into different news stories before feeding to the TDT system. One of the methods that are broadly used in *Story Segmentation* is detecting change in use of vocabularies in order to conclude that a different news story has been started [23]. However, since the main focus of this research work is on written news articles from online news broadcasting websites, the problem of story segmentation is out of scope of this work and we are not covering any details in this ground.

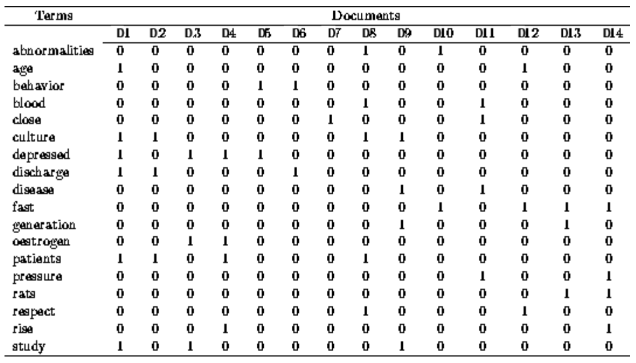
2.4 First Story Detection

First Story Detection (FSD), also known as New Event Detection (NED) in the literature [32], is the process of recognize new topics as they appear and had not been discussed earlier than some period (Figure 2.3). Technology of this type is of particular interest to information, security, or stock analysts whose job is to look for new events that are of significance in their area. Typically FSD is done by reducing stories to a set of features that can be substantially smaller either as vector [1] or a probability distribution [15].

The main difference between FSD and Cluster Detection is that FSD algorithms do not rely on huge a number of instances of a new story in order to detect its novelty. Their performance is at their best when they are able to detect a new story by only seeing a single instance of it. Normally a model of previous stories are built and a new incoming story will be tested against the existing model in order to decide that if the incoming story instance can be classified as new. As Allan et al. [2] have discussed, simple feature representations are not very effective at detecting new stories from the stream and more sophisticated approaches are needed.

What has been most popular in the literature [10, 37, 39] is representing documents

Figure 2.2: Example of Term Document Matrix of representing documents as a bag of words.



Terms	Documents													
	D1	D2	D3	D4	D5	D6	D7	D8	D9	D10	D11	D12	D13	D14
abnormalities	0	0	0	0	0	0	0	1	0	1	0	0	0	0
age	1	0	0	0	0	0	0	0	0	0	0	1	0	0
behavior	0	0	0	0	1	1	0	0	0	0	0	0	0	0
blood	0	0	0	0	0	0	0	1	0	0	1	0	0	0
close	0	0	0	0	0	0	1	0	0	0	1	0	0	0
culture	1	1	0	0	0	0	0	1	1	0	0	0	0	0
depressed	1	0	1	1	1	0	0	0	0	0	0	0	0	0
discharge	1	1	0	0	0	1	0	0	0	0	0	0	0	0
disease	0	0	0	0	0	0	0	0	1	0	1	0	0	0
fast	0	0	0	0	0	0	0	0	0	1	0	1	1	1
generation	0	0	0	0	0	0	0	0	1	0	0	0	1	0
oestrogen	0	0	1	1	0	0	0	0	0	0	0	0	0	0
patients	1	1	0	1	0	0	0	1	0	0	0	0	0	0
pressure	0	0	0	0	0	0	0	0	0	0	1	0	0	1
rats	0	0	0	0	0	0	0	0	0	0	0	0	1	1
respect	0	0	0	0	0	0	0	1	0	0	0	1	0	0
rise	0	0	0	1	0	0	0	0	0	0	0	0	0	1
study	1	0	1	0	0	0	0	0	1	0	0	0	0	0

using bag-of-words (BOW) using vector space model as shown in Figure 2.2 and calculate similarities between corresponding vectors using different similarity measures such as cosine-similarity. For novelty detection, the representing BOW's vector of an incoming new article gets tested against other news articles in the system for a given window, and decision will be made if there are any existing news article that can be associated with the new incoming one, or a true novel news story has been discovered.

$$weight_t(d,w) = \frac{1}{Z_t(d)} f(d,w) \cdot log \frac{N_t}{df_t(w)} \tag{2.5}$$

$$Z_t(d) = \sum_w f(d,w) \cdot log \frac{N_t}{df_t(w)} \tag{2.6}$$

$$sim_t^c(d,q) = \sum_w weight_t(d,w) \cdot weight_t(q,w) \tag{2.7}$$

In a seminal work from Palo Alto Research Centre [5], researchers developed a system that uses incremental TF-IDF model, generation of source-specific model, and term weighting as shown in Equation 2.5 for detecting NED system from news articles streams. They have argued that since each stream (news websites) have their own style of writing and their own way of anchoring the news, it is essential to train a temporal modal for each of the streams instead of one unified stream. Secondly, they compared each incoming document with all the documents in the system on a specific time window using Equation 2.7 and if the similarity value between the new document and existing document was below a minimum threshold then the incoming news article would receive label of being new-story. Their main contribution is the way of calculating the similarity between the documents: each document gets segmented into a number of overlapping parts that contains a few sentences. Each segment of the documents is compared against all the segments of the other document and the average similarity of the segments between the two news articles will be the degree of similarity between the two articles. In this work, authors have argued their feature weighting schema and segmentation-comparison have increased the accuracy of the system. However, it is important to note that their method is very expensive and can become quickly infeasible as soon as the volume of incoming data increases.

$$sim_t^c(d,q) = sim(d,q) \cdot (1 + 2^{-age/halftime}) \tag{2.8}$$

Moreover, their system has consideres *time-decay* in calculating their similarities. According to Equation 2.8 two documents are far apart in times are less similar than the ones are located in closer time windows. However, as discussed in their paper [5], considering such cost function can by itself be expensive and not very helpful.

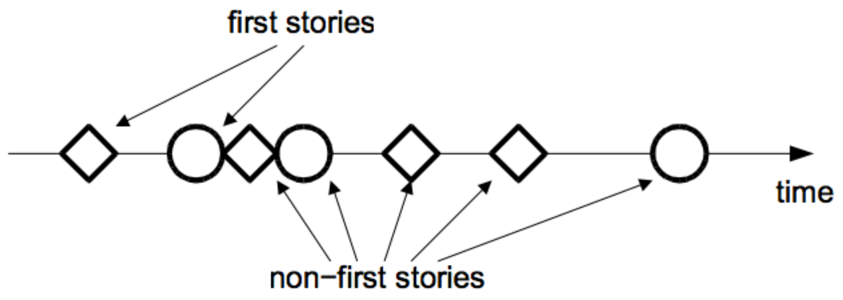


Figure 2.3: Typical New Event Discovery(NED) system [5]. Square and Circle are representing two different unique events that have accrued in the stream.

Novelty detection has been studied in other fields that have temporal sequences of data, such as monitoring sensors for detecting anomaly, or stock market prices to detect new behaviour regarding ongoing investments. As a result there are many signal processing and trend analysis techniques are used in this field that can be adapted for the proposed of TDT. Ma and Perkins [19] have developed an online algorithm for modelling temporal sequences using an online support vector regression algorithm for novelty detection.

2.5 Similarity Computation

Similarity measures can define the degree of similarity between two objects of interest in terms of some the defined characteristics of interest. For example, if documents are represented as a probability distribution over their terms, then the similarity of any two chosen text documents are dependent on the degree of similarity of the probability distributions. Therefore, feature representation (i.e. BOW) can dictate possibility of similarity functions to be used.

BOW representation is one of the most famous representation used in the field of text mining. Since documents are represented using a vector of their terms, cosine-similarity is one of the most used candidate for finding similarity.

$$similarity = \frac{\sum_{i=1}^{n} A_i \times B_i}{\sqrt{\sum_{i=1}^{n} (A_i)^2} \times \sqrt{\sum_{i=1}^{n} (B_i)^2}} \tag{2.9}$$

Cosine similarity, as defined in Equation 2.9, computes the degree of similarity of two vectors which ranges from 0 (no similarity) to 1 (being identical). It should be noted that as the vocabulary of documents grow, the terms used in their vector representations grow proportionally. As a result, the distance between vectors can grow *exponentially*, a phenomenon known as the curse of dimensionality. Two similar documents, due to the growing dimensions (i.e. number of terms in the global dictionary), can become dissimilar very quickly and result in loss of the overall accuracy of the system.

If feature is are represented using probability distributions, then *Hellinger distance* would be an ideal candidate to measure the closeness of the probability distributions.

$$H^2(P,Q) = \frac{1}{2} \int (\sqrt{\frac{\mathrm{d}P}{\mathrm{d}\lambda}} - \sqrt{\frac{\mathrm{d}P}{\mathrm{d}\lambda}})^2, \mathrm{d}\lambda \tag{2.10}$$

Formula 2.10 shows how Hellinger distance is calculated on two probability distributions of P and Q, and parameters in this equation are coming from *Random-Nikodym* derivatives of P and Q.

2.6 Curse of Dimensionality

Curse of dimensionality is the problem caused by the exponential increase in volume associated with adding extra dimensions to Euclidean space. As text documents represented using BOW vectors, each term can be considered as a distinguishable dimension. As the number of text documents grow so does the number of unique terms used in each of the individual text documents, consequently, the total dimension of representing documents in the space grow very rapidly. As dimension grows linearly the associated volume grows exponentially and as a result distance between documents (a.k.a. similarity) increases exponentially. In other words, documents that were very closer (in term of similarity distance) in lower dimensions are now far apart due to the added dimensions. Therefore, large dimensions not only increase computational cost, but also may distort the accuracy of computations.

Text documents, may use very different words based on the style of their writers and thus the number of unique terms (i.e. dimensions) increases very quickly. Therefore, it is advantageous to text mining applications to come up with features that are substantially lower in quantity and improve both the computation time and accuracy of the system.

2.6.0.1 Feature Selection

Feature selection approaches find a subset of features from the original set either using filtering or through search for increasing accuracy. For example, *information-gain* is considered as one of the most popular feature selection approach as only variables with high information-gain value are selected and ones bellow a certain threshold are discarded. One of the main contributions of this thesis work is utilizing bursty feature discovery for its feature selection task in order to reduce the number of features in the feature set before performing computation, for decreasing the computational cost and increasing accuracy.

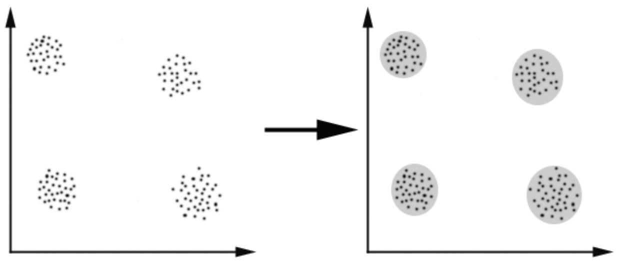
2.6.0.2 Feature Extraction

Feature extraction is the task of transforming data from high dimensional space into a space with fewer dimensions with minimum amount of loss. It is important to bear in mind that transformed dimensions are features that are constructed rather than plain selection (i.e. feature selection).
Principle Component Analysis (PCA), initially introduced by Pearson [25] and then independently by Hotelling [14], is a statistical method that uses orthogonal transformation to transform a set of possibly correlated features into a few linearly uncorrelated variables called *principle components*. It can be used for outlier detection in search of features that can be considered outliers.

2.7 Cluster Detection

Clustering or Cluster Analysis is the task of grouping objects in a way that objects belong to same group are more similar than objects in other groups(clusters) as shown in Figure 2.4. Document clustering in the field of text mining is the study of algorithms for grouping set of documents based on a defined similarity function. What is important to consider when study any particular clustering algorithm aside from its complexity, is the shape of the clusters that it can detect. Some clustering algorithms such as k-means [13] mostly detect clusters that have circular shape and while others like DBSCAN [36] can detect clusters with irregular shapes and sizes. Clustering algorithms can be further broken down to two categories of *dynamic clustering* and *static clustering* in terms of the number of the clusters they can detect [36]. In static clustering algorithms, number of clusters should be known in advance and provided to the clustering algorithm. This constraint can be very limiting in cases where number of clusters are not known ahead of time continuously are changing in temporal based applications. Dynamic clustering algorithms are capable of finding any number of clusters and do not require any prior knowledge about the number of clusters. DBSCAN is an example of such algorithms which has been explained in further details in Section 2.7.3.

Figure 2.4: An example of clustering analysis task.



The primary goal of cluster detection is to group stories on the same topic or event. The important difference between cluster detection and FSD is that cluster detection has to identify group of related topics/events while FSD has to detect the beginning

of an story. Furthermore, their evaluation is done differently, cluster detection will be punished once it misses a group of related topics while FSD will be heavily punished when it misses the beginning of a new emerging event [2].
The problem of TDT or in particular topic/event extraction can be formulated as unsupervised learning problem (clustering). It is important for a TDT system to be able to model events without any prior knowledge of the occurring events. There are two approaches discussed in the literature [6, 11, 26] about the clustering task: (1) clustering the features in the feature space or (2) clustering the documents. Since the nature of the features can dictate the availability of algorithms to be used for clustering, algorithms used in these two approach can be different but still some overlap may exist.

2.7.1 Document-Pivot Clustering

The main idea behind Document-Pivot clustering *event-detection* is clustering similar documents together and then select keywords from the discovered clusters that can describe a potential event. As stated in the study by Yang et al. [38], these techniques require many parameters that are very difficult be tuned. Moreover, some events can have very low number of supporting documents and as a result they get eliminated through this approach.
Several document-pivot clustering approaches have been tested in the text mining community for event-detection such as using Self-Organizing Neural Network [26], modelling clustering using probabilistic mixture models [21, 18], and other dynamic clusterings [6]. Due to the fundamental problem with the fine tuning parameters

in these techniques of extracting events, researchers have started to shift towards feature-pivot clustering techniques, that is described in the next section.

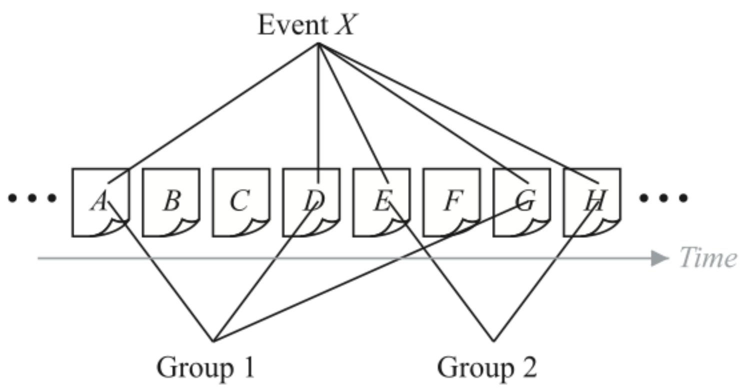


Figure 2.5: Each document can contribute to many bursty events [11].

2.7.2 Feature-Pivot Clustering

Feature-pivot clustering, as term suggests, is done by clustering the feature set instead of the document set. As a result, as shown in Figure 2.5, documents can belong to more than one cluster as its forming features are taking apart in different clusters. For example, a document that discusses government shutdown and its effect on stock markets can very well contribute to both events of *government shutdown*, and *collapse in the stock market*. Such documents help to create link a between different event clusters and help to summarize and explain some of these events. Therefore, it is reasonable to consider that some news articles may discuss more than one event and by clustering features instead of documents it is possible to discover such interrelated events. In this study, we adopt feature-pivot clustering as our main approach on event discovery and extraction, and we conduct a detail study to have a brief comparison between the two methods (document-pivot vs feature-pivot).

Fung et al. [11] developed a system in which frequency of terms were monitored for a period of time and bursty terms were selected as the main features. Then, the trend lines of selected features where clustered together using overlapping technique for time-series analysis in order to form events. Their work also discusses how time-series trend lines helps to detect the stop words that are specific to the corpus under the study.

Weng and Lee [35] developed an algorithm specialized on event detection from Twitter by performing signal processing on the temporal frequency of terms. Using different transformation, the resulting time-series show peaks and bursts a lot more clear and helps to store data using less information. Moreover, in their novel work, they have developed an quantification method for measuring *event significance*, which is calculating using cross correlation between the values of the signals forming the event.

2.7.3 DBSCAN

Density Based Spatial Clustering of Applications with Noise (DBSCAN) is a clustering algorithm proposed by Ester et al. [8]. This is a density-based clustering algorithm since it estimates the number of clusters initially from the estimated density distribution of corresponding nodes.

Two parameters: ϵ(eps) and minimum number of points(minPts) are required to identify and form a dense region. It randomly start with a unvisited point as the

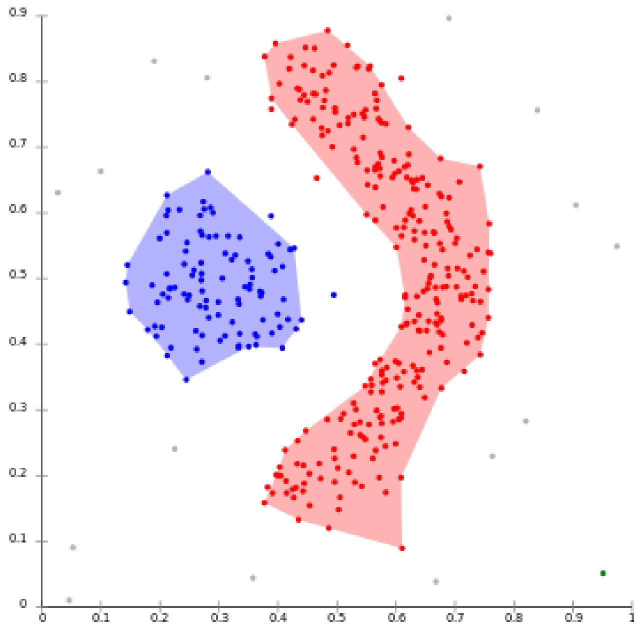


Figure 2.6: Example of shapes that DBSCAN can cluster.

starting position, then all the point's in ϵ-neighbourhood are retrieved. If the retrieved points are bigger that minPts, a new cluster will be formed, otherwise the point is labeled as noise. If a point is determined to be part of a cluster then all of its unvisited ϵ-neighbourhood points are retrieved and the same steps will be repeated. As shown in the Figure 2.6, two density regions have been formed and points that are far apart of these two region by ϵ are considered as noise or outliers.

DBSCAN visits each point potentially multiple times, so without any form of indexing and optimization the complexity of the algorithm is $O(n^2)$. It does not require prior knowledge of the number of clusters and can find arbitrarily shaped clusters. Moreover, the ability of handling outliers and noise makes DBSCAN one of the top performing algorithms in cluster analysis. This thesis utilizes DBSCAN for its event

Algorithm 2.1 DBSCAN Algorithm [7]

```
function DBSCAN(D, eps, MinPts)
    $C \leftarrow 0$
    for all unvisited point P in dataset D do
        mark P as visited
        $NeighborPts \leftarrow regionQuery(P, eps)$
        if sizeof(NeighborPts) < MintPts then
            mark P as NOISE
        else
            $C \leftarrow nextcluster$
            expandCluster(P, NeighborPts, C, eps, MinPts)
        end if
    end for
end function
function expandCluster(P, NeighborPts, C, eps, MinPts)
    add P to cluster C
    for all  for each point P' in NeighborPts do
        mark P' as visited
        NeighborPts' ← regionQuery(P', eps)
        if sizeof(NeighborPts') >= MinPts then
            NeighborPts ← NeighborPts joined with NeighborPts'
        end if
        if P' is not yet member of any cluster then
            add P' to cluster C
        end if
    end for
end function
function regionQuery(P, eps)
    return all points within P's eps-neighborhood (including P)
end function
```

discovery task and it shows that parameters such as ϵ and $minPts$ can be very helpful specifically in the event discovery task.

2.7.4 Subspace Clustering

As Kriegel et al. [17] and Parsons et al. [24] have discussed that in a high-dimensional data set usually there are no global feature selection methods that can be applied to overcome the challenges of high dimensionality. Thus, they have surveyed different algorithms that known as subspace clustering. In such algorithms, different subset of features, based on their relevance and correlation, are selected to cluster the data. The challenge is to find a set of features that can find a fit cluster. In other words, "The general aim of clustering algorithms designed for high dimensional data is to find clusters in arbitrarily oriented subspaces of the original feature space." [17]
One main application of the subspace clustering in Web Text Mining is to detect keywords that are considered major concepts for certain websites. Then those keywords can be served for classification or information extraction purposes. Such discovery is possible to be accomplished through an automative process which uses subspace cluster in its heart [24]. There are three categories that subspace clustering algorithms fall into:

- **Axis Parallel Clustering** The main idea behind axis parallel clustering is to reduce the search space of all the possible subspaces to only axis-parallel subspaces [17].

- **Pattern-based Clustering** The main idea behind pattern-based clustering is to treat features and objects interchangeably.

- **Correlation Clustering** The main idea is to assume that any cluster can be located in an arbitrarily oriented subspace S of the data space Rd. Such clusters may appears in hyperplanes of arbitrary dimensionality in the data space but no typical pattern in the data matrix do corresponds to these models based on a spatial intuition.

2.8 Bursty Event Detection

In statistical sense, burst is an intermittent change in activity or frequency of an event. And bursty event in text mining, as implied in this research work and in literature [12], is a sudden increase in number of topically similar documents that contain information about an event. The major contribution of this thesis is to find possible relations between bursty events in time, therefore bursty event detection is the major component of this framework.

Kleinberg [16] proposed in infinite-state automaton to model the arrival times of documents in a streams to identify bursts that have high intensity over limited durations of time. The states of the probabilistic automaton correspond to the frequencies of individual words, while the state transitions capture the burst, which corresponds to a significant change in word frequency. However, this model assumes documents in the stream are all about one single topic which renders this model useless for applications tracking multiple topics.

Chang et al. [11] applied spectral analysis using discrete Fourier transformation

(DFT) to categorize features for different event characteristics (e.g., important or not, and periodic or aperiodic events). DFT converts the signals from the time domain into the frequency domain, such that a burst in the time domain corresponds to a spike in the frequency domain. However, DFT cannot identify the period of a bursty event. Therefore, they employed Gaussian mixture models to identify feature bursts and their associated periods.

Snowsill et al. [30] presented an online approach for detecting events in news streams based on statistical significant tests of n-gram word frequency within a time frame. An incremental suffix tree data structure was applied to reduce the time and space constraints required for online detection.

2.9 Story Link Detection

Story Link Detection(SLD) is the final task in a TDT system by finding link between the two group of selected stories. Unfortunately, SLD has not been studied extensively by the research community due to the unclarity of its applications [2]. Normally SLD is done by comparing returned value of a *similarity* function, as a result the main researches in this field is done on comparing different similarity function computations. Mei and Zhai [22] have developed a method for modelling the events using probabilistic mixture models and computing their similarities between the events (i.e. θ_1 and θ_2) by computing similarities of their mixture model as given by Formula 2.11. Moreover, in their research, they have introduced a new notion of *theme* and *theme evolvement* to the field by arguing that events may start to evolve into different subtopics that can all still represent a single event.

$$D(\theta_1||\theta_2) = \sum_{i=1}^{|V|} p(w_i|\theta_2) log \frac{p(w_i|\theta_2)}{p(w_i|\theta_1)} \tag{2.11}$$

2.10 Concluding Remarks

In this chapter we have provided an overview of the field of event discovery and tracking by explaining the underlaying algorithmic concepts in discovering events from document streams. With advances in technology and availability of information, analyzing and processing documents are vital to individual and businesses who have to make decision based on existing load of information. TDT systems are mainly focused on providing an abstraction over the pool of available documents in order to enable individuals to navigate and stay informed about what matters the most to their interest.

Due to the existing of variations in definition of what to be recognized as "event", different algorithms are presented for event-discovery in the previous sections. Event-discovery can be formulated as a clustering problem in which we are looking to find clusters that represent a real-world event. In this thesis, based on the existing work, we are going to present a new clustering framework for discovering event from stream of news articles build on top of the existing methods discussed in this chapter.

Moreover, in order to keep with the pace of availability of overwhelming information, it is natural for humans to connect dots by finding relations between events in the course of time. Since finding meaningful relations between news articles in course of time has not being studied extensively in the community of text mining, our main focus in this reach work is to provide a framework towards discovering events that potentially can cause some other relevant events in the future and most importantly discovering such relationships.

Chapter 3

The Proposed Framework

News agencies are in the business of informing societies about ongoing events and issues around the globe. Some stories slowly emerge such as discussions about new upcoming presidential election and reach a peak as they get closer to the election pool and slowly disappear. Such stories are known ahead of time and their occurrence in the news media is expected. In contrast, for some events there are no available foresight on when they are going to occur, and once occurred, they will eventually turn into a very hot and challenging topic that can have different consequences on the societies as a whole.
For example, a sudden announcement about an ongoing issue in Middle-East can upset different sectors of economy such as oil and petroleum companies, while it may increase the stock price of green energy companies such as Solar City. The sudden occurrence of such events in news is called bursty events that bare vital information to people and businesses. Moreover, it is very important for individuals to imply

relationship between different occurred events in past in order to come up with more accurate picture of tomorrow, which is know as the art of “connecting the dots”.
Connecting the dots between news events are in a high demand with given advances in technology and availability of large volume of information. It is very important to note that connecting the dots in terms of finding the relations between different events can be domain dependent or even use case dependent. Semantical relationships may need to be taken into consideration and a solid definition of “relation” has to be defined.
Despite the importance of application of systems that provide such useful insights (i.e. burst detection, and link relation discovery), due to the complexity of the system it is not readily available for general public use. We strongly believe the problem is in the sophistication of the subparts that such systems contain and their interaction with each other. Therefore, as a part of this thesis work, aside from improving bursty event detection and introducing the notion of link discovery, we provide a general framework for constructing a working TDT system.
In text mining, event discovery falls into unsupervised machine learning realm and clustering analysis is the standard practice. Aside from choosing the right clustering algorithm for burst detection and performing the bursty event discovery task, it is important to pay very close attention to the preprocessing stage. Due to the fact that text documents contains large volume of unique words, the feature set (a.k.a. the dictionary) can be substantially large and may contain features may corrupt the end result. Thus, feature analysis is the main task before proceeding to cluster analysis. In this thesis we are proposing a procedure for only selecting bursty features (i.e.

words) as we believe they help to detect bursty events with better accuracy.
It is important to take into consideration the different usage of words in different fields and take into account special terms when performing link discovery between events, however, such considerations usually are very resource intensive and can render the system to be very domain dependent. In order to make a general framework that could perform reasonably well in different domains, we resort to statistical measures to find similarities between words and to discover words that can be semantically similar. For example, “slide” and “drop” are words to explain lower price for a security in stock market. Even though their relation is not given to the system, the system can infer that “slide” and “drop” are potentially similar either because in some text documents are repeated together in high frequency or there is a intermediate word like “low” that can make this connection. As it will be explained later in this chapter, such methods may not be highly accurate, but given that they do not require any user input, the link discovery can perform at reasonably good level of accuracy.
In this research, we propose a general framework for new emerging systems towards bursty event extraction and story link discovery from online news articles. The main contribution of this work, aside from proposing the general architecture of such system, is the algorithms used in Bursty Event Extraction and Story Link Discovery which are discussed in the details on the following chapters.
Figure 3.1 shows the overview of the proposed framework and the sequence of the tasks that will be done on the stream of news articles in order to perform event extraction and event relation discovery. This chapter will give a brief description of each sub part accordingly.

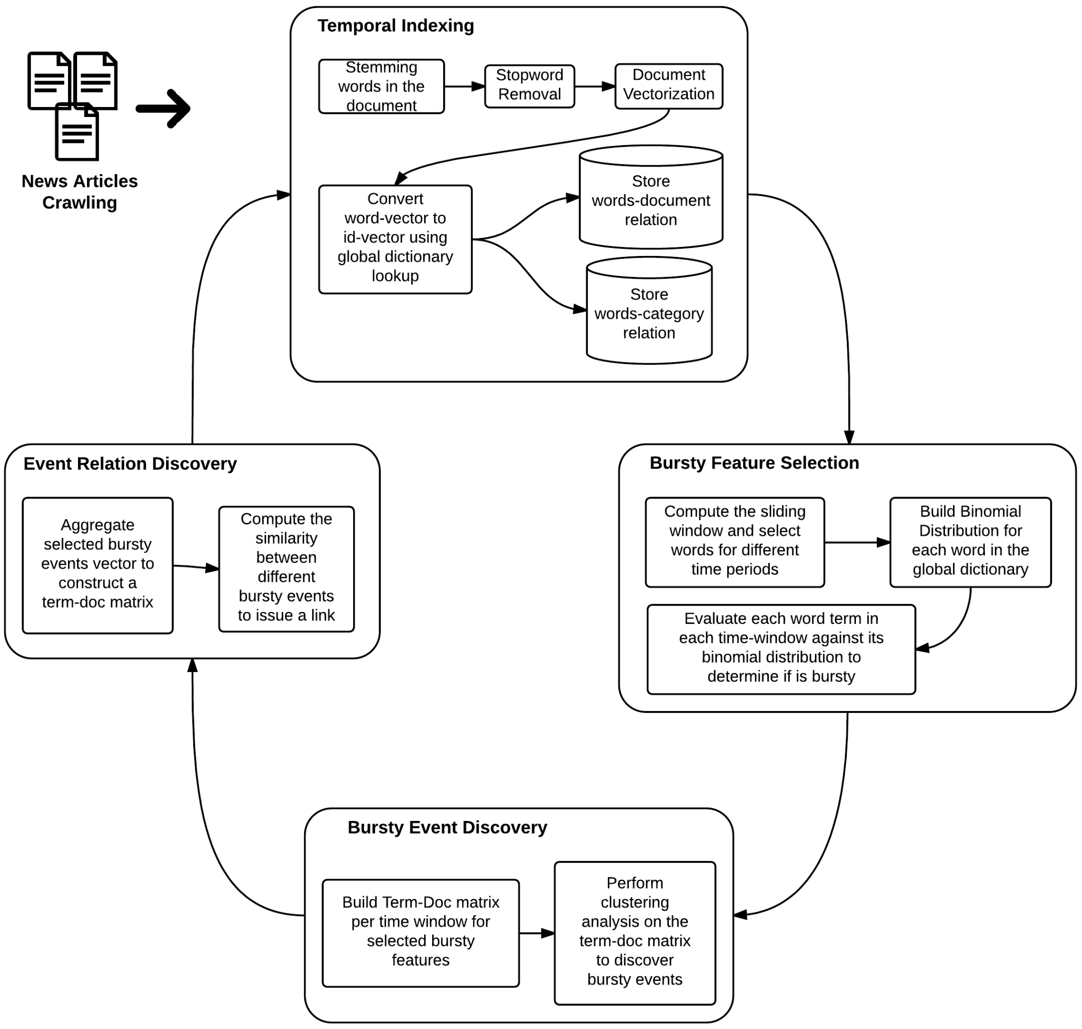

News Articles Crawling
Temporal Indexing
Stemming words in the document
Stopword Removal
Document Vectorization
Convert word-vector to id-vector using global dictionary lookup
Store words-document relation
Store words-category relation
Bursty Feature Selection
Compute the sliding window and select words for different time periods
Build Binomial Distribution for each word in the global dictionary
Evaluate each word term in each time-window against its binomial distribution to determine if is bursty
Bursty Event Discovery
Build Term-Doc matrix per time window for selected bursty features
Perform clustering analysis on the term-doc matrix to discover bursty events
Event Relation Discovery
Aggregate selected bursty events vector to construct a term-doc matrix
Compute the similarity between different bursty events to issue a link

Figure 3.1: Overview of The Proposed Event Relation Discovery Framework

3.1 Crawling

Crawling is the initial task in the subsystem of any TDT system. Crawling is the task of retrieving relevant resources and storing them for further processing tasks. For example, one can simply search every single news website for finding relevant news articles or simple use the *Google News*'s API for news gathering. Since this research work is for analyzing text documents, it is important that documents retrieved stored in text format. Therefore, it is possible to expand these framework to other sources such as news television programmings as long as there exists a way to translate and encode audio media into text for other components of the system to analyze.

3.2 Temporal Indexing

The novelty of this research work is event-discovery by selecting bursty features from the stream of news articles. Therefore, it is essential for a temporal indexing engine(TIE) to be in place in order to index news articles as they come and recording word frequencies on daily basis in order to building the temporal plot for each word. Due to the potential growth in the size of the news articles and vocabulary used in the stream, TIE has to be designed carefully by having such growth in mind. Moreover, in order to be able to discover the relations between current events with the events of past, TIE has to be capable of intelligent retrieval and fast pattern matching in order to boost the performance of the system and make the use of such system a reality. Figure 3.2 shows the storage schema for the temporal indexing which enables set of queries that are used for temporal data mining on text documents and is the essential part of the bursty event detection and link discovery.

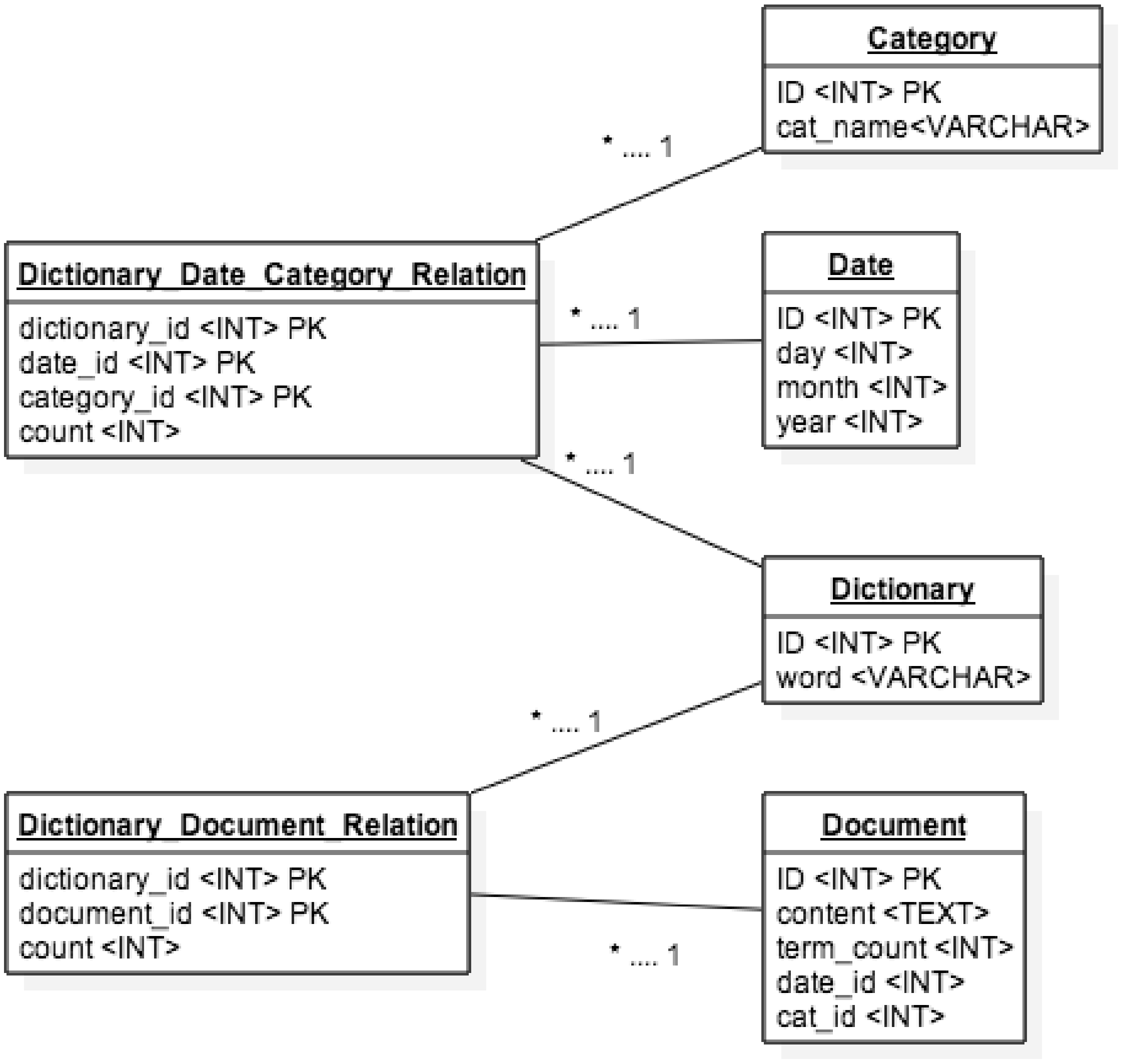


Figure 3.2: The overview of the database scheme for temporal mining

3.2.1 Document Vectorization

Documents are constructed from sequence of words and it is important to index documents based on the words that they contain. As explained in the previous chapter, the initial step is to break down the document into array of words that has build up the document and remove the unwanted characters such as comma and dot. Moreover, some words which are known as *stop-words* are required to be removed as they do not bare any important information and every single document in the set contains them. Words such as *determiner* are of that nature and are filtered out at this process stage. Doing so helps to reduce the complexity of the feature set in terms of volume and processing. It is worth to mention that stop word removal is an optional step and depending on the requirements can be switched off.

More sophisticated vectorization algorithms can be used depending on the usage of the framework. For example, terms such as “New Brunswick”, represents a location which consist of two words. A naive algorithm can break this concept into two unrelated words of “new” and “brunswick” and can lower the accuracy of the system. Rapid Automatic Keyword Extraction (RAKE) [27] is one of the popular algorithms that is capable of extracting terms of multiple words. RAKE first splits text documents into its sentences using sentence delimiter. Then, each sentence is broken into terms using stop words. Words grouped together after the break down are considered candidate keywords and by ranking them it will come up with the final keyword set. This algorithm is been suggested to be used as part of this framework.

Words can take on different forms such as singular vs plural and it is important to take such differences into account. As being discussed in the previous chapter, it is important to reduce all the words into their word stem in order to reduce redundancy in the feature set and increase the accuracy of the system. Both stemming and lemmatization can be used to accomplish this task. Using lemmatization requires the terms in the document to be tagged with their part of speech which can be a costly process and not required for all the applications. Therefore, we have used a rule based stemming API to accomplish this task.

3.2.2 Indexing

Once the text document being vectorized, the indexing process can take place. The objective of indexing are:

- Track what words are within each document.
- Having the ability to find all documents that contain a certain word or a set of words.
- Having the summery of the usage of a word in each day.
- Have the list of all the words being observed so far.

Table *Dictionary* in Figure 3.2 is the storage schema for storing all the terms observed. Once the document vectorization is done, all the terms in the vector set will be checked against this table and if the term already exists in the table, the term gets replaced with its *unique ID* in the vector set. Otherwise, the new term will be inserted into the table and receive a new unique ID and replacement takes place.
Once the vector of terms is replaced with the vector of IDs of terms(Algorithm 3.1), the indexing engine is ready to perform the indexing task. Depending on the category selected for the document to be index, document will be inserted into *Document* table and it will receive a unique *document id*. It is important to also count all the terms

Algorithm 3.1 Term vector to ID vector for indexing

```
Require: Dictionary_Table
 1: function TERM_TO_ID(term_vect)
 2:     for all term in term_vect do
 3:         if term exist in Dictionary_Table then
 4:             $term\_vect \leftarrow term\_id(Dictionary\_Table, term)$
 5:         else
 6:             $new\_id \leftarrow insert\_term(Dictionary\_Table, term)$
 7:             $term\_vect \leftarrow new\_id$
 8:         end if
 9:     end for
10: end function
```

in the document and store it as this value can be very useful for computing *TF-IDF* and can increase the overall performance of the system.

Table *Dictionary_Document_Relation* will capture the dependency of a word and a document. Vectorized terms id are now inserted into this table along with the newly inserted text document's id. As shown in the Figure 3.2, this table contains a *count* element that captures the number of times a certain term is repeated in the document. This count is very important for burst discovery and also will be used as the *TF* for building a *TF-IDF* model.

In order to have overall word usage history per day for each category, it is vital to store each word count per day that occurred in each category. Table *Dictionary_Date_Category_Relation* represent the schema for capturing this vital information. Using this table, one can find out how many times and what set of words are discussed per day in each category. In the indexing process, as more documents arrive, this table gets continuously updated for keeping the *count* element up to date.

By considering *category* in the indexing process, we are allowing the framework to capture and discover different bursty eventd for different set of documents. For example, “obama” could be a term that repeatedly is being used in the category of social issues due to the ongoing issue of health care for a long time. Thus, “obama” is not a bursty feature in that category because of being used heavily in past months. However, “obama”, at the same time can be considered a bursty feature in the finance category since there had been no occurrence of the term previously and the sudden occurrence can be due to a sudden burst of an event. As a result, introducing category based indexing allow such systems to distinguish between terms that are used in different categories and increase system’s accuracy.
Introducing the *date* component in the storing and indexing process, allows the system to look for the terms usage throughout a time axis and discover different usage patterns. Such temporal indexing can enable systems to perform text mining process through course of time instead of a fixed time set. Moreover, all the critical tables are getting updated in an incremental fashion which allows retrieval to be faster as the number of documents increases. In other words, such indexing allows *online text mining* algorithms to perform their task.

3.3 Bursty Feature Selection

The idea behind bursty feature identification is to generate a probabilistic model of $P_g(n_{i,j})$ for each feature f_i in a window w_j by counting the documents that contain the feature denoted as $n_{i,j}$. This model proposed by Fung et al. [11], estimates an unknown hyper-geometric distribution by performing the estimation on binomial distribution. As given in Formula 3.1, the probability distribution will model the success probability of occurrence of $n_{i,j}$ of feature f_i in a window w_j given the number of the documents in the window. Computing the binomial distribution is relatively fast compared to other form of hyper-geometric distributions, and the given architecture for storage component of this framework will enhance the computation as some of the critical numbers for each feature are pre-computed.

$$P_g(n_{i,j}) = \binom{N}{n_{i,j}} p_j^{n_{i,j}} (1 - p_j)^{N - n_{i,j}} \tag{3.1}$$

$$p_j = \frac{1}{L'} \sum_{i=0}^{L'} P_o(n_{i,j}) \tag{3.2}$$

$$P_o(n_{i,j}) = \frac{n_{i,j}}{N} \tag{3.3}$$

N is the number of the documents in the time window w_i and p_j is the expected probability of documents that contain feature f_j in a random time-window. Thus, the probability of observing a feature over all the time windows will be calculated and then average will be computed as given in Formula 3.2. L' is the total number of the windows, a parameter that needs to be configured in the system. As we will explain in the next chapter, we decided that each time-window will contain 5-days worth of documents and each window will have 2-days overlap with its previous window. These parameters are set experimentally and can vary for different domains.
Figure 3.3 shows a typical binomial distribution where probability of success is 0.5.

Algorithm 3.2 Bursty Feature Detection

```
Require: Dictionary_Table
 1: function COMPUTE_BURST_VALUE(term_list)
 2:     for all term in term_list do
 3:         $term\_day\_frequency[] \leftarrow$ FIND_TERM_DOC_DAY_COUNT($term$)
 4:         $term\_timewindow\_frequency[] \leftarrow$ SLICE_TIMEWINDOW($term\_day\_frequency, X$)
 5:         for all ftw in term_timewindow_frequency do
 6:             $N \leftarrow$ total number of documents in this time window
 7:             $n \leftarrow ftw$
 8:             $p \leftarrow$ EXPECTED_PROBABILITY($term$)
 9:             $binomial\_dist \leftarrow$ BINOMIAL($N, n, p$)
10:             $mean \leftarrow binomial\_dist'smean$
11:             $end \leftarrow mean + (3 * standarddeviation)$
12:             if  $n >= 0$ and $n <= mean$ then
13:                 return False
14:             else if $n > mean$ and $n < end$ then
15:                 $q \leftarrow float((mean + end))/2.0$
16:                 $slope \leftarrow 1.0/float(end - mean)$
17:                 $z \leftarrow (pg * slope) - q$
18:                 $burst\_probability \leftarrow 1.0/(1.0 + exp(-1.0 * z))$
19:                 return burst_probability
20:             else
21:                 return Bursty
22:             end if
23:         end for
24:     end for
25: end function
```

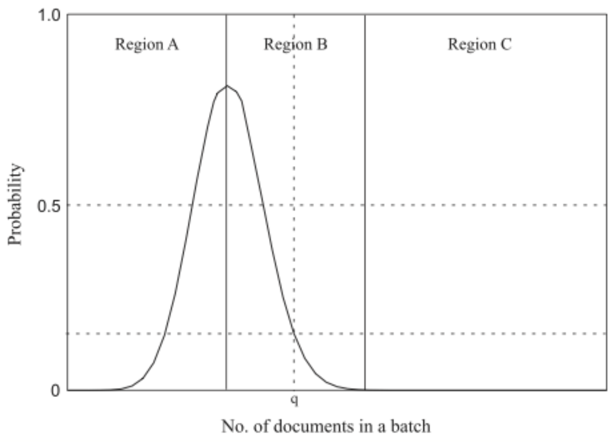


Figure 3.3: A generic view of a typical binomial distribution of the system [11].

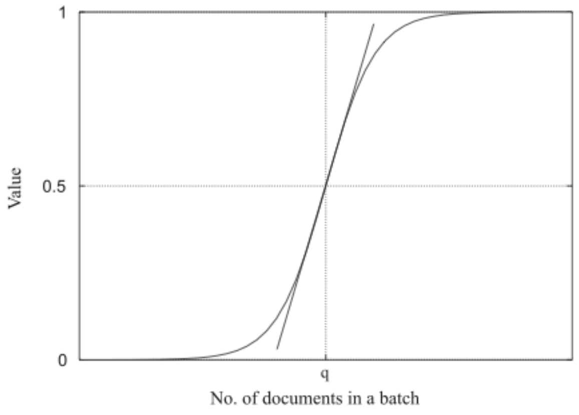


Figure 3.4: The sigmoid distribution [11].

This would represent a case where $p_j = 0.5$. As p_j grow, the graph will be shifted to the right hand side. As a result, words that are repeated very frequently in most of the windows will have a distribution shape skewed severely to the right and those words can be either *stop words* or words that do not bare important information for finding any sort of burst. It is worth to mention that using this method it is possible to implement an adaptive system for *discovering domain* specific stop-words

Algorithm 3.3 Bursty Feature Detection(continued)

```
1: function EXPECTED_PROBABILITY(term)
2:             ▷ Compute expected probability of occurrence of the term in all time windows
3:     $expected\_value \leftarrow 0$
4:     $term\_day\_frequency[] \leftarrow$ FIND_TERM_DOC_DAY_COUNT($term$)
5:     $term\_timewindow\_frequency[] \leftarrow$ SLICE_TIMEWINDOW($term\_day\_frequency, X$)
6:     for all ftw in term_timewindow_frequency do
7:         $expected\_value \leftarrow ((ftm/total_doc_in_tw) + expected\_value)/n + 1$
8:     end for
        return expected_value
9: end function
```

for reducing dimensions in text mining applications, hence increase the performance and accuracy.

Note that the binomial distribution in Figure 3.3 is divided into three different region. Region A is from 0 to x where $P_g(x)$ is maximum and Region B is from the peak to the point where it return back to 0. If the right-hand side of the distribution never reaches back to 0 it is concluded that the feature is one form of a *stop-word* and can be ruled out as not bursty feature since is being repeat in large volume in most of the time windows.

There are three different scenario to consider in order to rule out a feature as bursty in a given time-window after its binomial distribution is computed:

- When $n_{i,j}$ is in Region A, suggest that probability of the feature in W_i is less than or equal to the probability f_j is drawn randomly. Therefore, it is not a bursty feature.

- When $n_{i,j}$ is in Region B, depending on its location in R-B can be considered as bursty feature. If it is more closer towards Region C that it has higher probability of being bursty whereas as is closer to Region B is less likely. Therefore, by applying a sigmoid function as shown in Figure 3.4 we can give a weight to the burstiness of the feature. In this scenario, a threshold can be set and features with smaller weight are filtered out.

- When $n_{i,j}$ is in Region C implies that the value is higher than the prior probability and is exhibiting an abnormal behaviour and it will be detected as a bursty feature in W_i.

Once the binomial distribution for each feature in the feature set is computed over the selected periods, then each feature within each window will be tested with their own binomial distribution in order to find out if their occurrence on a given window is bursty or not.

3.4 Event Discovery

Event discovery is done using clustering the selected bursty features from the bursty-feature discovery phase. There are different methods to select a subset of features as we discussed in the previous chapter. We argue and show later that selecting *bursty features* from previous phase will increase both precision and recall of the system. As the features are selected for each time-window in the previous phase, bursty events are defined by clustering the selected features. As explained in Section 2.7.2, the idea of clustering features instead of documents is called feature-pivot clustering and its advantages have been discussed in Section 2.7.2.

Based on the documents in a selected window(W_i) a term-document matrix for the bursty features will be constructed in which each bursty feature is represented using a vector consist of documents in the W_i. Let $f_{i,j}$ be a randomly selected bursty feature in time window W_i then $f_{i,j} =< d_{i,1}, d_{i,2}, \dots, d_{i,k} >$ where $d_{i,k} = 1$ if document k in ith time-window contains the selected feature f_j, otherwise $d_{i,k} = 0$. The term-document matrix is used to compute similarity, as discussed in Section 2.5, between different features for clustering purposes. Moreover, term-document matrix is capable of capturing implicit relationship of terms which in some cases can be as accurate as manually providing such relations to the system.

Once the feature representation is accomplished through constructing the term-dcoument matrix, using a clustering algorithm, feature-pivot clustering can take place. In this thesis, the DBSCAN is the major clustering algorithm that is being used due to its performance and clustering capabilities. DBSCAN, as discussed in section 2.7.3 is capable of clustering different shapes in space which made a logical scenes to be used in this project as text documents can have different unconventional form in space once represented in vector space model. It is worth mentioning that DBSCAN is open to different similarity measures functions and a wide range of similarity functions can be used for computing distance between features.

The result of clustering algorithm is the list of potential bursty events. In order to increase the reliability of the system each of the clusters are evaluated using silhouette score. By computing mean intra-cluster distance of elements in a cluster it helps to measure the density and similarity of objects in a cluster among other clusters. *Silhouette Score* vary from -1 to 1, any value bellow 0 is not acceptable and one can

Algorithm 3.4 Building time window bounded bursty term-doc matrix for DB-SCAN

```
Require: Dictionary_Table
 1: function BURSTY_EVENT_DISCOVERY(bursty_terms)
 2:     term_doc_matrix[][] ← init to zero
 3:     for all term in bursty_terms do
 4:         docs_count[] ← load id of documents contained this term
 5:         for all i in docs_count do
 6:             term_doc_matrix[term][i] = docs_count[i]
 7:         end for
 8:     end for
 9:     bursty_events[] ← DBSCAN(term_doc_matrix, epsilon, minPt) return
    bursty_events
10: end function
```

conclude the discovered cluster is meaningless while positive values represent quality of the formed cluster. Not only Silhouette Score can be used for evaluation of our framework it also serve as an online monitoring measure of the running system and giving the possibility of dropping less assuring and noisy cluster without reporting them and also reporting a interval of confidence on the discovered clusters.

3.5 Event Relation Discovery

Once events are discovered from Event Discovery phase, relation of the recently discovered event should be studied with the discovered events in the past. Since events in the past can have causal relationship with the recent discovery, discovering such link will allow to see the bigger picture from the past in order to predict the future. Moreover, some bursty events that may happened over the course of time can be grouped together in order to represent a bigger silent ongoing issue. For example, different bursty events regarding Tesla Motors and SolarSystem through out the time can shed lights on the bigger topic known as *Clean Energy* that may have never been discussed boldly in the news stream yet again its affect can be felt in the industries as a whole.

There are different ways to go about this problem and define different type or relationships between events (i.e. causal vs topic). Since the problem is discovering the relationship between different text clusters in time, we have provided our own definition and approach in order to introduce the importance of such concept in academia and create a foundation for further improvements. Since the event discovery process is one form of feature-pivot clustering, and since the features are represented using vector space model, it is possible to construct a vector representation for the discovered events by aggregating their underlaying feature vectors.

Algorithm 3.5 Bursty event vector aggregator

```
1: function EVENTVECTOR(features[])
2:     $vectorSize \leftarrow length(features[0])$
3:     $eventVector \leftarrow array(vectorSize)$
4:     for all  i from 0 to length(features) do
5:         for all j from 0 to vectorSize do
6:             if features[i][j] = 1 then
7:                 eventVector[j] ← 1
8:             end if
9:         end for
10:    end for
11: end function
```

Once the bursty event's vector representation is constructed by aggregating its underlying feature vectors, it is ready to be compared with other event vectors for events that discovered in the past. However, since it is possible for some words to occur in only certain time windows and not all, it is very important that the framework maintain a global dictionary for constructing a global term-document matrix. Computing and maintaining such matrix can be very expensive in long run therefore we have proposed a quick fix to this problem to preserve the speed and maintainability of the framework.

Once the vector of a bursty feature is constructed, a group of past events is selected for link discovery, and as a result it is possible to aggregate all the words in different bursty events to build a encompassing sub-global dictionary of the selected bursty features and reconstruct their vector representations accordingly. Once all the selected and the new discovered bursty event are transformed into the new vector representation they can be compared against the new discovered bursty event using a similarity function such as *cosine-similarity* and if the similarity value was above the set threshold a relation link will be created.

This still remains an open problem for the type of the discovered relationship, and at the moment it is up to the user to analyze further to find the causal relationship between the two identified related events. However, as we will discuss more in the next chapter, this is an interesting challenge that this research work will be proposing to the research community.

3.6 Visualization

Visualizing the discovered bursty events and the graph of related events is the final step that allow users to interact with the system. Due to the availability of different platforms (i.e. web, mobile, etc) we are not going to suggest any particular visualization technique. In this framework we decided to output an intermediate file that represents bursty events in each time window as “nodes”. Edges between any two related events shows a relation between the two and capturing the weight of the relation as an attribute of the edge. As a result, the intermediate file will contain description of “graph” that any graphing tool software is able to open and allow the user to interact with the graph. For the testing purpose of this thesis work, we decided to use *GDF* file format for storing our event graph. GDF simple syntax is easy to understand and requires small storage space for a relatively large graph.

3.7 Concluding Remarks

In this chapter we provided the overall architecture and methods that are used in developing bursty event extraction and link discovery framework. Temporal Indexing is the stepping stone for creating bursty event discovery system and in this chapter we described in details the design and implementation of this subsystem from different aspects. Temporal Indexing Engine needs to take into account the rate of growth of the system and therefore we presented design notes that helps the system to expand as efficiently as possible.

Moreover, we presented our algorithm for burst detection in order to reduce the number of available features into a smaller but useful list of vocabularies. There are different statistical methods such as *Principal Component Analysis* that helps to reduce feature sets. However, our suggested method perform reduction task by taking into consideration the usage pattern of features in course of time. Thus, we argue that this method is an *informed* reduction method well suited for event discovery applications.

We explained how a new term-document matrixes will be constructed for each time-window only based on the selected bursty features. This allows that any suitable type of clustering algorithms get used for event discovery task on the term-document matrix. We chose to cluster bursty terms in each time window as we discussed the advantages of *feature-pivot* clustering in Section 2.7.2.

We covered the details of how to compute links between discovered events in different time windows using different similarity functions. This allows links with different thresholds to be formed and identified in order to be analyzed further by human agents. The main contribution of this thesis is to provide a method to identify such links between different events separated by a time interval and help human agents by proposing such leads to be investigated further. In the next chapter we discuss details of implementing the explained framework and the retrieved results.

Chapter 4

Implementation and Evaluation

4.1 Data Set and Crawling

For the purpose of validation and testing the proposed framework, using a custom designed news crawler, we have downloaded over *12,000* news articles between *January 14th, 2014* to *March 9th, 2014* from news agencies that are listed in Table 4.1.

URLs in Table 4.1 point to the RSS feeds of the news agency websites, however, since the RSS pages did not contain the full story of the news article, our crawler retrieved every single news story page and striped out the main content from the body of different websites. By downloading the complete story and trying to remove as many boilerplate data as possible we tried to build a more solid and reliable dataset. Even though the efforts have been put in the crawling task to reduce the number of noisy inputs into the system, some of the news article contained some noisy data that our boilerplate remover could not eliminate. The interesting point is

Table 4.1: List of news websites used in the dataset.

CNN Money `http://rss.cnn.com/rss/money_latest.rss`
WSJ Business `http://online.wsj.com/xml/rss/3_7014.xml`
Reuters Business `http://feeds.reuters.com/reuters/businessNews`
Livetradingnews `http://www.livetradingnews.com/feed`
CBS Business `http://feeds.cbsnews.com/CBSNewsBusiness`
Washingtonpost Business `http://feeds.washingtonpost.com/rss/business`
Euronews Business `http://feeds.feedburner.com/euronews/en/business`
USAToday Money `http://rssfeeds.usatoday.com/UsatodaycomMoney-TopStories`

with the existing noise in some of the news article, our framework still managed to perform in an acceptable range. This fact could be the major motivator for proposing the adaptation of this framework in real-world applications such as opinion minion for companies or brands.

As shown in Figure 4.1 the number of documents vary from day to day. Moreover, in some days due to failures in our computer systems, our crawler was unable to download enough document and as a result such days, that were not very frequent, are removed from the analysis and are not shown in the graph.

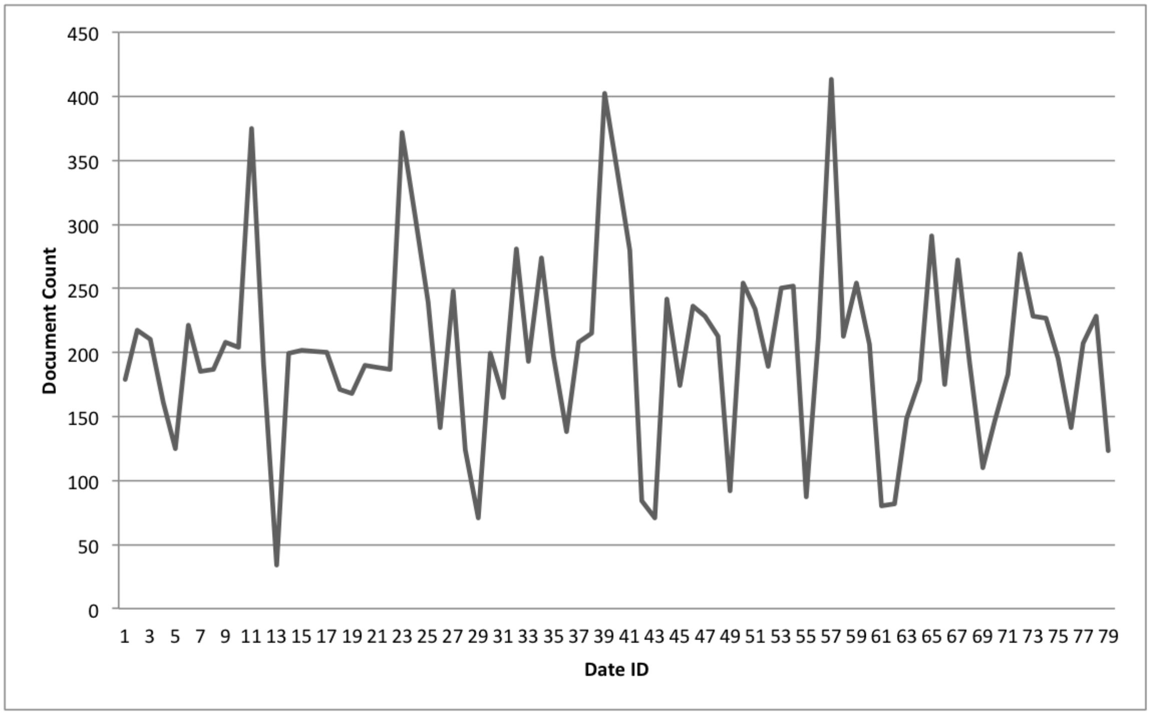


Figure 4.1: Distribution of documents per day

4.2 Bursty Event Discovery

Due to the nature of news propagation and occurrence, the bursty detection algorithm will group period of days together before analyzing. In this work we are using Sliding Time Window approach to group documents in five consecutive days (slide duration=5) together with slice delta of 3 days. Such slicing will allow for stories that are developing in a short period of time (i.e. few days) get detected by our framework.

Once the documents are aggregated using the sliding time window techniques, then each term's frequency graph will be studied in the chosen time-window in order to determine if the term is bursty. Once the bursty features are detected, only those

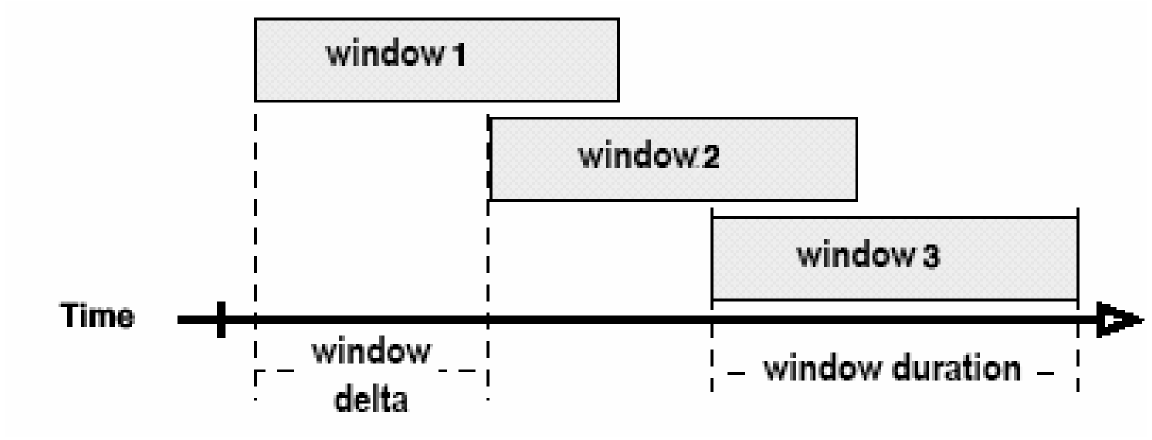
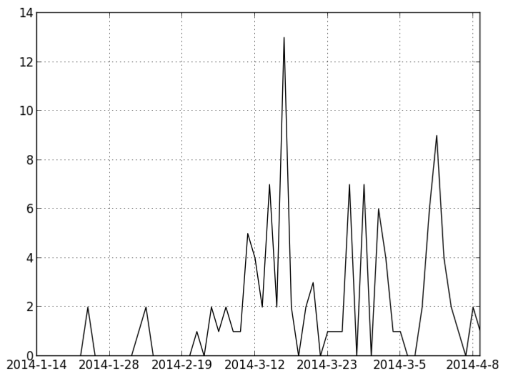


Figure 4.2: Sliding Time Window

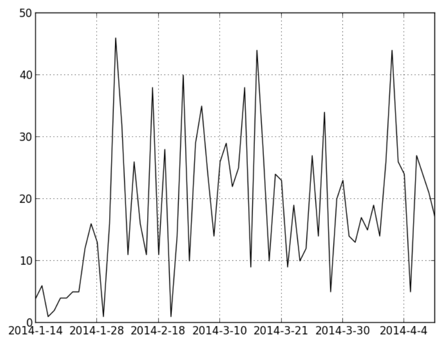
features will be used to representing documents in the selected time window and will be used for clustering purposes.

Table 4.2 shows some of the bursty events that have been detected by the system in different periods of time for illustration purposes. Due to the limited available space, we only selected a couple of examples to show the points that bursty events can be very important and since our dataset consist of news related to finance and business, the bursty events that has been detected (i.e. Microsoft and GM) can help investors to take action accordingly and generate leads for their investments.

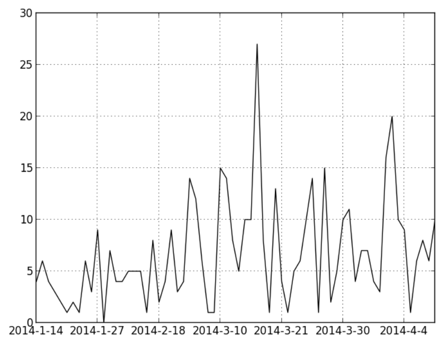
Figure 4.3 represents the frequency graph of the words that has formed *GM*'s bursty event on *airbag* and *ignition switch* on *24 Feb to 29 Feb*. What Figure 4.3 is revealing is that since these words can be used in different context, they are having very different frequency time series graph, however at some point in time (24 Feb) they formed an bursty event to identify GM's issue. It is clear that comparing the frequency graph between words cannot alone be a clear metric for grouping words



(a) airbag



(b) switch



(c) crash

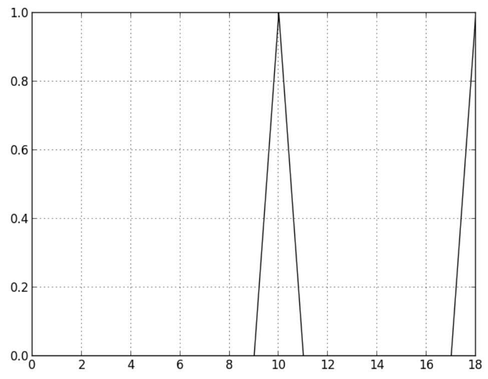
Figure 4.3: Frequency Time Series of the Bursty Words

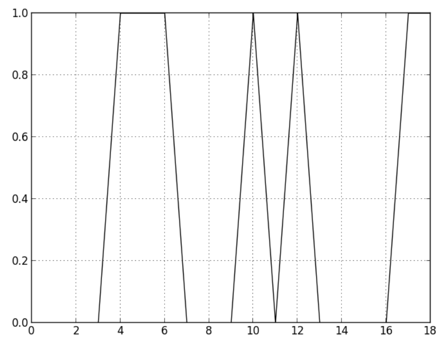
Table 4.2: Example of discovered bursty events

Burst Period	Bursty Featurs	Description
14 Jan to 19 Jan	oil, channel, crude	“Canada Loses Patience on Pipeline, Tells US to Decide” which is representing a bursty event about ongoing talks between Canada and US on the pipeline project.
23 Jan to 1 Feb	snow storm, ice	Ice that was formed on the Greate Lake was causing hazard to ships transporting goods and due to press release of coast guards received a media attention.
24 Feb to 29 Feb	airbag crash, switch	GM had issued a recall for vehicles that had airbag and ignition switch problems that had led to death of more than 19 people.
13 March to 19 March	dc, outset, reaction ukrain	Ukrain crisis was starting because of the upcoming referendum and discussion of joining Russia.
21 March to 26 March	cargo spill, vessel	The Malaysian Flight 370 had gone missing and there were some spills mark on the sea and vessels were launched to look for any survival.
2 April to 7 April	box cortana microsoft	Microsoft launched a virtual assistance Cortana in response to Apple’s Siri.

to detect bursty events.

Figure 4.4 shows the result of burst-detection algorithm ran through the frequency time-series of the words in Figure 4.3 and the result is easier to interpret as the plot only shows periods where the words were detected to be bursty and their probability of burstiness that was calculated using the sigmoid function. It should be noted that burst-graphs are more compact and therefore storing such graphs for huge number of words is less costly compared to the corresponding raw frequency graph. Moreover, since there are less data points on the burst-timeseries graph, it is faster to compare


(a) airbag


(b) switch

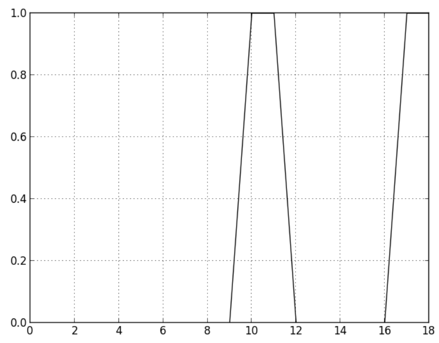

(c) crash

Figure 4.4: Burst Time Series

two burst time series than to compare two frequency time series which increase the performance as a result. Additionally, frequency time series does not reveal the importance of occurrence of a typical word in a period with respect to other periods and as a result most of the points on the frequency time series graph can be useless and possibly noise as they do not bare any important information. In contrast, burst time series graph is the result of comparative analysis of the occurrence of a word in different periods.

4.3 Link Discovery and Visualization

As explained in Section 3.5, once the clusters are formed within each time-window the next step is to find potential link between discovered clusters. Due to the performance issues, it is not recommended to look back to all the available time windows but only by a fixed number. However, since our dataset contains only three months of news articles and we have a total number of 17 time windows, it was reasonable to check all the previous time windows for each discovered bursty event to find potential relations.

Figure 4.5 shows a complete run of the link discovery engine of the framework on the discovered bursty events. By laying out the bursty events in each time window vertically, links can be drawn vertically to other bursty events that are related. Since the relational link is the result of computing similarity between two bursty events that will have result between 0 to 1, the strength of each link will be dictated by their computed similarity value. By computing source and destination nodes and relation

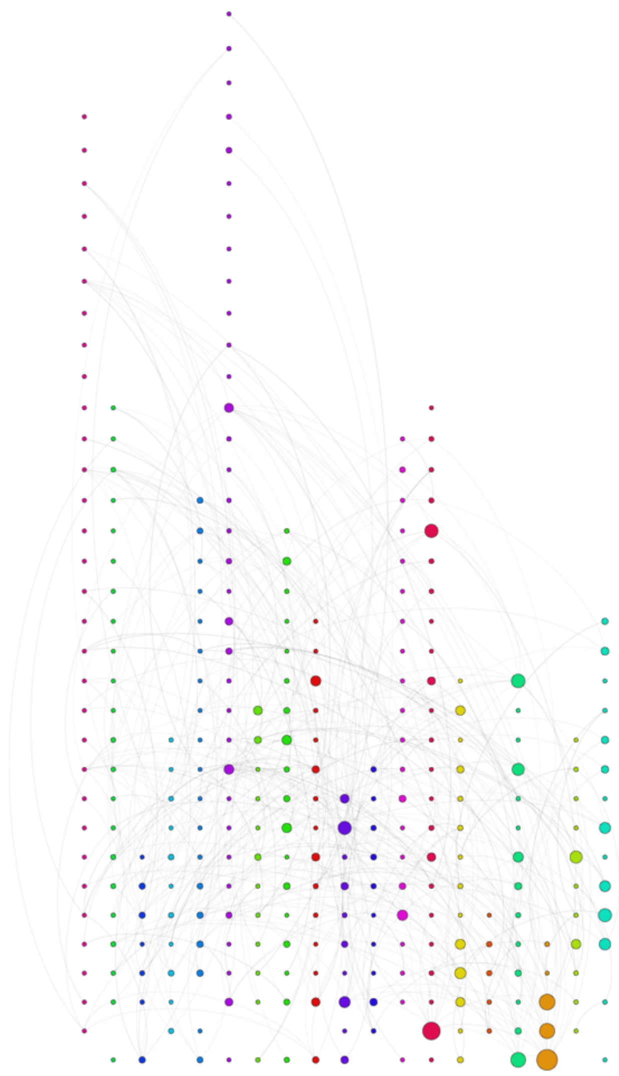


Figure 4.5: Discovered Event Graph

strength values in an XML file, we visualized the result using *Gephi*. Capturing strength value, allowed us to dynamically filter out links with smaller strength value to come up with a cleaner and readable relation graph. Of curse generating the *XML* file containing the bursty events relation information enables us to have different visualization user interface implemented on different platforms such as mobile or web.

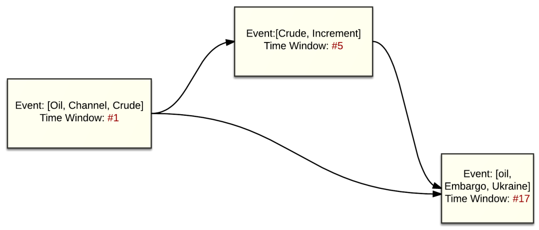


Figure 4.6: Example of three different related bursty events

Figure 4.6 shows a close up view of one of the related events from Figure 4.5. As it is shown a general discussion about speculation on oil price regarding Ukraine potential crisis has been turned into real crisis of oil price increase and tension in Europe and sanctions(Embargo) Russia for its actions regarding Ukraine.

4.4 Validation and Discussion

What is note worthy about deciding to perform feature-pivot clustering in contrast of document-pivot clustering, other than allowing each document to contribute to more than one bursty feature discovery, is the fact that formed clusters are set of bursty terms that very well can explain what the cluster is representing. Cluster labelling or clustering naming has been one of the important issues in text mining as they reveal important characteristics about supporting documents they contain, and our framework makes the process easier and closer to automated cluster naming.

4.4.1 Silhouettes Score

Silhouettes Score [28] is a score given to a cluster between -1 to 1 measuring the quality of the formed clusters. Clusters with negative Silhouettes Score are mathematically inconsistent while positive values describe the quality and density of the formed clusters.

Let i represent a point in a cluster, then $a(i)$ is the average dissimilarity between point i and all the other points within i's cluster. Dissimilarity will be measured using the same similarity function used in the clustering process. Let $b(i)$ be the lowest average dissimilarity of i to any other cluster which i is not a member. Then Silhouettes Score will be defined as given in Formula 4.1.

$$s(i) = \frac{b(i) - a(i)}{max\{a(i), b(i)\}} \tag{4.1}$$

Thus, the average $s(i)$ over all data of a cluster can serve as a measure of how tightly the data points are within the cluster. Therefore, by computing Silhouettes Score for each of the detected bursty events (a.k.a clusters) we assign a quality value for interpretation purposes. Moreover, by averaging all $s(i)$ within a period it is possible to measure the performance of the system over a period of time.

4.4.2 Validation

Discovering bursty events and their relation in course of time requires a large dataset that span over number of months in order to be adequate for performing such processing. Therefore, number of the available documents and topics potentially will exceed human's ability to read all the individual documents and tag them accordingly. Therefore, for validation purposes we decided to use an unsupervised mathematical notation for post clustering analysis to find the quality of discovered clusters and selecting a subset of discovered bursty events and their relations for manual validation. Using not only Silhouettes Score allowed us to measure the quality of the discov-

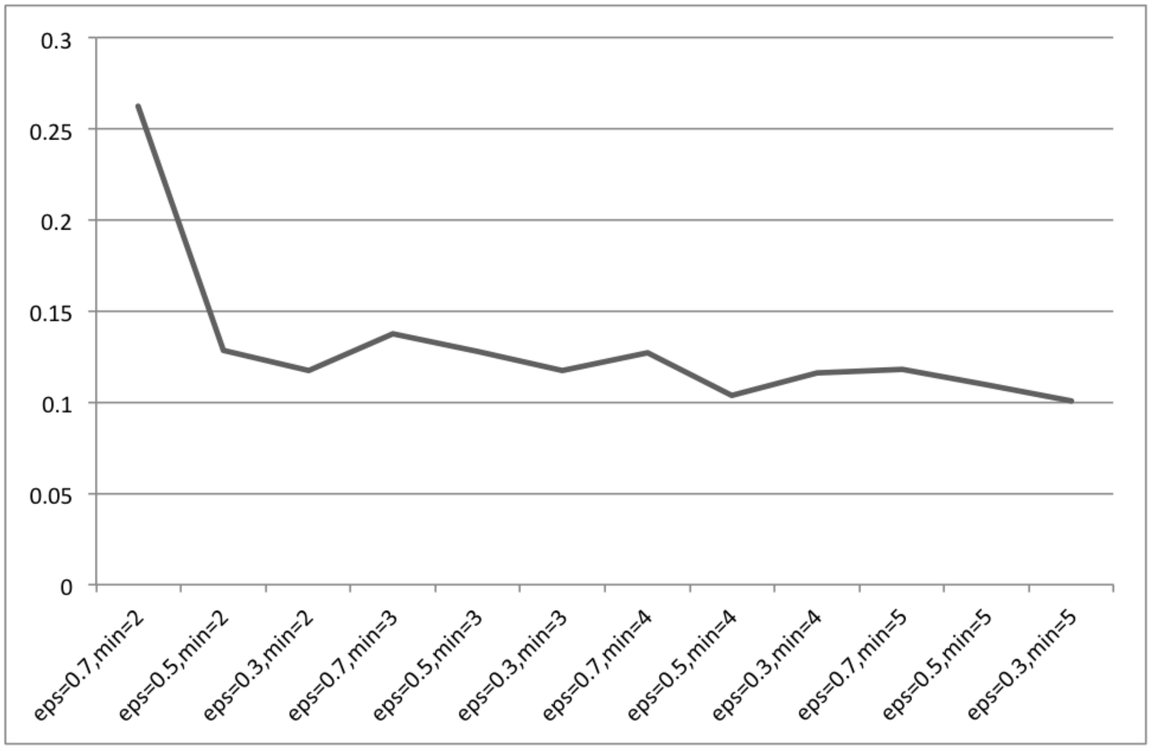


Figure 4.7: DBSCAN different parameters setting performance

ered clusters but also became part of the online feedback measure for our running framework to report the performance of the framework under operation.

Moreover, DBSCAN uses two different parameters for performing clusters known as *epsilon* and *min-size*. Such values mostly are set experimentally and using Silhouettes Score allowed us to choose values that maximize the performance of our clustering

Table 4.3: Discovered Bursty Events Accuracy

Burst Period	Number of Discovered Bursty Events	Window's Accuracy	Correctly Discovered (# of True Positive)
Windows #1	32	62.5%	20
Windows #2	23	60.86956522%	14
Windows #3	8	62.5%	5
Windows #4	12	58.33333333%	7
Windows #5	20	60%	12
Windows #6	35	60%	21
Windows #7	13	61.53846154%	8
Windows #8	19	63.15789474%	12
Windows #9	16	68.75%	11
Windows #10	10	60%	6
Windows #11	11	72.72727273%	8
Windows #12	22	72.72727273%	16
Windows #13	23	65.2173913%	15
Windows #14	14	64.28571429%	9
Windows #15	6	66.66666667%	4
Windows #16	14	64.28571429%	9
Windows #17	5	60%	3
Windows #18	12	66.66666667%	8
Windows #19	16	68.75%	11
Average Accuracy		64.15662913%	
Standard Diviation		4.28	

algorithm. As a result, Silhouettes Score is working as a feedback loop into our clustering algorithm in each time-window to come up with the best parameters feet. Figure 4.7 shows different Silhouettes Score for different parameter settings and as is shown on the graph, $epsilon = 0.7$ and $min = 2$ worked the best on our dataset. And as discussed Section 4.4.1, Silhouettes Score of 0.27 is in an acceptable range showing the formed clusters are mostly dense.

Table 4.3 is summary of the discovered bursty events in each of the time windows and their accuracy. Each discovered bursty event is been checked against the supporting documents news articles that contained the terms of the bursty event in the time-window. If we did not convinced there is a relation between the terms that have made up the bursty event, we would mark the detected bursty event as false positive. The total accuracy of each time window is calculated as the total number of discovered bursty events in a time window divided by the total of the *correctly* (True Positives) discovered bursty events in that window. As a result the average accuracy of the system on our test data is $64\% \pm 4$.

4.5 Summary

The proposed framework implemented as a full functioning system which downloaded and analyzed more than *12,000* Economic news articles from different sources. Downloaded news articles, due to the nature of HTML pages, contained some degree of unrelated words.

The system managed to index all the downloaded documents and perform bursty feature extraction algorithm to detect bursty features. Later, using our proposed event detection algorithm we performed the clustering task of bursty features by constructing the term-document matrix of bursty features and representing news articles only based on their bursty features. Then, the discovered bursty events were tested against all the previous bursty events for potential link discovery.

Due to the required size of the dataset, it was not possible for all the 12,000 documents to be reviewed and labeled by human operator. Thus, bursty events were

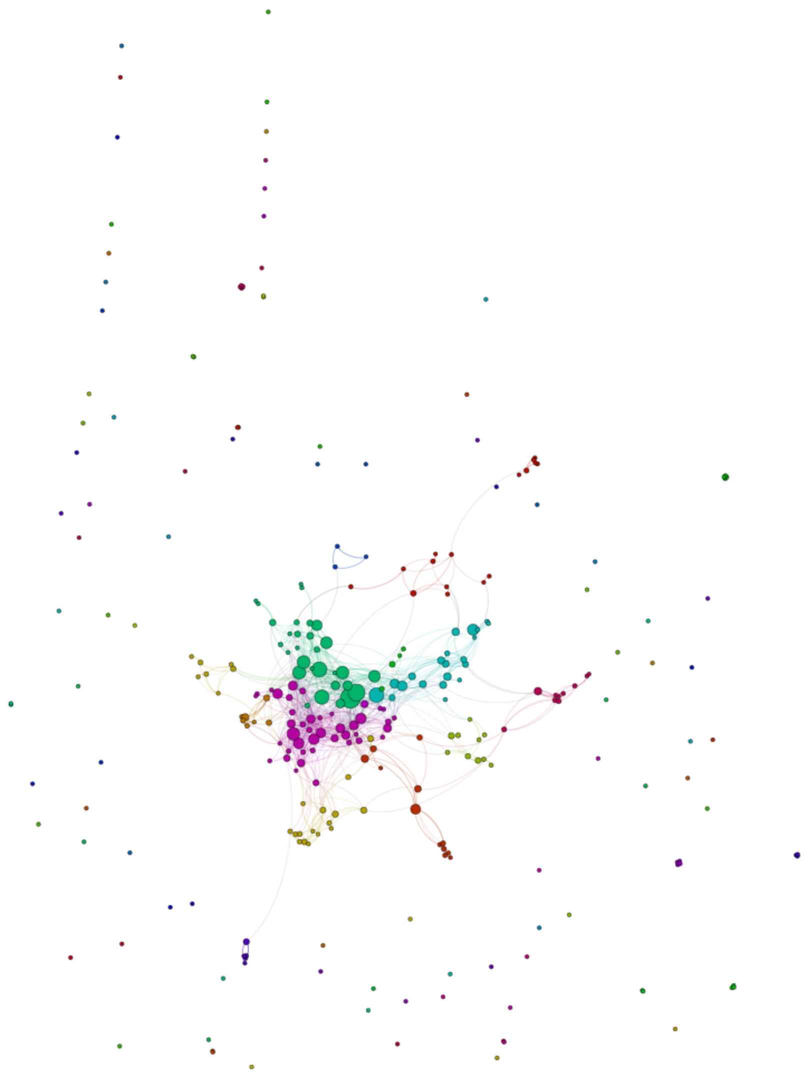


Figure 4.8: Clustered Event Graph

randomly selected and their supporting news article were reviewed to confirm their correctness which on average had 64% accuracy. It is worth to mention, that the level of the accuracy is for the full functioning system in which none of the dataset items were hand picked and cleaned. In other words, the system's reported accuracy is based on an unfiltered real world dataset.

The event graph is being constructed from the discovered bursty events and their relations as shown in Figure 4.5. Later, to test the hypothesis that discovered event graph may contain some useful information such as hidden global ongoing issue, we performed a community discovery algorithm on the graph. The result, as shown in Figure 4.8, revealed event a sequence of related events can get broken down to some smaller but related sets that may explains some ongoing issues that never being discussed clearly in the news articles but nonetheless their foot print can be seen from cluster of related bursty events. Exploring further on this ground requires an extensive research which is out of scope of this work and we can raise this question for future works that can be on done based on this research work.

Chapter 5

Conclusion and Future Work

5.1 Conclusion

The main contributions of this thesis, was proposing a novel approach to discovering bursty events from stream of news articles and discovering potential links between bursty events. The proposed framework takes into account the challenges that exist to perform temporal text mining on a large document dataset and provided details of subsystems for indexing and discovering tasks.

A proper indexing is one of the key tasks that needs to be done in order to boost the performance of the system and enable the system to perform on near real-time basis. By pre-computing some of the measures upon retrieval of a document and creating appropriate relations, algorithms can aggregate their data faster and provide result much quicker. For building the global dictionary during indexing, we proposed RAKE keyword extraction algorithm as it helps to capture keywords that consist of more than one word separated by space.

We proposed an informed method of feature reduction by taking into account the historical usage of the terms in the news article streams. Our proposed feature reduction method, detects bursty features by studying usage trend of terms and detecting abnormal spikes in their usage trend in order to nominate them for clustering. We showed that discovering bursty features and relying on them for bursty event discovery task is a reliable and accurate method and proven to work even on a noisy dataset.

We constructed a new term-document matrix only using the discovered bursty features that was detected. Then, we used DBSCAN clustering algorithm because of its power to cluster irregular-shape data points. Even though the kind of clustering algorithm is a matter of taste and relevancy of the task in hand, we decided to utilize DBSCAN for this task as it enabled us to perform *dynamic* clustering and not worrying that are data points may form irregular shapes in the term-space. Moreover, DBSCAN parameters allowed us to reason more about the quality of the formed clusters in terms of distance from other clusters and then minimum number of terms to be contained in each cluster.

We proposed an aggregation process for building a vector representing a discovered event using the term vectors that the event contained. This allowed us to use conventional similarity measures such as cosine similarity to measure the distance between discovered events and reason about their relation. It is important to note that using *vector space model* for modelling events allowed us to find similarity between events that do not share the same set of feature terms. We used the similarity value returned by the similarity function as the weight of relation between any two related events.

We proposed the graphing visualization technique as the main way to visualize the discovered bursty events and their relations. Thus, the output of a system is a generic graph description file that can be viewed with any major graph visualization software. This allows the system operators to navigate through the event graph and see many relations all in one unified picture. Moreover, this enable researchers and operators to investigate the discovered event graph more in depth using graph and social network analysis techniques.

Aside from human interoperation, we have used *Silhouettes Score* as a way of measuring the quality of discovered clusters. Moreover, since Silhouettes Score is computed automatically by the system, it is used as an online performance monitoring measure of the system.

Due to the lack of studies and available research works in the field of *bursty event discovery* and *bursty event relation discovery*, we were unable to report a comparative study and we only relied on our own validations. Given our real world noisy dataset, our system had average accuracy of 64% on discovering the bursty events.

5.2 Future Work

The potential application of this research work is endless and thus different part of this framework may need to be modified according to the applications need. However, we will discuss some possible improvement and additions that we think is possible to be made to our proposed framework that can be useful in any application usage.

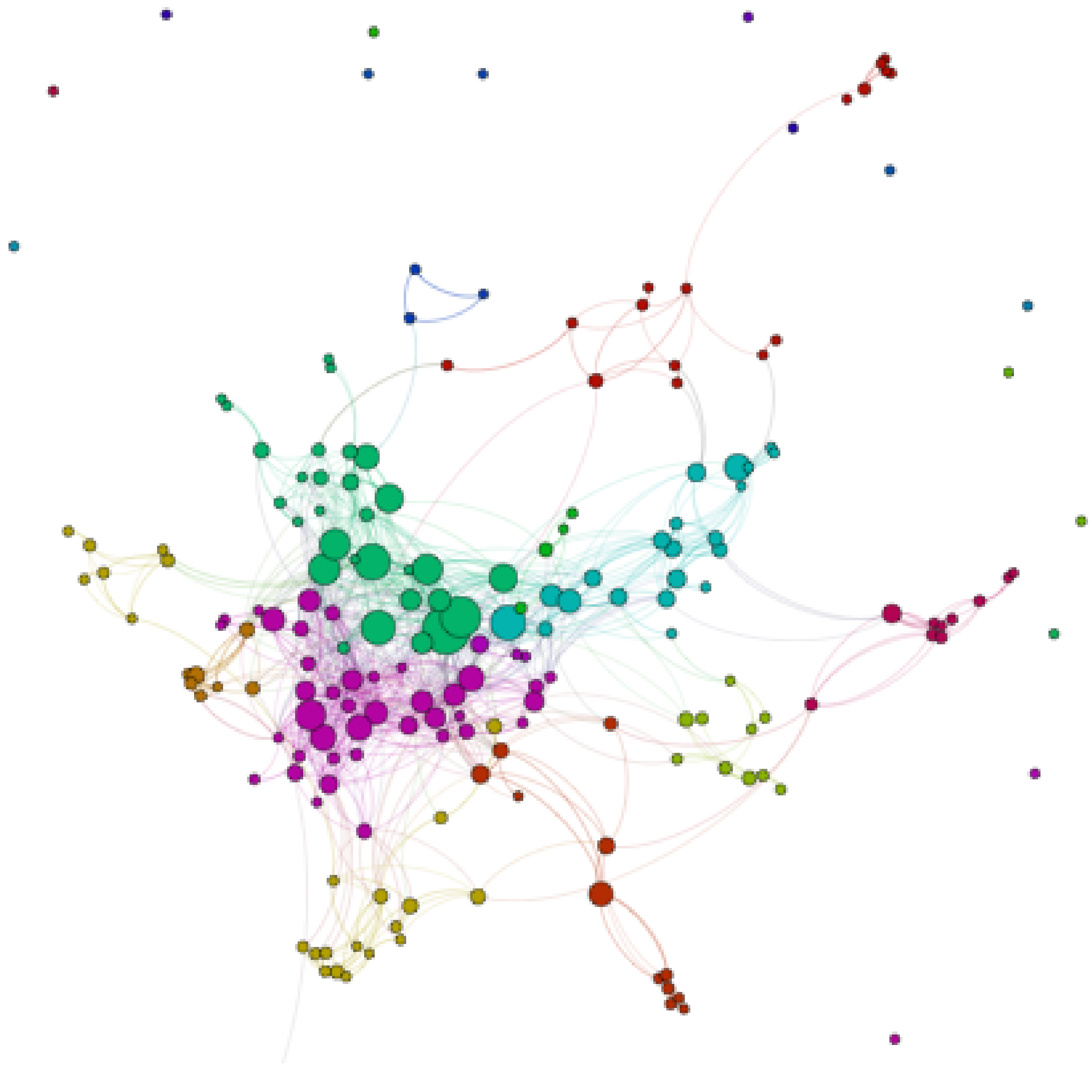


Figure 5.1: Clustered Event Graph

Currently, the computed *Silhouettes Score* is used only as a monitoring measure of the system. However, it is possible to make a feedback mechanism so that system changes its behaviour in accordance to its current *Silhouettes Score*. For example, instead of using fixed parameters for clustering algorithm, the system will able to use different settings in order to increase its *Silhouettes Score* in each time window. As show in Figure 4.5, discovered bursty events and their relations will form a connected graph. We believe that further analysis of the graph can reveal insights that may be useful. For example, sequence of related events may possibly represent two or more related non-bursty events and they can only get discovered by clustering the bursty events themselves. Figure 5.1 shows this idea, we performed Chameleon Clustering algorithm on one of the connected component of the event graph. Even though all the bursty nodes are well connected in this graph, the graph clustering algorithm divided them into different groups based on their edges. Further analysis may reveal facts and helps operators to draw better conclusions.

Some nodes in the event graph, as shown in Figure 4.5, have higher in-degree or out-degree edges. Such nodes are playing role similar to *hub* as they connect many bursty events in the graph. Studying their significance may reveal interesting applications.

References

[1] J. Allan, H. Jin, M. Rajman, C. Wayne, D. Gildea, V. Lavrenko, R. Hoberman, and D. Caputo, Proceedings of summer workshop at CLSP,final report, 1999.

[2] James Allan, *Introduction to topic detection and tracking*, Topic Detection and Tracking (James Allan, ed.), The Information Retrieval Series, no. 12, Springer US, 2002.

[3] She Antenucci, Gregory Handy, Akshay Modi, and Miller Tinkerness, *Classification of tweets via clustering of hashtags*, (2011).

[4] the Atefeh and Wael Khreich, *A survey of techniques for event detection in twitter*, 2013.

[5] Thorsten Brants, Francine Chen, and Ayman Farahat, *A system for new event detection*, Proceedings of the 26th annual international ACM SIGIR conference on Research and development in information retrieval, ACM, 2003, pp. 330–337.

[6] Dave Engel, Paul Whitney, and Nick Cramer, *Events and trends in text streams*, Text Mining, John Wiley & Sons, Ltd, 2010, pp. 165–182.

[7] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu, *A density-based algorithm for discovering clusters in large spatial databases with noise.*, Kdd, vol. 96, 1996, pp. 226–231.

[8] Martin Ester, Hans-Peter Kriegel, Jrg Sander, and Xiaowei Xu, *A density-based algorithm for discovering clusters in large spatial databases with noise*, Proceedings of the Second International Conference on Knowledge Discovery and Data Mining, AAAI Press, 1996, pp. 226–231.

[9] I. K. Fodor, *A survey of dimension reduction techniques*, vol. 9, 2002, pp. 1–18.

[10] M. Franz, A. Ittycheriah, and J. S. McCarley, *First story detection: Combining similarity and novelty-based approaches*, Slides at the TDT-2001 meeting, 2001.

[11] Gabriel Pui Cheong Fung, Jeffrey Xu Yu, Philip S. Yu, and Hongjun Lu, *Parameter free bursty events detection in text streams*, Proceedings of the 31st international conference on Very large data bases, 2005, pp. 181–192.

[12] D.A. Grossman and O. Frieder, *Information retrieval: Algorithms and heuristics*, Kluwer international series on information retrieval, Springer, 2004.

[13] J. A. Hartigan and M. A. Wong, *Algorithm as 136: A k-means clustering algorithm*, **28** (1979), no. 1, 100–108.

[14] H. Hotelling, *Analysis of a complex of statistical variables into principal components*, no. 24, 1933, pp. 417–441.

[15] H. Jin, R. Schwarts, S. Sista, and F. Walls, *Topic tracking for radio, tv broadcast, and newswire*, Proceedings of the DARPA Broadcast News Workshop, Morgan Kauffman Publishers, 1999, pp. 199–204.

[16] JON KLEINBERG, *Bursty and hierarchical structure in streams*, vol. 7, Kluwer Academic Publishers, 2003, pp. 373–397.

[17] Hans-Peter Kriegel, Peer Krger, and Arthur Zimek, *Clustering high-dimensional data: A survey on subspace clustering, pattern-based clustering, and correlation clustering*, vol. 3, 2009, p. 1.

[18] Zhiwei Li, Bin Wang, Mingjing Li, and Wei-Ying Ma, *A probabilistic model for retrospective news event detection*, Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval, ACM, 2005, pp. 106–113.

[19] Junshui Ma and Simon Perkins, *Online novelty detection on temporal sequences*, Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining, 2003, pp. 613–618.

[20] Das Martins, *A survey on automatic text summarization*, (2007), 192–195.

[21] Qiaozhu Mei and Zhai, *A mixture model for contextual text mining*, Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining, ACM, 2006, pp. 649–655.

[22] Qiaozhu Mei and ChengXiang Zhai, *Discovering evolutionary theme patterns from text: an exploration of temporal text mining*, Proceedings of the eleventh

ACM SIGKDD international conference on Knowledge discovery in data mining, 2005, pp. 198–207.

[23] P. Mulbregt, I. Carp, L. Gillick, S. Lowe, and J. Yamron, Proceedings of the DARPA Broadcast News Workshop, Morgan Kauffman Publishers, 1999, pp. 77–80.

[24] Lance Parsons, Ehtesham Haque, and Huan Liu, *Subspace clustering for high dimensional data: a review*, vol. 6, 2004, pp. 90–105.

[25] K. Pearson, *On lines and planes of closest fit to systems of points in space*, no. 2, 1901, pp. 559–572.

[26] Kanagasabi Rajaraman and Ah-Hwee Tan, *Topic detection, tracking, and trend analysis using self-organizing neural networks*, Advances in Knowledge Discovery and Data Mining, Springer, 2001, pp. 102–107.

[27] Stuart Rose, Dave Engel, Nick Cramer, and Wendy Cowley, *Automatic keyword extraction from individual documents*, Text Mining, Applications and Theory, Wisly, 2010, pp. 4 – 20.

[28] Peter J. Rousseeuw, *Silhouettes: A graphical aid to the interpretation and validation of cluster analysis*, vol. 20, 1987, pp. 53–65.

[29] James G. Shanahan, Yan Qu, and Janyce Wiebe, *Computing attitude and affect in text: theory and applications*, **20**.

[30] T. Snowsill, F. Nicart, M. Stefani, and T. De Bie, *Finding surprising patterns in textual data streams*, IEEE, 2010, pp. 405 – 410.

[31] B. Tang, M. Shepherd, E. Milios, and M. I. Heywood, *Comparing and combining dimension reduction techniques for efficient text clustering*, International Workshop on Feature Selection for Data Mining, 2005.

[32] Wenyin Tang and Flora S. Tsai, *Adaptive threshold setting for novelty mining*, Text Mining (Michael W. Berry and Jacob Kogan, eds.), John Wiley & Sons, Ltd, 2010, pp. 129–148.

[33] Jordi Turmo, Alicia Ageno, and Neus Català, *Adaptive information extraction*, vol. 38, 2006.

[34] Xuanhui Wang, ChengXiang Zhai, Xiao Hu, and Richard Sproat, *Mining correlated bursty topic patterns from coordinated text streams*, Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining, ACM, 2007, pp. 784–793.

[35] Jianshu Weng and Bu-Sung Lee, *Event detection in twitter.*, ICWSM, 2011.

[36] Rui Xu and D. Wunsch, *Neural networks ieee transactions on*, **16** (2005), no. 3, 100–108.

[37] Y. Yang, J. Zhang, J. Carbonell, and C. Jin, *Topic-conditioned novelty detection*, In Proceedings of the International Conference on Knowledge Discovery and Data Mining, 2002.

[38] Yiming Yang and Xin Liu, *A re-examination of text categorization methods*, Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval, 1999, pp. 42–49.

[39] Y. Zhang, J. Callan, and T. Minka, *Novelty and redundancy detection in adaptive filtering*, Proceedings of SIGIR-02, Tampere, Finland, 2002, pp. 81–88.

Vita

SEYED POORIA MADANI KOCHAK

Degrees:

Bachelor of Science (Honours) Computer Science

University of Prince Edward Island, Charlottetown, PEI

2007-2012

Publications:

- Madani, Pooria. Ghorbani, A. A., *Bursty Event Discovery Framework from Online News Outlets*, Canadian AI 2015.
- Aliabadi, Abtin Zohrabi. Madani, Pooria, et al. *Classifying Organizational Roles Using Email Social Networks.* Advances in Artificial Intelligence. Springer Berlin Heidelberg, 2013. 301-307.

Conference Presentations:

- Madani, Pooria. *Information Discovery from Twitter using Hashtags*, University of New Brunswick - Research Expo, 2013.