Text Clustering Using Bursty Information

P. Zagorisios

Master Thesis

—♦—

Ioannina, February, 2015

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ Η/Υ & ΠΛΗΡΟΦΟΡΙΚΗΣ
ΠΑΝΕΠΙΣΤΗΜΙΟ ΙΩΑΝΝΙΝΩΝ

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
UNIVERSITY OF IOANNINA

Text Clustering using bursty information

Η ΜΕΤΑΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ ΕΞΕΙΔΙΚΕΥΣΗΣ

υποβάλλεται στην

ορισθείσα από την Γενική Συνέλευση Ειδικής Σύνθεσης

του Τμήματος Πληροφορικής Εξεταστική Επιτροπή

από τον

Ζαγορίσιο Παναγιώτη

ως μέρος των Υποχρεώσεων για τη λήψη του

ΜΕΤΑΠΤΥΧΙΑΚΟΥ ΔΙΠΛΩΜΑΤΟΣ ΣΤΗΝ ΠΛΗΡΟΦΟΡΙΚΗ

ΜΕ ΕΞΕΙΔΙΚΕΥΣΗ

ΣΤΙΣ ΤΕΧΝΟΛΟΓΙΕΣ ΚΑΙ ΕΦΑΡΜΟΓΕΣ

Φεβρουάριος 2015

ΕΥΧΑΡΙΣΤΙΕΣ

Αρχικά θα ήθελα να ευχαριστήσω θερμά τον επιβλέποντα καθηγητή μου κ. Αριστείδη Λύκα για τις πολύτιμες συμβουλές του, την υπομονή και εμπιστοσύνη που μου έδειξε για την εκπόνηση της μεταπτυχιακής μου εργασίας. Επιπλέον, γιατί ειναι αυτός που με "μύησε" πρώτος στον μαγικό κόσμο των δεδομένων.

Ένα μεγάλο ευχαριστώ στον μεταδιδακτορικό ερευνητή Αργύρη Καλογεράτο για την καίρια βοήθειά του, την υπομονή και την παρότρυνσή του όλα αυτά τα χρόνια, αλλά και για τον χρόνο που μου αφιέρωσε. Ήταν τιμή μου να συνεργαστώ μαζί του.

Επιπλέον, θα ήθελα να ευχαριστήσω τον καθηγητή κ. Κωνσταντίνο Κοτρόπουλο που μας παρείχε πρόσβαση σε δεδομένα αλλά και τον κ. Andreas Kaltenbrunner που μου έδωσε την ευκαιρία να συνεργαστώ μαζί του στην Βαρκελώνη στο Fundacio Barcelona media.

Τέλος τους γονείς και την αδερφή μου για την στήριξη και την υπομονή τους όλο αυτό το διάστημα.

TABLE OF CONTENTS

1 Introduction **1**

2 Related Work **3**

2.1 Topic Detection and Tracking (TDT) 3

2.1.1 Online vs Offline TDT 6

2.1.2 Text Stream Visualization 6

2.2 Bursty Information 8

2.2.1 Event Burst 8

2.2.2 Burst detection 9

2.2.3 Kleinberg model 10

2.3 Vector Space Model 13

2.3.1 GVSM 14

2.4 Event Modeling 15

2.4.1 Document based techniques 16

2.4.2 Bursty Feature representation 18

2.5 Spherical K-means 20

3 Correlated Bursty Terms **22**

3.1 Terminologies 22

3.2 Term similarity matrices 23

3.3 Correlated Bursty Term Clustering 25

3.3.1 Bursty Term Graph 26

3.3.2 Cluster representatives 27

3.3.3 Merging step 29

4 Experiments **30**

4.1 Data-sets 30

4.2 Text Stream Generator 33

4.3 Experimental Protocol 35

4.3.1 Evaluation Metrics 37

4.4 Experimental Results 38

5.3 Conclusions and Future Work ... 48
5.3.1 Conclusions ... 48
5.3.2 Future Work ... 49

Bibliography ... 50

List of Figures

2.1 TDT Definitions 4
2.2 The Memetracker tool 6
2.3 The ThemeRiver tool 7
2.4 The BlogPulse tool 8
2.5 Histogram of a topic from TDT5 data set 8
2.6 Documents similarity vs event activeness 9
2.7 Kleinberg's infinite automaton 10
2.8 The hierarchical structure of burst 11
2.9 Kleinberg's 2-state automaton 11
2.10 Single pass document clustering 17
2.11 Energy-based Single Pass clustering 18

3.1 An example of BI 24

4.1 Text stream distribution of D1 and D2. 46
4.2 Text stream distribution of D3. 47
4.3 Text Stream distribution of D5. 47

List of Tables

2.1 Summary of representation models 20

4.1 Characteristics of text document collections 31
4.2 Selected topics for D1,D2 and D3 32
4.3 D5 - subset of GoogleNews 32
4.4 D4 - subset of TDT5 33
4.5 Parameters of the Text Stream Generator 35
4.6 Stream Statistics 36
4.7 Term-Similarity matrices results on the D5 using spherical K-means 38
4.8 Term-Similarity matrices results on the D4 using spherical K-means 39
4.9 Term-Similarity matrices results on the D3 using spherical K-means 40
4.10 Term-Similarity matrices results on the D1 using spherical K-means 41
4.11 Term-Similarity matrices results on the D2 using spherical K-means 43
4.12 Random Initializationation vs CBTC - results on the D5, D4 using spherical K-means 45
4.13 Random Initializationation vs CBTC - results on the D3 using spherical K-means 45
4.14 Random Initializationation vs CBTC - results on the D1,D2 using spherical K-means 46

ABSTRACT

Panagiotis Zagorisios.
MSc, Computer Science Department, University of Ioannina, Greece.
Text Stream Clustering using bursty information.
Supervisor: Aristidis Likas

Document clustering in text streams is a text mining problem with increasing interest in recent years due to its relation to the problems of topic detection and tracking. The existence of temporal information in the form of document timestamps provides the opportunity to modify and improve the typical approaches for document representation and clustering. A way to exploit temporal information is through the detection and exploitation of bursty terms in text streams, ie terms that appear in many documents during the same time window.

At first, a description of methods that have been developed for the detection of bursty terms is presented. Next several document representations are presented that integrate the bursty information in the typical vector space model, and novel forms of term similarity matrices are proposed that take into account the correlation between bursty terms. Moreover, we propose an alternative approach that first partitions the bursty features into groups and then uses this information for the clustering of the text stream collections. Finally, experimental results on benchmark collections are provided followed by empirical conclusions on the performance of the compared approaches.

Εκτεταμενη Περιληψη στα Ελληνικα

Παναγιώτης Ζαγορίσιος.
MSc, Τμήμα Πληροφορικής, Πανεπιστήμιο Ιωαννίνων.
Ομαδοποίηση ροών κειμένων κάνοντας χρήση πληροφορίας αιχμής.
Επιβλέπων καθηγητής: Αριστείδης Λύκας

Το πρόβλημα της ομαδοποίησης σε ροές κειμένων παρουσιάζει αυξανόμενο ενδιαφέρον, κυρίως εξαιτίας των εφαρμογών του σε συστήματα ανίχνευσης και παρακολούθησης γεγονότων (TDT) σε ειδησεογραφικά κείμενα. Η ύπαρξη χρονικής πληροφορίας σχετικά με τη χρονική στιγμή εμφάνισης ενός κειμένου, δημιούργησε την ανάγκη να τροποποιηθούν οι τυπικές προσεγγίσεις για την ομαδοποίηση κειμένων, ώστε να εκμεταλλεύονται αποδοτικά την πληροφορία αυτή. Μια προσέγγιση στην εκμετάλλευση της χρονικής πληροφορίας βασίζεται στον εντοπισμό λέξεων που εμφανίζονται ταυτόχρονα σε πολλά κείμενα του ίδιου χρονικού παραθύρου τους οποίους ονομάζουμε "όρους αιχμής" (bursty terms).

Στην εργασία αυτή παρουσιάζονται μεθοδολογίες που έχουν προταθεί για τον εντοπισμό όρων αιχμής και των αντίστοιχων χρονικών διαστημάτων αιχμής για κάθε όρο.Στη συνέχεια παρατίθενται κάποιες μέθοδοι που έχουν προταθεί για την ενσωμάτωση της πληροφορίας αιχμής στην αναπαράσταση των κειμένων, προσπαθώντας κυρίως να βελτιώσουν την κλασική διανυσματική αναπαράσταση ενός κειμένου (bag of words). Παρόλα αυτά σε καμία από αυτές τις μεθόδους δεν λαμβάνεται υπόψη η συσχέτιση των όρων.

Στη συνέχεια της εργασίας μελετάται κατά πόσο η ομαδοποίηση σε ροές κειμένου δύναται να βελτιωθεί με την εξόρυξη πληροφορίας που αφορά την συσχέτιση όρων αιχμής. Γι αυτό τον λόγο, προτείνεται η χρήση πινάκων ομοιότητας όρων (λέξεων) που κατασκευάζονται από όρους αιχμής και την πληροφορία συσχέτισής τους.

Επιπλέον προτείνεται μια άλλη μεθοδολογία που βασίζεται καταρχήν στην ομαδοποίηση των όρων αιχμής ώστε να σχηματίσουν εννοιολογικές ομάδες (topics), και στη συνέχεια στην εκμετάλλευση των εννοιολογικών ομάδων για την ομαδοποίηση των κειμένων της συλλογής. Οι μεθοδολογίες που παρουσιάζονται συγκρίνονται πειραματικά χρησιμοποιώντας γνωστές συλλογές κειμένων και παρατίθενται εμπειρικά συμπεράσματα σχετικά με τις δυνατότητες της κάθε προσέγγισης.

CHAPTER 1

INTRODUCTION

Thanks to the development of the second version of the world wide web (Web 2.0), there is a continuous growth in the available digital content. The vast amount of this information is described by the term "big data", which came to the fore the last years. Moreover, the founder of the Web, Tim Berners-Lee, presented in public some interesting results, when the available data gets linked up[1]. Tools for clustering are necessary for organizing and analyzing any available information. Clustering is one of the 6 common tasks in the data mining field [15], the other 5 are classification, regression, anomaly detection, summarization and association rule mining. Each clustering process aims to discover groups of "similar objects" and is a widely studied problem in the text domains.

Nowadays, text documents could be produced by many sources. From user platforms like twitter and facebook to newswire sites and blogs. Conventional clustering techniques have difficulty in handling the large amount of text data (text streams) that produced over time, as new challenges are faced. Topic detection and tracking (TDT) is a related problem to that of text stream clustering. The goal of TDT is the identification of clusters of documents from a non-stationary text collection, where each cluster contains documents that discuss the same real life event.

Moreover, text representation is an important issue because it affects the performance of a text mining algorithm. The most widely used representation for the task of text clustering is the Vector space model-bag of words (VSM). In [21] the bursty feature representation is proposed, that is appropriate for text streams as it captures the temporal dimension of a document. The role of bursty features, have been investigated and in other works proposing different representations [23] [58] that extend the classical vector space model. However, none of the above works considers the relation between bursty features. Despite its simplicity, the major limitations of VSM is that terms are statistically independent. The generalized vector space model (GVSM) is a generalization of the vector

[1] `http://www.ted.com/talks/tim_berners_lee_the_year_open_data_went_worldwide`

space model that estimates the correlation between terms [46].

The research question that this thesis addresses is if the clustering procedure of text streams could be further enhanced by exploiting bursty feature correlations. We extend the GVSM term similarity matrix using the bursty features discovered by the Kleinberg's 2 states automaton [29]. Furthermore, we propose several forms of matrices, based on the correlation of bursty terms. Finally, we introduce the Correlated Bursty Term Clustering (CBTC) algorithm for the initialization of the k-means, appropriate for text stream data collections.

The rest of this thesis is organized as follows. Chapter 2 describes the related work. In chapter 3, our methodology is documented. The experimental clustering results in benchmark datasets are reported in chapter 4. Finally, in chapter 5, we present our conclusions as well as future research directions.

CHAPTER 2

RELATED WORK

2.1 Topic Detection and Tracking

2.2 Bursty Information

2.3 Vector Space Model

2.4 Event Modeling

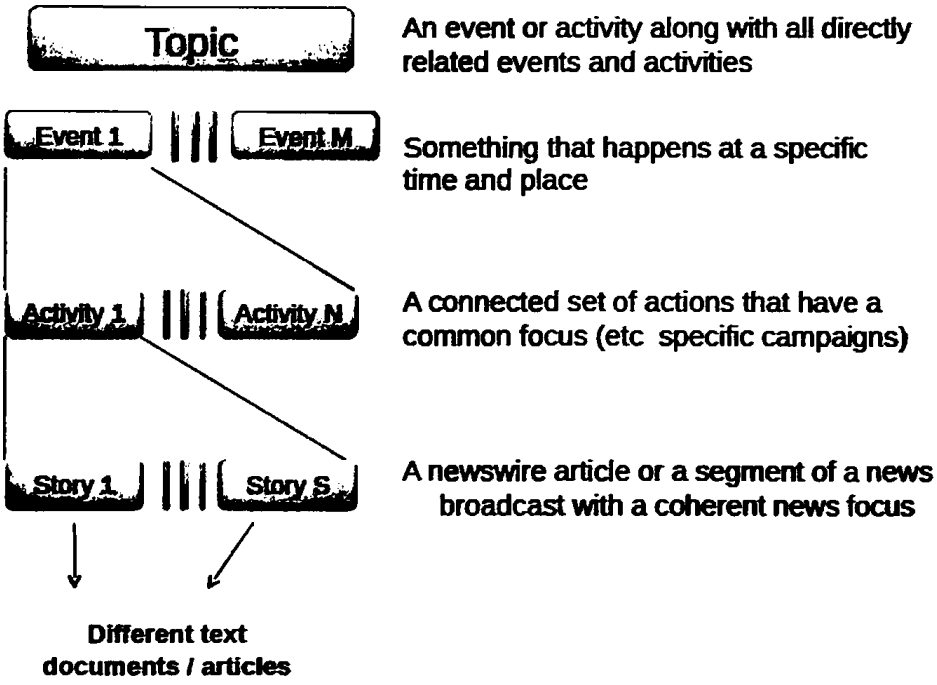
2.5 Spherical K-means

2.1 Topic Detection and Tracking (TDT)

The TDT program started in 1997 as a pilot study, conducted initially by a small group of researchers. The research was pursued under the DARPA Translingual Information Detection, Extraction and Summarization (TIDES) program and the goal was to develop technologies for organizing the text news from a variety of broadcast news media in order to detect and track the appearance of news topics. A detailed overview of the results of the pilot study could be found in [1]. A fundamental concept in TDT is the notion of an "event" and a "topic". During the pilot study "event" and "topic" meant the same thing, an incident that happens at a certain place and time. Later, in the second year, the definition of "topic" altered and broadened to include not only the triggering event but also other events and activities that are directly related to it. The notion of a "topic" differs form the notion of a general category like sports, politics etc. For example the "final game of the FIFA world cup 2014" and the "final game of UEFA champions league 2014" although belonging to the same category, the one of sports, they refer to totally different real life events. As a result, they are two distinct topics.

We could imagine a TDT system operating as follows. Given a text stream, the framework should discover the documents-stories that refer to a topic which have not been detected before. Furthermore, each topic referring to a real life event, should be tracked in order to discover more stories that mention it. Other important notions of the TDT project can be found in the figure 2.1.

Figure 2.1: TDT Definitions



According to the TDT community , five different research applications are defined [1] :

1. **Story Segmentation (SS)** - detect story boundaries
   The automatically transcribed speech data, coming from news agencies, need to be segmented into stories. In other words, the stream information should be spotted in discrete stories.

2. **Topic Tracking (TT)** - Discover all stories that discuss a target topic.
   Given an sample of stories that discuss a specific topic, TT framework should find all the subsequent stories, within the remaining corpus, that refer to the same topic.

3. **Topic Detection (TD)** - Identify clusters of stories that discuss the same topic.
   For the TD task no prior knowledge is provided as happens for the topic tracking

[1] http://www.itl.nist.gov/iad/mig/tests/tdt/

task. It is an unsupervised process. The TD system should detect clusters of stories that discuss the same topic.

4. **First Story Detection (FSD)** - If a story is the first story of a new, unknown, topic.
   FSD, also known as new event detection, detects the first story that refers to an event. An FSD system should output either YES or NO to the question: "does this story discuss a new topic?". Like topic detection, first story detection is an unsupervised task.

5. **Link Detection (LD)** - Detect whether or not two stories are linked.
   A LD system should output a decision score determining whether a pair of stories discuss the same topic. In that case the stories are linked by the same topic.

The above tasks are not necessarily independent, because for the successful implementation of one of them, probably has another one as a prerequisite. For example task 3 should be solved before we perform any of the rest.

The five tasks were shaped after the first year of the pilot study. The initial tasks were segmentation, tracking and detection [1]. Furthermore, the topic detection task, was divided into the online and the retrospective process. The online method is known as new event detection (FSD) and the offline as retrospective event detection (RED). The objective of RED is the identification of all the events in a corpus of stories, by grouping the stories into clusters where each cluster represents an event.

At this point, we should notice that because of long and intensive research in the TDT field, occasionally other problems have been defined that are closely related. Fung et all [18] formalize the problem of "bursty event detection", as the discovery of a set of bursty words, that are able to descibe an event, while Platakis et all in [40] study the problem of discvovering hot topics in blogs, from the same perspective. From now and on, we will be using the notion of "topic" and "event" interchangeably.

Regarding the research so far, we could claim that there are two trends in the literature on how a topic would be represented. The document-based, originated from the TDT community, and the feature-based. In the document-based approach, the organization of documents into clusters defines the different set of topics. Therefore, the content of an event is described by the documents of each cluster or descriptive queries. However, it has been observed that events from news sources, appear frequently in bursts. In the latter approach, the feature-based, a topic is signaled by the burst of the frequencies of terms associated with it. The organization of the terms into groups form the detected events. The question which arises is how to identify the features whose frequency is significantly increased, and how to group them in order to shape the final events.

2.1.1 Online vs Offline TDT

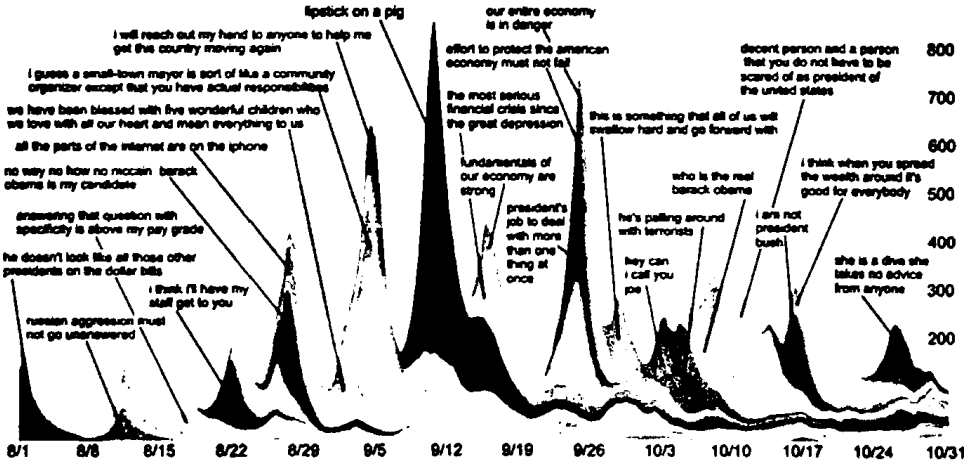
According to [22] topic detection models could be classified into probabilistic and non-probabilistic. In the first category a "topic" is considered as a distribution over either documents or words, while in the second, the documents are clustered directly. Our research is related to the non-probabilistic case. In both cases there are two approaches to detect and track topics.

The off-line (retrospective) approach assumes that the entire text stream is available for analysis. This, however, requires memory resources for storing and processing the available information. So decisions are taken periodically, ie non-real time. In contrast, the on-line approach considers that documents arrive in the system according to a chronological order. Thus, the required memory should be sufficient only for the processing of the current information flow. As a result the information is scanned only once, which requires the system to take decisions in a specified time interval or in strict-real time(immediate mode).

In [45] are described in detail the state of art approaches to immediate mode of on-line new event detection. In [35] an online new event detection framework is illustrated that deals with the above mentioned issues and could be used in practice. Finally, [8] describes the first Turkish news portal supporting event detection and tracking procedures.

2.1.2 Text Stream Visualization

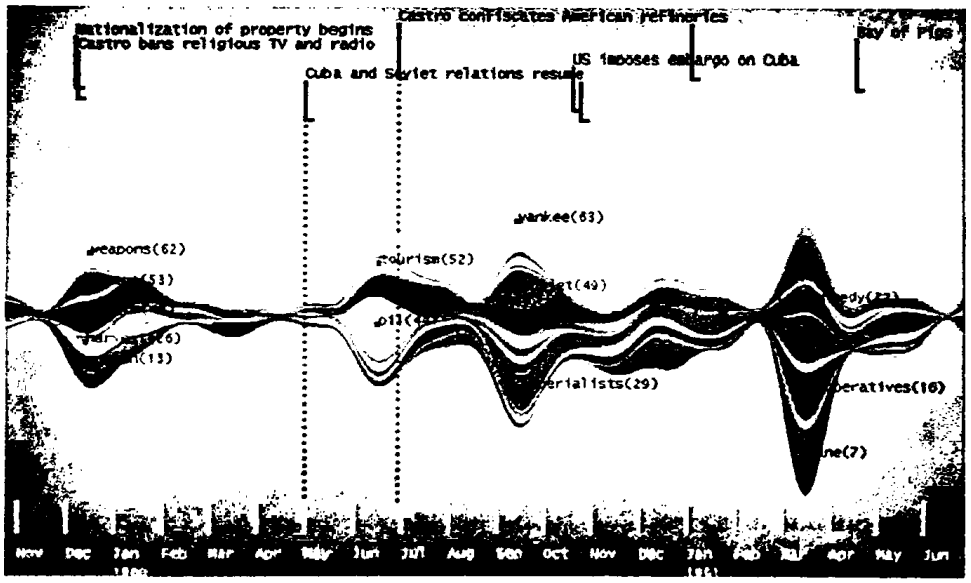
Figure 2.2: The Memetracker tool



The methods that have been developed for the task of topic detection and tracking, as well as from the broader related research areas, could be used for the visualization of text streams. Many tools and frameworks have been proposed. The Event River visualizes each event, by extracting the name of the involved people and the places related to it [34].

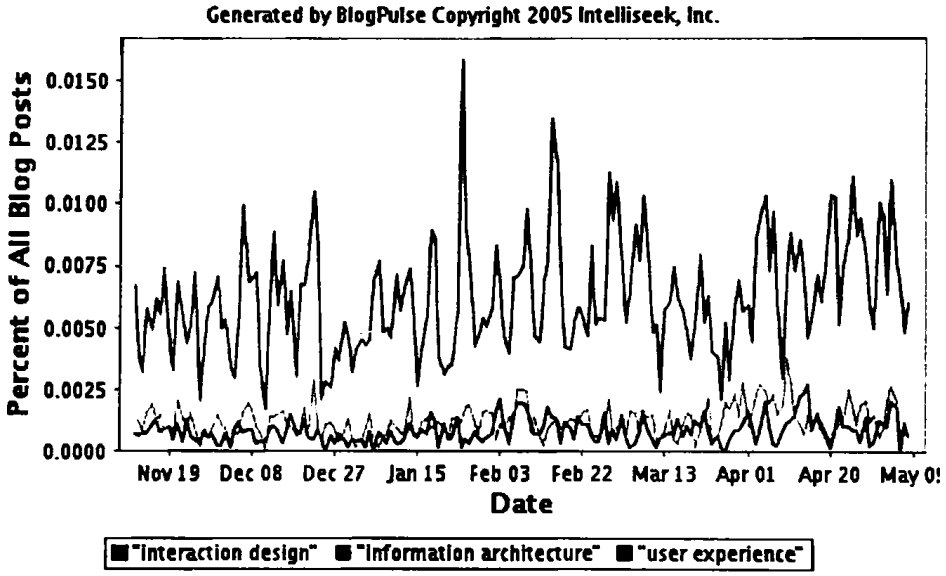
Another well known framework is the Meme-tracker. The novelty of this framework is that it displays short, distinctive phrases that appear in many documents. Furthermore someone could observe how these "memes" (phrases) evolve in time [33]. ThemeRiver is a system that visualizes thematic variations of a collection of documents, over time [19]. Blogpulse [7] was a search engine (no longer available) and analytical system for detecting trends in blogs, with visualization features . Another system, navigating the blogosphere, from a spatiotemporal perspective, is blogscope, which takes advantage of bursty keywords and keywords correlations [4].

Figure 2.3: The ThemeRiver tool



Another tool for analyzing unstructured text streams, by presenting trending keywords, is produced by Microsoft Research Labs and is called Narratives [16]. Finally, extended research for the topic detection and tracking field, has also been conducted, not only on data derived from traditional newswire sources like news agencies and weblogs, but also on the Twitter micro-blog platform. As a result, several tools have been developed. For example TopicSketch [48] is a framework for real-time detection of bursty topics, providing a descriptive "snapshot" of the current stream. On the other hand, CLEar [47], is a system that supports bursty topic detection, popularity prediction, event summarization, contextualization and visualization. A further extension of the above systems, that worths mentioning, is Truthy which is produced and maintained by Indiana University. It's, actually, a framework for real-time analysis of memes diffusion, in social media by mining massive streams of public micro blogging events [42].

Figure 2.4: The BlogPulse tool

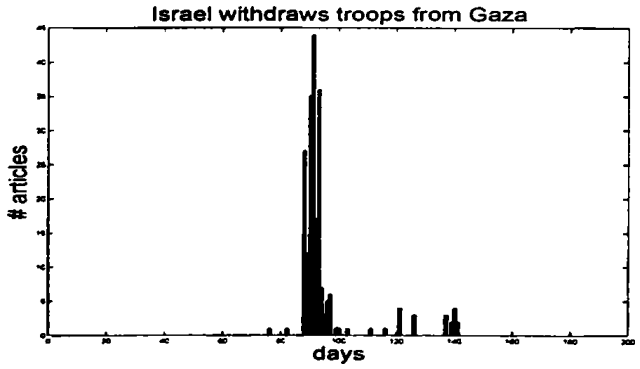


2.2 Bursty Information

2.2.1 Event Burst

The burst of an event is signaled by the appearance of a large number of relevant documents in a relative small period of time. This depends on the number of documents and their spread in time. Thus events are characterized by a life cycle according to their popularity during time. There are events with long, short or periodic lifespan. For example events about football matches are periodic (every weekend), while an event related to a war has a longer lifespan than an event about a car accident.

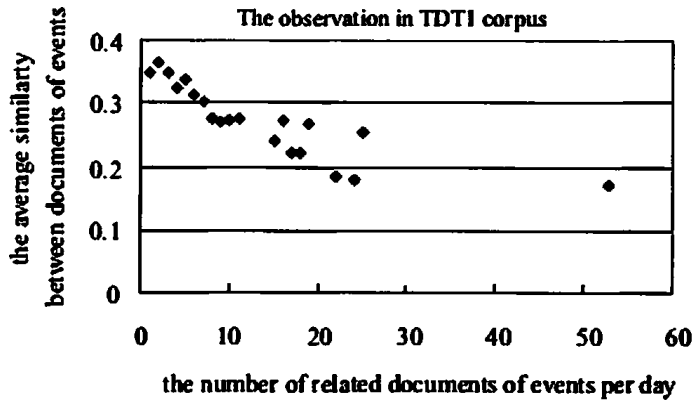
Figure 2.5: Histogram of a topic from TDT5 data set



One further problem of topic detection and tracking is the context shifting. During the life-cycle of an event, the context of documents related to it tends to alter. This is because when an event is active, more and more documents are published that present diverse

aspects of it and this results in reduction of the average similarity among the documents of the same event. As Kleinberg observes in [29], "the burst-and-diverse" phenomenon of events corresponds to the dynamic behavior of a document stream. According to figure 2.6 (retrieved from [9]) that concerns the TDT1 copus, when an event is active the number of related documents (x-axis) is increased, while the average similarity between them is reduced (y-axis).

Figure 2.6: Documents similarity vs event activeness



2.2.2 Burst detection

The rapid increase in the frequency of a term in a short period is called burst. Thus, a word is bursty, when its frequency is encountered at an unusual high rate. The detection of bursty events is known as burst detection procedure and it has a variety of applications. From monitoring the network traffic or the price of a stock market to astronomical observations and web topic Mining. Several methodologies for burst detection have been proposed [6, 56, 59].

In the field of data/text mining, occasionally, several burst detection algorithms have been developed. [29] detects bursty words in email collection, using an infinite-state automaton and computes the optimal state sequence using a statistical procedure. A variation of Kleinberg's method for BBSs (Bulletin Board Systems) and blogs text collection, where the distribution of documents is not uniform, is described in [17]. In [52] a parameter free burst detection algorithm is presented, based on sophisticated data structures, which detects bursts in multiple windows sizes simultaneously.

Fung et all [18] suggests a probabilistic framework, based on the binomial distribution, to identify bursty words. Furthermore they apply spectral analysis in order to categorize words in four categories to discover important and less-reported events. Lappas et all in

[32] uses concepts from discrepancy theory, to model the burstiness of a word, developing a parameter-free approach.

In addition, [11] presents the OMRBD algorithm, that detects bursty words by maintaning multiple sliding windows of different resolution. Moreover, in [51] the difference between a bursty word and a buzzword is highlighted, proposing a buzzword detection method. As Yi states "*Buzzwords are terms of high momentum for a relatively short period of time. Note that not all bursty events by the Kleinberg's model can be considered as buzz because the model doesn't take into account the relative duration and the mass of bursts*".

Finally, Vlachos et all in [44] deals with the identification of bursts from the query logs of Microsoft's search engine. While in [27] it is examined the use of bursts of edits in the discussion pages of Wikipedia articles to explore the process of how a topic is created.

2.2.3 Kleinberg model

A popular method for burst detection is proposed in [29], where author tries to investigate the role of time in his personal e-mail by modeling the text stream using an infinite-state automaton (figure 2.7).

Figure 2.7: Kleinberg's infinite automaton

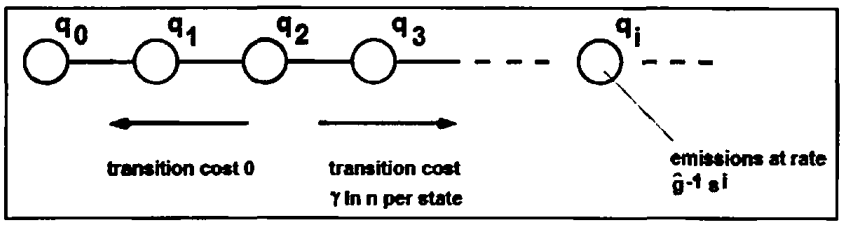


Figure retrieved from [29]

In this automaton, bursts appear as state transitions, corresponding to points in time around which the frequency of the examined word changes significantly. Moreover, a hierarchical tree structure is created by the state transition-burstiness connection (figure 2.8).

The most basic bursty model is the automaton with two states. It is ideal to describe the timeline of a term. In this thesis, we used the batch mode of the 2-state automaton, where documents arrive into consecutive batches. Suppose that there are T batches, where the sequence $(d_1, ..., d_T)$ expresses the total number of documents in each time slice, while $(r_1, ..., r_T)$ is the number of documents that contain the examined word. The algorithm

is a kind of a Hidden Markov model. The automaton has 2 states, one with low emission rate $p_0 = |Rd|/T$ and another with higher rate $p_1 = s \cdot p_0$, $s \succ 1$ (resolution). T is the whole time range and $\sum_{i=1}^{T} r_i = |\ Rd\ |$.

Figure 2.8: The hierarchical structure of burst

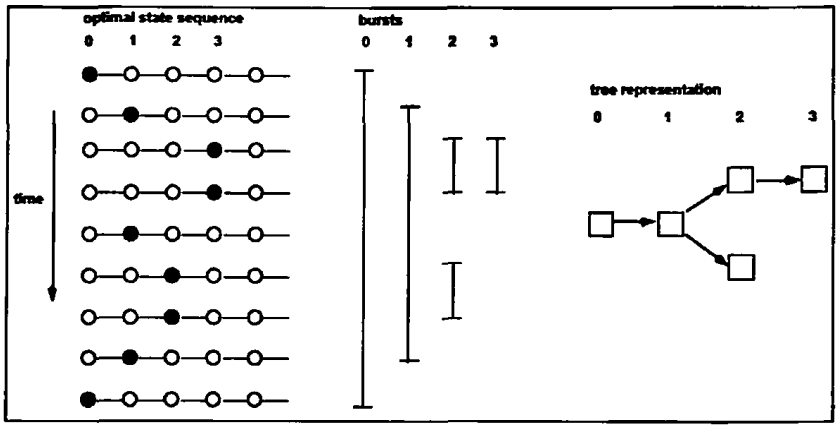
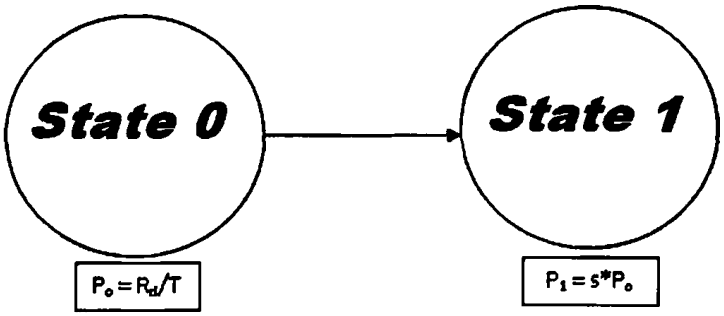


Figure retrieved from [29]

Figure 2.9: Kleinberg's 2-state automaton



In the proposed model each (r_i, d_i) is considered to be an output symbol, that is produced probabilistically according to the internal state of the HMM. When such tuple sequence is given, the goal is to find an optimal state sequence $q = (q_1, ...q_T)$, where each q_i minimizes the cost that is calculated by the following equation:

$$\sigma(i, r_t, d_t) = -ln\left[\binom{d_t}{r_t} p_i^{r_t}(1-p_i)^{d_t - r_t}\right] \tag{2.1}$$

The state transition sequence s that minimizes the above cost function is derived using a dynamic programming algorithm, called Viterbi method. For a given input sequence $< r, d >= (< r_1, d_1 >, \ldots, < r_T, d_T >)$, the Viterbi method identifies the most likely state sequence (viterbi path) of a hidden Markov model as follows.

1. t=0, $C_0(t) = 0$ $C_1(t) = \infty$
2. $t = t + 1$
3. Compute cost $C_j(t)$ for j=0,1
   (a) $C_j(t) = min(C_j(t-1) + r(q,j)) - \sigma(j, r_t, d_t)$
   (b) r(q,j) is the state transition from the previous state q and is defined as $r(q,j) = \gamma * logT$ for $q < j$ and $r(q,j) = 0$ for $q \geq j$
4. Repeat steps 2 and 3 for all batches of documents
5. Select the state sequence with the minimum cost

Consecutive appearances of state 1 (bursty state) are considered as bursty moments of the word. Thus, the length of a burst is defined from the appearance of the first (t1) until the last bursty moment (t2), while the weight of the burst is calculated from equation: .

$$w = \sum_{t=t_1}^{t} {}_2(\sigma(0, r_t, d_t) - \sigma(1, r_t, d_t)) \tag{2.2}$$

The weight, which is non-negative, is equal to the improvement in cost incurred by using state p_1 over the interval [t1,t2] rather than state p_0.

The algorithm has two parameters. The resolution (s) and the conversion cost (γ). The first one controls the difference between the burst (high emission rate) and the normal state (low emission rate). As Kleinberg mentions, small values of s often lead to long bursts, while higher values increase the strictness of the algorithm criterion for how rapid an increase of activity should in order to be considered as burst.

The second one (conversion cost) defines the cost of the automaton when changing state. Higher (lower) values mean that the burst must be sustained over longer (shorter) period of time in order to be detected by the algorithm. The default value of this parameter is set to 1 which indicates that the cost is proportional to the increase in state number.

Unfortunately Kleinberg didn't propose any way of tuning these parameters that were manually set. However, two algorithms are proposed in a recent study [13] for the estimation of the above parameters. The disadvantage of these methods is that the estimation

of the parameters is investigated at the word and not at the corpus level. Thus, it is time consuming to find the best parameters for each word in the vocabulary, especially when the vocabulary length is of order 10^3 in magnitude. Moreover the Kleinberg's burst detection algorithm is not appropriate for online burst detection. Finally, it is a computational expensive method as for its implementation is used a dynamic programming algorithm.

2.3 Vector Space Model

The most widely-used text representation in the field of text mining is the Vector Space Model (or term vector model). In VSM, a document is represented by a vector of weights corresponding to text "features". The weight of each feature quantifies its importance to describe the document content. Thus, not all text features are useful in representing a documents. As features, according to the common approach of Bag of Words (BOW), distinct words are considered.

A document d, is represented as a vector of $\mathbf{V}$ features. In equation 2.3 T denotes the document transpose and d_{ij} the weight of *j-th* feature (f_j) inside document d_i.

$$d = [d_{i1}, \ldots, d_{i|V|}]^T, d \in R^V \tag{2.3}$$

The most popular weighting scheme is the normalized tf-idf, which is the product of two statistics. Term frequency and inverse document frequency.

$$tf - idf_{t,d} = tf_{t,d} \times idf_t = tf_{t,d} \times log(\frac{N}{df_t}) \tag{2.4}$$

Although, various ways for calculating the values of both statistics have been proposed, we used the simplest and most common choice, where $tf_{t,d}$, is the raw frequency of term t in document d. The idf_t, is the logarithmically scaled fraction of documents that contain the term t, obtained by dividing the total number of documents (N) by the number of documents containing the term, and then taking the logarithm. It is a measure of how much information the term provides. In other words, whether the term is common or rare across all documents. When the TFIDF representation is used, cosine similarity has been proved to be an effective measure for the task of document clustering. Given two documents vectors, d_i and d_j, cosine similarity computes the cosine of the angle between them.

$$sim_{cos}(d_i, d_j) = \frac{d_i^T d_j}{||d_i||_2 ||d_j||_2} \in [0, 1] \tag{2.5}$$

An extension of VSM-BOW is the bag of phrases (VSM-BOP), where consecutive words are regarded as distinct features (word n-grams). Despite the simplicity of VSM, there are some deficiencies, concerning the assumption of term independence. According

to this hypothesis, the terms are statistically independent. Moreover, in the vector space representation, the order in which the terms appear in the document is lost. In addition, documents are represented in high dimensional and sparse feature space due to the large vocabulary size. Finally the VSM-BOW cannot handle language phenomena such as synonymy and polysemy, because the context a word appear is missing. Thus, due to the semantic sensitivity, documents with similar context but different terms vocabulary are hard to be associated.

2.3.1 GVSM

Because of the difficulties with VSM, many variations have been proposed that map the document vectors to a new feature space. This new space of features is known as concept space and its dimension is equal or less than the initial. A document projection to the concept space can be defined as

$$P^{vsm} \rightarrow d' = Sd \in R^{V'} \qquad V' \leq V \tag{2.6}$$

The matrix S, of dimension $V' \times V$, is called semantic matrix. The cosine similarity between two documents in the concept space can be computed with equation 2.7 where l_i^S is the normalization factor of document d_i.

$$sim_{sem}^{(cos)}(d_i', d_j') = (Sd_i)^T(Sd_j) = (l_i^S Sd_i)^T(l_j^S Sd_j) = l_i^S l_j^S (d_i^T S^T Sd_j) \tag{2.7}$$

As reported by Kalogeratos in [26], the methods that have been proposed so far, either interpret the above equation as a dot product of the document images to the new feature space $R^{V'}$, or as a measure that considers the correlation between features which is expressed by the matrix S^TS.

The Generalised Vector Space model (GVSM), estimates the similarity between documents based on how their terms are related [46]. The image of a document to the concept space is given by:

$$d' = Xd \tag{2.8}$$

In the above equation, d is part of document collection X. More specifically, X is a NxV Document-Term matrix, whose rows and columns are indexed by the documents and vocabulary terms respectively. This, according to equation 2.7, implies that S^TS= X^TX=SIM_{GVSM} is a term similarity matrix. SIM_{GVSM} represents the inner-products of term vectors, introducing term to term correlation by deprecating at the same time the pairwise orgononality assumption of the VSM. It is actually a $V \times V$ term similarity matrix where the *r-th* row has the dot-product similarities between term v_r and the rest terms of the vocabulary.

There are works that purpose different measures to capture the terms correlations [5, 12, 14, 26]. In [5] the Context Vector Model (CVM-VSM) is introduced, where two co-occurrence frequency measures are proposed in order to construct the term similarity matrix. One relies on low level counting the co-occurrence of terms in the same documents and the other tries to discover terms associations at a higher semantic similarity level. Finally, [14] suggests the use of the covariance matrix between terms and the matrix of Pearson's correlation coefficients to estimate the term similarity matrix.

2.4 Event Modeling

As reported before, in literature two approaches are followed in off-line or on-line mode, in order to model an event:

1. Document-Based approaches

   - [2] Each new document that arrives in the system, is compared with all clusters of documents that have been created so far. If its similarity measure with the cluster exceeds a threshold then is assigned to it. Otherwise, this document starts a new event. According to this basic algorithmic approach several ideas have been proposed to improve the clustering procedure. More specifically:
   - [30, 31, 49] To improve the clustering accuracy they take into account entities like places, date, time and persons' names that are involved in each event.
   - [30, 49] They observed that events of specific topics use the same sets of words. Thus, before performing clustering they divided the documents into the respective topic categories.
   - [30, 49, 54, 55] Different reweighting in terms of each topic category, results in significant improvements in the clustering accuracy.
   - [36] Proper names, locations, temporal expressions and normal terms are extracted forming the corresponding ontology of each document. Thus, each document story is represented by these four sub-vectors and the similarity of two documents is conducted by comparing a pair of their corresponding sub-vectors.

2. Feature-Based approaches

   - [18] A feature pivot clustering algorithm is proposed in order to group bursty features. An event is signaled by the appearance of a group of bursty features.
   - [20] An unsupervised greedy event detection algorithm is applied in a pool of features, that have been categorized in the previous step in four non overlapping groups. The method is able to detect aperiodic and periodic events.

- [11] The bursty features are detected with the OMRBD algorithm, and then affinity clustering propagation is performed in order to form the events.
- [52] The burst of an event is considered as a set of bursty features, which are extracted from short text streams.
- [43] A keyword graph is proposed based on word co-occurrences in documents. Community detection methods, derived from social network analysis, are used to discover and characterize the events.

For the purposes of this work, we follow the document-based approach. Therefore, an event consists of a set of documents, thus our problem is transformed to a problem of documents clustering, with the difference that we should take into consideration the chronological sequence of documents. Therefore, the time factor adds one more dimension to the problem of clustering. Stories that discuss the same event tend to be in temporal proximity. Hence lexical similarity and temporal proximity are two major criteria for our problem. In addition, the high dimensional and sparse feature space in combination with language phenomena are challenges that affect the development of an effective algorithm. Moreover, in the case of event detection and tracking, because of the temporal and unpredictable nature of streams, the above issues complicate even more the problem of document clustering.

2.4.1 Document based techniques

As expected, conventional clustering methods are not suitable for detecting events in text stream, because the do not take into account the temporal relationship between the documents. It is likely, to be clustered to the same group, documents that have similar semantic content, but refer to different events. On the other hand, algorithms that have as input the number of clusters to be created are not appropriate. The predefined number of clusters can not cover the changing nature of information flow. It is almost unlikely to know in advance how many events will be identified because of the complex and unpredictable nature of text streams.

On-line methods

An important issue in on-line clustering (centroid and non-centroid) algorithms, is to determine the similarity threshold. To be more precise, a known approach in the online event detection is the single-pass clustering algorithm (figure 2.10).

However, as pointed out by Allan in [2] and verified by out experimental studies, a fixed threshold in the on-line process is not appropriate. On the contrary, if the threshold were adjusted each time to the 'energy' of the event, the results would be much more efficient. Intuitively, what we desire is a low threshold when an event is active to allow the clustering of documents without reducing significantly the accuracy of the clusters.

Moreover, when an event is inactive, by increasing the threshold the algorithm would avoid to group together irrelevant to the event documents.

Variations in this algorithm were proposed either by increasing similarity threshold over time [2], or using a time window which specifies the number of previous documents that should be taken into consideration for clustering [55]. But both approaches reach to the conclusion, that documents relating to events that spans a long time period, delegate to different groups.

Figure 2.10: Single pass document clustering

Event Detection Algorithm:

$E = null$;
For each news document d from on-line news stream
    $e = \text{ARGMAX}_{e \in E}(\text{sim}(e,d))$;
    If $\text{sim}(e,d) >=$ threshold then
        e.VectorUpdate(d);
    else
        $e_{new} = CreateNewsEvent(d)$;
        add e_{new} into E;
    end if
end for

In [9] the "life profile" for each event is proposed in order to select the appropriate value of similarity threshold. Moreover, in [10] the life cycle of each event is modeled by assigning an amount of "energy" to each one. This amount is declining over time, while it grows as the number of correlated with the event documents increases. The threshold is defined as the amount of energy of each event (figure 2.11).

Offline methods

In off-line event detection, because of the fact that we have available the whole corpus we could find statistical information for each term, concerning the flow of the text stream. In addition, we could define, from the beginning, the representatives of the events. All these urge that we could study the problem of event detection and tracking also from this perspective, the off-line one, as we don't have to deal with the difficulties of the on-line

version. Furthermore, we could take into consideration the burst intervals of each term. The conclusions of such a research could be useful for the on-line process because the development of an online event detection and tracking system, should first of all, deal with the memory issues and then proceed to the clustering procedure.

Figure 2.11: Energy-based Single Pass clustering

Energy-based Event Detection Algorithm:
E = *null*;
For each news document d **from on-line news stream**
    e = ARGMAX$_{e \in E}$(sim(e,d));
    If sim(e,d) >= threshold$_{data}$ **then**
        e.EnergyUpdate(d);
        e.VectorUpdate(d);
    else
        e_{new} = *CreateNewsEvent*(d);
        add e_{new} **into** E;
    end if
end for

The methods that have been proposed so far, introduce the bursty information in the text representation (VSM-BOW). Then, they are evaluated using classical clustering algorithms like spherical K-means.

2.4.2 Bursty Feature representation

A fundamental approach, in this direction is presented in [23], where the bursty feature representation is proposed. It is based on the bursty information of features as extracted from the two-state Klenberg's automaton. A document vector $d \in R^V$ is transformed as follows:

$$d_{ij}^{(t)} = \begin{cases} FP_{ij} + \delta w_j, \text{ if } f_i \in B \text{ and } t \in p_j \\ FP_{ij}\ , \text{ otherwise,} \end{cases} \quad (2.9)$$

where B denotes the set of bursty features, FP_{ij} the binary weight of term j in document i at time t, $\delta > 0$ the burst coefficient, w_j and p_j the bursty weight and its corresponding interval. So each word in the vocabulary may be either no bursty at all, or bursty at one

ore more non-overlapping time periods. As a result bursty words are strengthened by a factor of δw. In the experiments an improvement is observed in the cluster purity and entropy as well as the class entropy measure.

In their next work [21], the same authors tested the performance of their previous idea using the TDIFD weighting scheme with the normalized (w') and unormalized bursty weight (w), proposing the B-VSM representation with 5 different variations. The first two versions of B-VSM use the entire feature space $d \in R^V$ (equation 2.3)

$$\textbf{(SAB)}\ d_{ij} = \begin{cases} tfidf_{ij} + w'_j, if f_j \in \text{B} \\ tfidf_{ij}\ , \text{otherwise,} \end{cases} \quad (2.10)$$

$$\textbf{(SMB)}\ d_{ij} = \begin{cases} tfidf_{ij} \cdot w_j, \text{ if } f_j \in B \\ tfidf_{ij}\ , \text{otherwise,} \end{cases} \quad (2.11)$$

while the last three, use only the bursty feature space **B**.

$$d_i = [d_{i1}, \ldots, d_{i|B|}]^T \quad (2.12)$$

$$\textbf{(BAB)}\ d_{ij} = tfidf_{ij} + w'_j \quad (2.13)$$

$$\textbf{(BMB)}\ d_{ij} = tfidf_{ij} \cdot w_j \quad (2.14)$$

$$\textbf{(BT)}\ d_{ij} = tfidf_{ij} \quad (2.15)$$

One distinction from the previous research [23], is that in [21] only the largest bursty weight of each feature as its burstiness score are taken into account. Moreover the time stamp t of each document is not regarded in any of the five weighting schemes. As a result, the bursty weight of a feature f_j is applied to the whole document collection and not to the documents that are published during its bursty interval p_j. Their experimental results, on a subcorpus of the TDT3 dataset, reveal that the B-VSM is able to detect the top-k bursty topics as well as to improve significantly the recall and precision.

The method in [58] follows the similar concept with previous works [21, 23], aiming to mine, retrospectively, events from text streams through the procedure of clustering. The major contribution is the BurstVSM representation. In BurstVSM, the vocabulary contains only the bursty terms (equation 2.12) and no extra weight is added

$$\textbf{(Burst-VSM)}\ d_{ij}^{(t)} = \begin{cases} tfidf_{ij}, \text{ if t} \in p_j \\ 0\ , \text{otherwise,} \end{cases} \quad (2.16)$$

where p_j is the bursty period of feature j.

Moreover the bursty feature space is computed by parameterizing the Kleinberg's method, using a sliding window to recompute the probabilities p_0 and p_1. Last but not least, [58] deals with multiple bursts contrary to [21]. The benefits from taking into account the multiple bursty periods of each word are highlighted in the Table reftab:novel.

Table 2.1: Summary of representation models

	semantic information	temporal information	dimension reduction	trend modeling
VSM	✓	X	X	bad
boostVSM	✓	partially	X	moderate
BurstVSM	✓	✓	✓	good

table derived from [58]

We should notice that according to [23], a bursty-feature-only representation, although it reduces the feature space, frequently degenerates into a zero-vector due to the sparsity of the bursty features. Finally, a close method to the aforementioned works is [24]. Assuming that traditional vector space model cannot capture the temporal aspect of text streams, the bursty feature space is explored either with the Kleinberg's two state automaton or using the burst detection method proposed in [32]. The major contribution is the bursty distance measurement to calculate the similarity between a pair of documents, as well as the local burstiness score, based on the local word occurrence.

2.5 Spherical K-means

K-means is a fast and easy to understand clustering algorithm. It assumes a representative of each cluster and an objective function that evaluates the quality of each partition. Given a dataset and the number **K** of desired clusters, the algorithm, initially selects K-representatives, usually by random, as the centers of the clusters. After that, each object (document) is assigned to the partition with the closest center. Then, the new centers are recalculated and the algorithm repeats the assignment of all objects (documents) to the new cluster representatives. The above procedure could be summarized in the following two steps.

1. Reassignment step: each object(documents) is assigned to the cluster whose center is nearest to it

2. Update step: Each cluster representative is updated in such a way that the objective function is optimized in every algorithmically step.

The k-means converges because in each iteration is created a more homogenous partition (or the clustering error is reduced if take into account the distance measure), terminating at a local maximum(minimum) where no more changes in the clusters could be occurred. The main disadvantage of the algorithm is that although it converges quickly and monotonically, it results in a local minimum (or maximum), depending on the initialization of the centers.

Its time complexity is O(tNV), where $t \leq MAX\_ITER$ is the number of iterations until convergence. N is the number of documents and V the length of the vocabulary. When the centers are computed as the arithmetic mean (centroids) the K-means minimizes the sum of sum of squared euclidean distances between the objects of the cluster and the centroids. Another well-known method is K-medoid where a cluster is represented with the medoid object, defined as the one that has the maximum average similarity to the objects of its cluster.

Spherical k-means(spk-means) is a variant of k-means that utilizes the cosine similarity for the data vectors normalized with respect to L2-norm. The maximized objective function is the clustering cohesion. The optimal representative for a cluster is its normalized centroid and the overall clustering cohesion of a partition C is given by:

$$Cohesion(C) = \sum_{j=1}^{\kappa} \sum_{d_i \in c_j} (r_{c_j})^T d_i \tag{2.17}$$

where r_{c_j} is the representative (normalized centroid) of cluster c_j.

CHAPTER 3

CORRELATED BURSTY TERMS

3.1 Terminologies

3.2 Term similarity matrices

3.3 Correlated Bursty Term Clustering

3.1 Terminologies

As text stream is considered the appearance of documents according to a predifined time-line

$$Stream = [S_1, \ldots, S_T] \tag{3.1}$$

where S_i represents the set of documents with the *i-th* time-stamp. We denote a corpus of text streams with N documents as X, and T the time period that it spans. Each document $d_i^{(t)}$ that appears in one of the T batches, is represented as a vector of V features using the TFIDF representation.

For the identification of the bursty feature space B, we used the Kleinberg's two-state automaton, that returns the bursty weight w_j and interval p_j of each feature f_j. Although the value of equation 2.2 is a positive number, this does not imply that the individual weights of each time slot that is enclosed in the interval [t1,t2] are all positive.

$$w^t = \sigma(0, r_t, d_t) - \sigma(1, r_t, d_t) \tag{3.2}$$

After experimental efforts, we observed that the following burst-reweighing scheme works better for our data sets.

$$d_{ij}^{(t)} = \begin{cases} tfidf_{ij} \cdot w_j, \text{ if } f_j \in B \text{ and } t \in p_j \\ tfidf_{ij}\ , \text{otherwise,} \end{cases} \quad (3.3)$$

where w_j is given by 2.2. Thus, after applying the above equation to the document collection X, we have at our disposal the corpus representation XB given by 3.3.

3.2 Term similarity matrices

The experiments in all previous works [5,12,14], that purpose different variations of the term similarity matrix, conducted for static document collections. On the other, none of the works [21,23,58], that study the problem of text stream clustering, use the correlation between features in order to improve clustering performance.

In our research, we are interested in studying the problem of topic detection and tracking in a retrospective way(offline). Using the vector space bag of words model, we try to exploit correlations of busrty words, in order to form term similarity matrices that would be able to improve the clustering of text streams. The term similarity matrix of the GVSM $(X'X)$ captures the relation of terms that co-occur in the same documents. Initially, we propose the SIM_{GVSM}^{bursty} term similarity matrix, which contains not only the co-occurrence but also the co-burstiness of terms.

$$SIM_{GVSM}^{bursty} = XB' \cdot XB \quad (3.4)$$

Note that the SIM_{GVSM}^{bursty} and SIM_{GVSM} matrices enclose only the relations of terms that appear together, in at least one document. However, there are documents that contain totally different words which are semantically related. We claim that if such words are bursty and refer to the same event, their intervals of high frequency would be overlapping. Such a similarity term matrix could capture not only the co-bursty features that co-occure in the same document vectors, but also features that happen to be bursty during the same period. For this reason we propose the $SIM1_B$ as well as $SIM2_B$ similarity matrices.

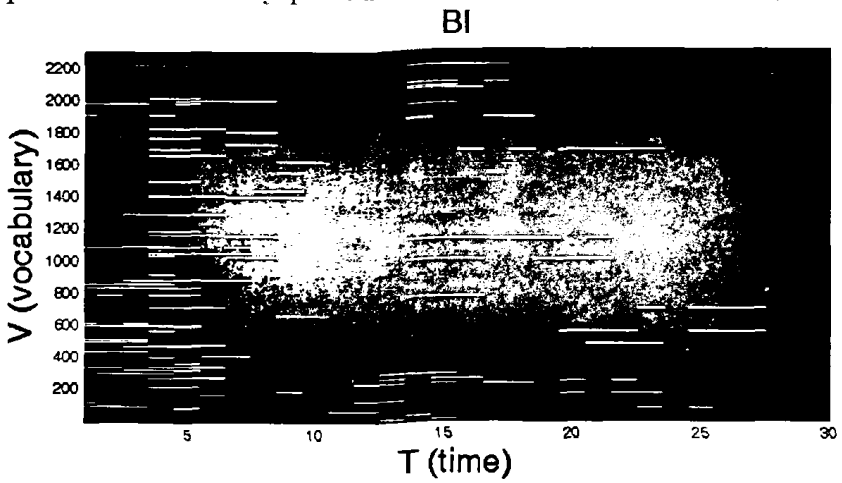
In order to construct the two matrices, first we need to create the BI matrix, of dimension $V \times T$ (figure 3.1), which indicates the time slot a feature is bursty and its corresponding burstiness score.

$$BI(i,j) = \begin{cases} 0 \text{ if } f_i \notin B \\ \\ w_i^j, w_i^j > 0 \text{ if } f_i \in B \end{cases} \quad (3.5)$$

where w_i^j is the burst weight of feature i at time slot j according to equation 3.2.

Figure 3.1: An example of BI

The white spaces are the bursty periods of each term. that the bursty weight is positive.



$$SIM1_B = BI \cdot BI' \tag{3.6}$$

$$SIM1_B(i,j) = \sum_{\kappa=1}^{T} BI_{i\kappa} BI'_{\kappa j} \tag{3.7}$$

$$SIM2_B = SIM1_B/T - A \tag{3.8}$$

$$A = (G' \cdot G)/T \tag{3.9}$$

$$G(i) = \begin{cases} 0, \text{ if } \mathrm{f}_i \notin B \\ \text{or} \\ \sum_{j=1}^{T} BI(i,j), \; if f_i \in B \end{cases} \tag{3.10}$$

In a text stream, it is common for some events to overlap in time. For example the "Ebola virus epidemic in West Africa" and "XXII Olympic Winter Game" events, emerged according (to Wikipedia[1]) in news sources on February of 2014 . Moreover the probability the bursty features of these events to occur in the same time slots is very high. However, the conceptual content of such events is totally different. In such cases the $SIM1_B$ and $SIM2_B$ similarity matrices would capture the co-burst relation of terms that refer to semantically unrelated events. To tackle this limitation, we propose the $SIM1_B^{cor}$ and $SIM2_B^{cor}$ term matrices.

[1] http://en.wikipedia.org/wiki/2014

$$SIM1(i,j)_B^{cor} = SIM1_B(i,j) \cdot COR(i,j) \tag{3.11}$$

$$SIM2(i,j)_B^{cor} = SIM2_B(i,j) \cdot COR(i,j) \tag{3.12}$$

where COR is the matrix that contains the correlation coefficients from X. Although the correlation matrix has been proposed in [14], in our case we are interested only in the bursty terms. All the above term matrices are symmetric, non negative and positive definite. However in order to apply them in a document collection we normalize their columns. The normalization is done by dividing the elements of each column by the square root of the euclidean norm of term weights. As it is anticipated, after normalization the matrices are no longer symmetric.

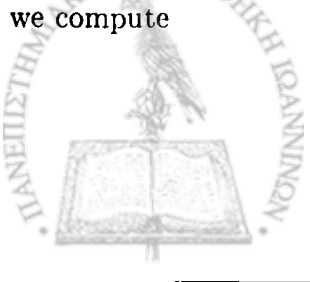
The size of all matrices is $O(V^2)$. The advantage of the above matrices, contrary to SIM_{GVSM}^{bursty} and SIM_{GVSM}^{bursty}, is that they are more sparse since they contain only the co-burstiness and not the co-occurrence relation of the words. The disadvantage of all matrices is that the memory requirements is large enough, especially when the number of terms increases. However, this could be handled with semantic kernels. But such a study is beyond the scope of this thesis.

3.3 Correlated Bursty Term Clustering

The bursty interval of bursty features denotes the period during which the event associated with them is "hot". In other words, the majority of document stories that discuss the same event are published around this time period, thus their temporal proximity tends to be close. Furthermore and according to Kleinberg's burst detection method (batch mode), in order for a term to be characterized as bursty, it depends on the number of related documents. The larger the number of documents that contain the candidate term, the larger the probability this term to be part of B. At last, the fact that a great amount of research work that study the problem of topic detection follow the feature based approach (via the discovery of bursty words) make us to do the following assumption.

The bursty terms could suggest to us to the most important documents of each event. We regard as important, a document that is close to the centroid (or medoid) of the documents that are derived from the same class. The discovery of such documents, is expected to improve the performance of all algorithms that are part of the K-means family, because their results depend on the selection of the initial cluster centers.

The first part of the method, we propose for the selection of the K centers, is based on the philosophy of the feature-based methods. i) Initially, we create K' group of features applying the Spectral Clustering algorithm to a bursty term graph, then ii) we compute



the K'-representatives using the synthetic prototypes procedure [25]. The representatives are computed from the set of documents that contain the bursty words and are published during their bursty interval. Moreover, we assign these documents ($Docs_B$) to the nearest cluster representatives forming K' clusters of documents. iii) At the end, we reduce the number clusters from K' to K by merging the most similar clusters according to the cosine similarity of their representatives. The pseudo-code of our method follows.

---

function CBTC(D, Pdocs, Pterms, λ, β, K, K', adj)

1. $c^{(f)} \leftarrow GraphCuster(adj, K')$
2. $\{SP, y_c, K'\} \leftarrow ConstructBurstySP(c^{(f)}, D, Pdocs, Pterms, \lambda, \beta\ )$
3. $\{SP\} \leftarrow MergeClusters(y_c, SP, K, K', Pdocs, Pterms, \lambda, \beta)$

3.3.1 Bursty Term Graph

In order to cluster the bursty features, we should define a relation between them. For that reason, we decided to create a term co-occurence graph from the set of bursty terms B. In literature there are several works following this concept either for event detection or topic summarization.

In [39] an event detection algorithm is proposed using a keyword co-occurence graph.In [43] a graph is created, from noun phrases and name entities, and then a community detection algorithm is applied to discover events. However, none of the these works consider the burstiness of each feature. Moreover in [37] a network is built for a specific trending topic, where nodes are bursty and nonbursty words and an edge denotes a co-occurence relation. In addition, in [50] the problem of "bursty event tagging" is studied where an event is described from a set of tags. The authors observed that tags from various web sources reflect the users' interests over time, thus by applying graph clustering techniques they detected bursty events. Finally a more sophisticated method for event detection from social text streams, like blogs and emails is presented in [57].

Our graph network G(V,E) consists of bursty terms only ($|B| = |V|$). Nodes correspond to terms, while an edge e_{kj} between two nodes n_k and n_j is added, if the following two conditions, for terms w_k and w_j, are satisfied:

1. Their bursty intervals overlap.
2. During the overlapping period, the terms w_k and w_j co-occur in at least one document.

The weight of each edge is derived from [11, 44]:

$$w(k,j) = \frac{1}{2}\left(\frac{|D_k \cap D_j|}{|D_k|} + \frac{|D_k \cap D_j|}{|D_j|}\right) \tag{3.13}$$

where D_k is the set of documents that include term f_k and are published in its bursty interval. Vertices with degree less than 1 are eliminated from the graph. Using the resulting graph, we apply the Spectral Clustering algorithm [38] to partition the bursty terms into K' non-overlapping clusters. Finally, we omit groups with less than two terms.

---

function *GraphCluster*(adj, K')

input : an adjacent matrix adj that represents the structure of the bursty-features graph and K' the number of desired clusters

output : $c^{(f)} = \{c_1^{(f)}, \ldots, c_{K'}^{(f)}\}$ the clustering solution with K' groups of terms

1. $c^{(f)} \leftarrow SpectralClustering(adj, K')$
2. $c^{(f)} \leftarrow c^{(f)} - \{c_i^{(f)}, ..., \forall i = 1 : K', |c_i^{(f)}| < 2\}$
3. $K' \leftarrow |c^{(f)}|$
4. **return**$(c^{(f)})$

3.3.2 Cluster representatives

In the second step of our algorithm, we determine the documents that are related to each bursty term. The association criterion is for a bursty term to appear in a document at least once. The considered documents of each term, are only those that are published during its bursty interval.

Then, the representative of each group of terms is computed, from the documents that are part of the set *Docs*.For this task, we used the synthetic prototypes [25] instead of centroids or medoids. This is due to the fact that, i) the set *Docs* may possess little amount of documents, ii) the high dimensionality and sparsity of the data and iii) the examined set could possibly contain documents from different topics, therefore we require a method that would compute the representative of the cluster from documents of the dominant class. For these reasons, we chose the synthetic prototypes representation.

Given a cluster of documents, the algorithm computes its medoid as the document with the maximum average similarity to the objects of the cluster. This is the first representative. Then, selects iteratively the closest documents to the representative. The number of the closest documents is defined as a percentage ($Pdocs$) of the cluster size. At

the end of each iteration, it calculates the new representative as the centroid of the selected documents. Finally, the method chooses the most highly weighted features (*Pterms* percentage of the vocabulary length) to construct the synthetic prototype.

We should notice, that the set *Docs* is changing at every iteration. Moreover, a document may contain multiple bursty terms, which are clustered in different groups during the first step. Thus, it is possible a document to appear more than one iteration in the set *Docs*. All the documents that are published during the bursty intervals, and contain at least one bursty term, are stored in the variable $Docs_B$. The set $Docs_B$ is the pool of documents from which we are going to create the document clusters.

In the last step, we cluster the documents that belong to the set $Docs_B$ into K' groups. This is performed by assigning each document to the cluster with the closest representative. The similarity measure that is used to perform the comparison is the cosine similarity (equation 2.5). By performing this clustering step, we partition the set $Docs_B$ into K' disjoint groups. In that way, we associate the clusters of bursty terms with groups of documents.

---

function $ConstructBurstySP(c^{(f)}, D, Pdocs, Pterms, \lambda, \beta)$

input : $c^{(f)}$ is the clustering solution of function GraphCluster
D is the text stream collection
Pdocs, Pterm, λ and β parameters for the synthetic prototype method

output : $SP = \{SP_1, \ldots, SP_{K'}\}$ the set of synthetic prototypes
$y_c = \{y_{c1}, \ldots, y_{K'}\}$ the clusters of documents

let : f_k denote the k-feature, $f_k \in B$
$c_i^{(D)}$ the i-th group of document with $1 \leq i \leq K'$
D_{f_k} is the set of documents containing term f_k
ConstructSP: method for the syntetic prototypes
AssignToClosest: clusters the documents $Docs_B$ to the K' closest centers (SP)

end let

1. $Docs_B \leftarrow \oslash$
2. **for** $i = 1 \ldots K'$
3. Docs $\leftarrow \oslash$
4. **for each** $f_k \in C_i^{(f)}$
5. Docs $\leftarrow Docs + \{D_{f_k}\}$
6. **end for**

7. $Docs_B \leftarrow Docs_B + \{Docs\}$

8. $SP_i \leftarrow ConstructSP(Docs, Pdocs, Pterms, \lambda, \beta)$

9. **end for**

10. $y_c \leftarrow AssignToClosest(SP, Docs_B)$

11. **return** (SP, y_c)

3.3.3 Merging step

In the last step of the algorithm, cluster reduction is performed. In each iteration, the clusters with most similar representatives are merged forming a new cluster. After that, the new representative is calculated, using again the synthetic prototype procedure. The procedure ends until the number of clusters is equal to the desired number.

---

function MergeClusters(y_c, SP, K, K', Pdocs, Pterms, λ, β)

input : y_c and SP are the outputs of function ConstructBurstySP
K is the number of clusters in which we would like to reduce the set y_c
Pdocs, Pterm, λ and β are the parameters of constructs of SP method
output : SP: synthetic prototypes centers
let : ClosestCentres: method that find the most similar synthetic centers
Merge: method that merges two clusters
end let

1. **repeat**

2. $\{\kappa, e\} \leftarrow ClosestCentres\ \{S, P\}$

3. $y_{c_{\kappa e}} \leftarrow Merge\{y_{c\kappa}, y_{ce}\}$

4. $y_c \leftarrow y_c - \{y_{c\kappa}, y_{ce}\} + \{y_{c_{\kappa e}}\}$

5. $SP_{\kappa e} \leftarrow ConstructSP(y_{c_{\kappa e}}, c, Pdocs, Pterms, \lambda, \beta)$

6. $SP \leftarrow SP + \{SP_{\kappa e}\} - \{SP_\kappa, SP_e\}$

7. $K' \leftarrow K - 1$

8. **until** $K' \equiv K$

9. **return** SP

CHAPTER 4

EXPERIMENTS

4.1 Data-sets

4.2 Text Stream Generator

4.3 Experimental Method

4.4 Experimental Results

4.1 Data-sets

In our experiments, we used 4 different text datasets. D1-D2 are subsets of the 20-NewsGroups by choosing randomly, 100 documents from each selected category. D3 is a version of the Reuters-21578 benchmark text collection, by selecting the 10 top-sized categories. From each category we chose 100 documents at random, except for the last one, where we took all its documents as its size is below 100.

D5 is a subset of GoogleNews dataset [1], that contains English-written articles from the "Technology" category. From this text collection we kept the classes with more than 20 documents and we extracted the main content from each article. More details about this dataset and the way it was annotated, can be found in [28].

D4 is a subset of TDT5 [2] text collection, which contains 250 topics gathered form 15 different newswire sources between April and September of 2003. The 75% of the topics are monolingual (English, Arabic or Mandarin Chinese) and the rest multilingual. From this dataset we kept the English-written documents and those appearing in two or more

[1] http://www.db-net.aueb.gr/GoogleNewsDataset/

[2] https://catalog.ldc.upenn.edu/docs/LDC2006T19/TDT2004V1.2.pdf

categories were removed. From the resulting categories, we considered only those with more than 50 document stories.

Pre-processing of a raw text collection is a required step before applying any clustering algorithm. A standard method consists of two steps. Initially stop-words, numbers and alphanumerics are eliminated. Then, the stemming algorithm [41] is applied, in order to replace each word by its corresponding word stem. The derived word stems form the vocabulary of the text collection. For the pre-processing, we used the "Text to matrix Generator toolkit" [53].

To reduce the feature space in D1,D2 and D3, we ignored the words that appear in less than 5 documents, while the threshold, for D4 and D5, was set to 3. For each of the last two datasets, we calculated three quantiles from the terms document frequency distribution. In that way, the distribution was divided in three equal parts. We excluded the corner parts (terms with very high and very low document frequency) and we kept the rest words for our vocabulary. The documents with no words, after the previous preprocessing, were not taken into consideration. The characteristics of the above text datasets are presented in Table 4.1, while Tables 4.4 and 4.3 present the events and the topics from TDT5, GoogleNews, 20-NewsGroups and Reuters-21578 datasets respectively.

Table 4.1: Characteristics of text document collections

Name	Classes	N	V	Balance	V'
D1	10	1000	2352	1	45,89
D2	10	1000	2310	1	44,54
D3	10	993	1566	0,93	44,16
D4	30	4972	4717	0,06	21,54
D5	11	268	1298	0,4318	59,07

N denotes the number of documents, V the size of the vocabulary, V' the average document vocabulary and Balance the ratio of the smallest to the largest class

The pre-defined timeline of the TDT5 and GoogleNews datasets was used and documents that were published the same day were inserted in the same batch. Because of the lack of timestamps in the 20-NewsGroups and Reuters-21578 data, we used a simulation method in order, artificially, to produce topic bursts.

Table 4.2: Selected topics for D1,D2 and D3

Dataset	Source topic
D1	20-NGs: graphics, windows.misc, pc.hardware, mac.hardware, windows.x, autos, motorcycles, politics.guns, politics.mideast, politics.misc
D2	20NGs: atheism, graphics, ibm.pc.hardware, forsale, autos, sport.baseball, crypt, religion.christian politics.guns, politics.misc
D3	Reuters -21578: acq, corn, crude, earn, grain, interest, money-fx, ship, trade, wheat

Table 4.3: D5 - subset of GoogleNews

id	topicID	Name
1	65	AT&T Unveils Shared Wireless Data Plans
2	186	Apple Considered Investing in Twitter
3	15	Google Nexus 7 tablet goes on sale in US
4	555	VMware buys Nicira for $1.05 billion
5	646	Google unveils price for gigabit Internet service
6	5	Digg acquired by Betaworks
7	252	Microsoft Reboots Hotmail As Outlook
8	425	FTC Fines Google for Safari Privacy Violations
9	454	Nokia cuts Lumia 900 price in half to $50
10	19	Apple Brings Products Back Into EPEAT Circle
11	496	Yahoo confirms 400k account hacks

Table 4.4: D4 - subset of TDT5

id	topicID	Name
1	55005	Sosa ejected, cheating suspected
2	55012	National Do Not Call Registry
3	55016	Gay Bishop
4	55029	Swedish Foreign Minister killed
5	55047	Kobe charged with sexual assault
6	55063	(SARS) Quarantined medics in Taiwan protest
7	55069	Earthquake in Algeria
8	55072	Court indicts Liberian President
9	55076	Protests at 2003 Masters Tournament
10	55078	Looting at Iraqi nuclear site
11	55080	Spanish Elections
12	55087	Earthquake in Turkey
13	55089	Liberian former president arrives in exile
14	55090	Blackout in US and Canada
15	55098	Bush and Blair Summit
16	55103	Two Britons among terror suspects
17	55105	UN official killed in attack
18	55106	Bombing in Riyadh, Saudi Arabia
19	55107	Casablanca bombs
20	55109	Israel withdraws troops from Gaza
21	55117	Cambodian Elections
22	55118	World Economic Forum in Jordan
23	55125	Sweden rejects the Euro
24	55128	Mad cow disease in North America
25	55155	Chinese Submarine Accident
26	55166	Suicide bombers hit Moscow concert
27	55181	Palestine: Ahmed Qureia tapped as next prime minister
28	55200	Iraq: Protection of antiquities
29	55227	Bin Laden Videotape
30	55240	US troops fire on Mosul crowd

4.2 Text Stream Generator

The major assumption we made in order create topic bursts, is that documents from the same class follow an exponential distribution. Thus, because of the topic bursts, bursts of terms would be produced too. In every time window a number of documents is assigned. The probability a term to characterized as bursty in each time window depends on the

number of documents that contain it. Below the pseudocode of the text stream generator is displayed[3].

The input of the method, consists of the length of the stream timeline T, the number of topics K and the maximum number of bursts per topic ($bursts_{max}$). Furthermore, the maximum and minimum lambda of the exponential distribution (l_{min}, l_{max}) as well as the maximum and minimum percentage of documents that participate in the topic bursts, are input of the generator (rd_{min}, rd_{max}). The exact number of bursts, documents and the lambda of each burst is chosen randomly in the respective intervals.

---

function TextStreamGenerator(T, K, IDS, $l_{min}, l_{max}, bursts_{max}, rd_{min}, rd_{max}$)

input : T the text stream period
K: the number of topics
IDS: the id of each document
l_{min}, l_{max}: min and max lambda of the exponential distribution
$bursts_{max}$: max number of bursts
rd_{max}: max ratio of stories in the bursty topic intervals
rd_{min}: min ratio of stories in the bursty topic intervals

output : text stream S(equation 3.1)

let : $Numbers_{bursts}$ denote the number of bursts per topic
$docs_{ratio}$ the percentage of documents from each category in the bursty intervals
select_random: method that selects randomly documents from a category
diff: method that returns the difference between two sets
distribute: method that distributes the selected document ids into T batches according to an exponential (or randomly)

end let

1. **for** i = 1 ... K
2. $Number_{bursts} \leftarrow rand(1, \text{bursts}_{max})$
3. $docs_{ratio} \leftarrow rand(rd_{min}, rd_{max})$
4. $Docs \leftarrow select\_random(C_i, docs_{ratio})$
5. $Docs_{rest} \leftarrow diff(C_i, Docs)$
6. **for** $j = 1 \ldots Numbers_{bursts}$
7. $lambda \leftarrow rand(\, l_{min}, l_{max})$

[3]It should be noted that the code is developed and maintained by the postdoctoral researcher Argyris Kalogeratos, email : kalogeratos@cmla.ens-cachan.fr

8. $S \leftarrow$ distribute $(T, lambda, Docs)$

9. $S \leftarrow$ distribute $(T, random, Docs_{rest})$

10. **end for**

11. **end**

12. **return**

4.3 Experimental Protocol

We compared our similarity matrices, SIM^{bursty}_{GVSM} - $SIM1_B$ - $SIM2_B$ - $SIM1^{cor}_B$ - $SIM2^{cor}_B$, with the GVSM matrix as well as the static VSM (tfidf) and the bursty feature representation (equation 3.3). We designed three experiments. The first experiment was conducted on the last two datasets D4 and D5, where the text stream was created based on the documents' daily timestamps.

For the D1, D2 and D3 datasets, due to the lack of documents' timestamps, we designed two experiments using the stream-generator to create topic bursts. In the first experiment *(Experiment A)*, the duration T was set to 30 and in the second *(Experiment B)* to 90 time units. Furthermore, in the first case, the number of bursts per topic was one or two, while in the second all topics had only one burst.

Table 4.5: Parameters of the Text Stream Generator

Experiment	A	B
T	30	90
lamba	[0.2:0.9]	[0.2:0.9]
#bursts per topic	1-2	1
%docs in bursts	0.7-0.9	0.7-0.9

For each burst, the parameter lambda of the exponential distribution was selected, randomly, from the interval [0.2:0.9]. We chose, this interval, because values higher than 1 restricted a burst in only one time window, while values close to zero sustained a burst over many time slots. The percentage of documents, from each topic, that would be attached to the bursty intervals was picked, randomly, from 0.7 to 0.9. Finally, each experiment was executed 5 times, by randomly initializing the stream generator.

The reason, we designed our experiments in that way, is to study the proposed similarity matrices in two kind of situations; in "complicated" (*Experiment A*) and "more relaxed" (*Experiment B*) text streams. In the first case, the overlapping between topics is higher. While in the second situation, the probability of simultaneously occurring two topic bursts in the same timeslot is lower. For that reason we introduce a supervised metric, the entropy of a text stream as the mean of entropy of all windows:

$$H_s = \frac{\sum_{t=1}^{T} Hs^t}{T} \tag{4.1}$$

$$H_s^t = -\sum_i (n(c_i^t)/n^t) \cdot log_2(n(c_i^t)/n^t) \tag{4.2}$$

where n^t is the number of documents in the t-th batch and $n(c_i^t)$ the number of documents from class c_i.

Table 4.6: Stream Statistics

Name	V	B	m	H
D1-T=30	2352	289,4	33,3	2,1157 ±0,9035
D1-T=90	2342	597	11,4	0,8161 ±0,7243
D2-T=30	2310	276	29,2	2,31 ±0,891
D2-T=90	2310	555	11,1	0,8903 ±0,7132
D3-T=30	1566	290,6	33,1	2,1798 ±0,8833
D3-T=90	1566	544	11	0.8387 ±0.7442
D4-T=183	4717	4020	23,8483	2,0529 ±0,5811
D5-T=31	1298	400	8,6452	0,2369 ±0,5429

V denotes the vocabulary size, B the number of bursty terms, m the average number of documents per timestamp and H the stream entropy

The new image of a document, after applying on it a term similarity matrix SIM, is given by the product $d' = d \cdot SIM$. Therefore, the image of a text collection is the product $X' = X \cdot SIM$. We multiplied all the similarity matrices with representations X and XB.

To assess the quality of each matrix, we tested each mapping document collection using the spherical k-means algorithm. The pre-defined number of K-clusters corresponds to the number of events in a collection. In addition, the algorithm ran 100 times and each time the k centers were randomly initialized, but all representations were booted up using the same random document seeds.

For the evaluation of the CBTC algorithm, we conducted experiments on all five data sets running the spherical K-means in X and XB representation. For the D1, D2 and D3 data sets we used the parameters of *Experiment A* executed it 1 time, by randomly initializing the text stream generator. Moreover all methods were initialized using the same random document seeds.

4.3.1 Evaluation Metrics

The evaluation was based on three different supervised measures. At this point we define the following:

- N is the number of docs
- C is the cluster solution of k-cluster $c_1, ..., c_\kappa$
- c^p is the portion of documents according to their true class labels $c_1^p, ..., c_\kappa^p$
- N_i, the size of c_i^p
- n_i, the size of c_i
- n_{ij}, the number of documents that are clustered to c_j but they belong to c_i^p.

Purity

Purity can be interpreted as the classification accuracy, if all the samples of a cluster are predicted to be members of the dominant class.

$$Purity(c) = \frac{1}{N}\sum_{j=1}^{\kappa} max\{n_{ij}\} \tag{4.3}$$

F1-measure

F1 is the harmonic mean of precision ($P = \frac{TP}{TP+FP}$) and recall ($R = \frac{TP}{TP+FN}$), where TP, FP and FN denote True Positive, False Positive and False negative respectively.

$$F_1 = \frac{2 \cdot P \cdot R}{P + R} \in [0, 1] \tag{4.4}$$

Lower (higher) values of F_1 indicate worse (better) solution.

Normalized Mutual Information

The numerator is the Mutual Information measure and the denominator is the maximum between cluster and class entropy.

$$NMI = \frac{\sum(\frac{n_{ji}}{N})log2\frac{(\frac{n_{ij}}{N})}{(\frac{n_i^P}{N})(\frac{n_j}{N})}}{max\{H(c), H(c^P)\}} \tag{4.5}$$

We report the average value (denoted as 'avg') of each metric over the runs on a dataset as well as the best value (denoted as 'best') corresponding to the solution with the the highest clustering objective function (equation 2.17) among the 100 runs.

4.4 Experimental Results

- **Term-Similarity matrices**

GoogleNews

Table 4.7: Term-Similarity matrices results on the D5 using spherical K-means

	avg. Purity	**avg. F1**	**avg. NMI**	**best Purity**	**best F1**	**best NMI**
X	0,55694	0,56595	0,47404	0,56343	0,5928	0,5156
XB	0,79388	0,79319	0,77183	0,89552	0,89624	0,84097
X·SIM^{GVSM}	0,55728	0,58258	0,50198	0,58582	0,62992	0,5359
XB·SIM^{GVSM}	0,7194	0,74308	0,70972	0,83582	0,85363	0,79786
X·SIM$^{bursty}_{GVSM}$	0,7222	0,74389	0,69454	0,81716	0,83802	0,76973
XB·SIM$^{bursty}_{GVSM}$	0,78138	0,78692	0,77716	0,92164	0,92246	0,87379
X·SIM1$^{cor}_{B}$	0,70216	0,72757	0,66245	0,74627	0,78115	0,70301
XB·SIM1$^{cor}_{B}$	0,80679	0,81843	0,81033	0,9403	0,94077	0,90035
X·SIM2$^{cor}_{B}$	0,69616	0,72219	0,65649	0,78731	0,80006	0,71304
XB·SIM2$^{cor}_{B}$	**0,80847**	**0,81924**	**0,81064**	**0,95522**	**0,95472**	**0,92034**
X·SIM1$_B$	0,65396	0,68157	0,63284	0,65299	0,70403	0,65002
XB·SIM1$_B$	0,75784	0,78265	0,77505	0,83209	0,83747	0,8235
X·SIM2$_B$	0,65649	0,68351	0,63315	0,70149	0,72777	0,65982
XB·SIM1$_B$	0,76575	0,7893	0,78022	0,88433	0,89287	0,86173

We observe that the representation XB·SIM$^{bursty}_{GVSM}$ outperformed all the others. Probably, because in this text stream the events are well-separated. Moreover, from all

the datasets, this is the one with the lowest stream entropy.

TDT5

The D4 dataset is an imbalanced text collection, as there are some events with many and others with limited document stories. In that case the Purity is a biased metric as it favors the dominant class. For this reason we should pay attention to the NMI metric, in order to trade off the quality of the clustering, against the number of clusters. Although the best solution, for the spk-means algorithm, was accomplished using XB·SIM$_{GVSM}$, we should also notice the performance of XB·SIM1$_B^{cor}$ that achieved slightly less NMI, but better F1 score. On the other, the XB·SIM$_{GVSM}^{bursty}$ representation, gave the highest average measurements.

Table 4.8: Term-Similarity matrices results on the D4 using spherical K-means

	avg. Purity	avg. F1	avg. NMI	best Purity	best F1	best NMI
X	0,56383	0,51125	0,58706	0,61826	0,5767	0,62348
XB	0,61354	0,55601	0,6412	0,66895	0,63935	0,6714
X·SIM$_{GVSM}$	0,67542	0,62208	0,68046	0,73994	0,68698	0,71128
XB·SIM$_{GVSM}$	0,71793	0,64839	0,72391	0,76126	0,71067	**0,75881**
X·SIM$_{GVSM}^{bursty}$	0,69997	0,63661	0,69935	0,73632	0,69522	0,72526
XB·SIM$_{GVSM}^{bursty}$	**0,72068**	**0,65254**	**0,72698**	0,75201	0,69625	0,75144
X·SIM1$_B^{cor}$	0,68054	0,62268	0,68317	0,70374	0,66334	0,70293
XB·SIM1$_B^{cor}$	0,70582	0,63788	0,71551	**0,76287**	**0,72038**	0,75339
X·SIM2$_B^{cor}$	0,67924	0,62238	0,68213	0,70716	0,67449	0,70543
XB·SIM2$_B^{cor}$	0,70594	0,63856	0,71516	0,75905	0,7156	0,74873
X·SIM1$_B$	0,61754	0,55369	0,63469	0,64642	0,57477	0,65389
XB·SIM1$_B$	0,62709	0,55621	0,65211	0,64019	0,55793	0,66445
X·SIM2$_B$	0,61749	0,55452	0,63513	0,65648	0,58328	0,65463
XB·SIM2$_B$	0,62808	0,55715	0,65311	0,63717	0,55614	0,66371

Reuters

In the *Experiment B* (T=90), where the text stream is created from the D3 dataset, the XB representation was the best. But things were little different in Experiment A, where the stories are published in narrower time period (T=30). The reported best solution, with max Purity and F1, was derived from the same method as before (XB), while the maximum NMI from X·SIM1$_B^{cor}$ similarity matrix. In addition, the X·SIM2$_B^{cor}$ produced the highest average measurements for all three metrics.

Table 4.9: Term-Similarity matrices results on the D3 using spherical K-means

	T=30					
	avg. Purity	**avg. F1**	**avg. NMI**	**best Purity**	**best F1**	**best NMI**
X	0,77058	0,77449	0,74477	0,8711	0,86384	0,82431
XB	0,79441	0,80148	0,75424	**0,89648**	**0,89537**	0,82597
X·SIM$_{GVSM}$	0,75265	0,76099	0,72799	0,83887	0,82763	0,78172
XB·SIM$_{GVSM}$	0,77239	0,78397	0,73725	0,8433	0,83686	0,78152
X·SIM$^{bursty}_{GVSM}$	0,785	0,80196	0,77217	0,86485	0,86639	0,82157
XB·SIM$^{bursty}_{GVSM}$	0,78437	0,80077	0,75364	0,8719	0,87698	0,80743
X·SIM1$^{cor}_{B}$	0,79639	0,80745	0,77461	0,88419	0,88724	**0,83861**
XB·SIM1$^{cor}_{B}$	0,79047	0,80189	0,74171	0,88218	0,88126	0,80103
X·SIM2$^{cor}_{B}$	**0,7969**	**0,80795**	**0,77549**	0,87029	0,87261	0,83362
XB·SIM2$^{cor}_{B}$	0,79158	0,80287	0,7441	0,88379	0,88239	0,80426
X·SIM1$_{B}$	0,70368	0,72187	0,66663	0,76918	0,78179	0,71477
XB·SIM1$_{B}$	0,66448	0,67963	0,59367	0,69144	0,69447	0,6092
X·SIM2$_{B}$	0,70989	0,72733	0,67194	0,78691	0,7974	0,72769
XB·SIM2$_{B}$	0,66713	0,68125	0,59614	0,6856	0,69201	0,60591
			T=90			
X	0,77058	0,77449	0,74477	0,8711	0,86384	0,82431
XB	**0,82896**	**0,8411**	**0,82602**	**0,92991**	**0,92993**	**0,88855**
X·SIM$_{GVSM}$	0,75265	0,76099	0,72799	0,83887	0,82763	0,78172
XB·SIM$_{GVSM}$	0,79833	0,81848	0,79222	0,8709	0,88096	0,84261
X·SIM$^{bursty}_{GVSM}$	0,80139	0,82038	0,7921	0,86244	0,86954	0,83152
XB·SIM$^{bursty}_{GVSM}$	0,7888	0,8126	0,79615	0,89063	0,89701	0,86274
X·SIM1$^{cor}_{B}$	0,79728	0,81459	0,78368	0,87372	0,87071	0,82908
XB·SIM1$^{cor}_{B}$	0,78709	0,8057	0,78939	0,90413	0,90213	0,85801
X·SIM2$^{cor}_{B}$	0,79871	0,81489	0,78453	0,87412	0,87104	0,82917
XB·SIM2$^{cor}_{B}$	0,78924	0,80719	0,79124	0,89486	0,89214	0,85318
X·SIM1$_{B}$	0,75918	0,79097	0,74672	0,85801	0,85578	0,80142
XB·SIM1$_{B}$	0,76785	0,7996	0,77028	0,87009	0,88456	0,83716
X·SIM2$_{B}$	0,76537	0,7954	0,75129	0,86606	0,86156	0,80812
XB·SIM2$_{B}$	0,77027	0,80119	0,7715	0,87996	0,88731	0,83933

20-Newsgroups

The D1 dataset consists of documents that originated from classes with conceptual similar content. In both experiments (A-B) the best NMI achieved from the SIM^{bursty}_{GVSM} matrix. In the first case, where topic bursts are placed in 30 windows (*Experiment B*), the best solution of spk-means, with the maximum Purity and F1 score, attained by $SIM2^{cor}_{B}$ matrix. In the second case, the same term-term matrix

with the representation XB·SIM2$_B^{cor}$ worked well too. According to the run that maximized the cohesion of the clusters, the XB·SIM1$_B$ representation was that with the the values for Purity and NMI.

Table 4.10: Term-Similarity matrices results on the D1 using spherical K-means

	T=30					
	avg. Purity	**avg. F1**	**avg. NMI**	**best Purity**	**best F1**	**best NMI**
X	0,51049	0,52423	0,45747	0,593	0,59331	0,51949
XB	0,56356	0,5711	0,49279	0,6304	0,61925	0,5422
X·SIM$_{GVSM}$	0,54	0,57378	0,53851	0,585	0,5834	0,55855
XB·SIM$_{GVSM}$	0,57524	0,59583	0,54924	0,6108	0,60108	0,56421
X·SIM$_{GVSM}^{bursty}$	0,60046	**0,61152**	**0,55873**	0,6334	0,61883	0,57131
XB·SIM$_{GVSM}^{bursty}$	**0,60052**	0,6113	0,5544	0,6366	0,63224	**0,57827**
X·SIM1$_B^{cor}$	0,5618	0,57371	0,51602	0,6292	0,62051	0,56491
XB·SIM1$_B^{cor}$	0,59568	0,60199	0,52376	0,6416	0,62912	0,55398
X·SIM2$_B^{cor}$	0,55679	0,56885	0,51165	0,6336	0,62469	0,56566
XB·SIM2$_B^{cor}$	0,59568	0,60156	0,52319	**0,6538**	**0,64227**	0,56037
X·SIM1$_B$	0,50416	0,51563	0,41663	0,5618	0,56348	0,46306
XB·SIM1$_B$	0,47065	0,4771	0,36805	0,5036	0,5013	0,39098
X·SIM2$_B$	0,50999	0,52051	0,42155	0,5638	0,56659	0,46509
XB·SIM2$_B$	0,47407	0,48012	0,37182	0,5048	0,50504	0,39342
			T=90			
X	0,51049	0,52423	0,45747	0,593	0,59331	0,51949
XB	0,65452	0,6588	0,59367	0,7482	0,75071	0,65354
X·SIM$_{GVSM}$	0,54	0,57378	0,53851	0,585	0,5834	0,55855
XB·SIM$_{GVSM}$	0,62389	0,6498	0,61264	0,6718	0,68874	0,64655
X·SIM$_{GVSM}^{bursty}$	0,63971	0,64973	0,58802	0,705	0,70356	0,62124
XB·SIM$_{GVSM}^{bursty}$	0,69174	0,70394	**0,6545**	0,7808	0,78423	**0,70592**
X·SIM1$_B^{cor}$	0,60687	0,62007	0,55751	0,6674	0,66506	0,60209
X·SIM1$_B^{cor}$	0,70382	0,71608	0,64943	0,7922	0,79322	0,70249
X·SIM2$_B^{cor}$	0,60721	0,6194	0,55597	0,6518	0,64495	0,59265
XB·SIM2$_B^{cor}$	**0,70721**	**0,71783**	0,65111	0,793	0,79409	0,70438
X·SIM1$_B$	0,60367	0,6331	0,53627	0,6998	0,70581	0,59384
XB·SIM1$_B$	0,68303	0,7137	0,63211	**0,8048**	**0,80336**	0,69592
X·SIM2$_B$	0,60865	0,63656	0,53845	0,7222	0,72293	0,6049
XB·SIM2$_B$	0,68389	0,71484	0,63255	0,7908	0,79362	0,6881

The D2 dataset was selected on purpose, as opposed to D1. In the extended version of the stream (*Experiment B*), the XB·SIM2$_B^{cor}$ and XB·SIM1$_B$ representations

dominated on the best and average runs respectively, with much better results than X·SIM2$_B^{cor}$ and X·SIM1$_B$. In comparison with the classical VSM (tfidf) the representation with bursty information (XB) in the short version of the stream (T=30) produced slightly better results than the previous case (T=90).

Table 4.11: Term-Similarity matrices results on the D2 using spherical K-means

	T=30					
	avg. Purity	**avg. F1**	**avg. NMI**	**best Purity**	**best F1**	**best NMI**
X	0,57099	0,58035	0,49093	0,686	0,67073	0,56485
XB	0,59983	0,60353	0,51628	0,6876	0,6806	0,58481
X·SIM$_{GVSM}$	0,60616	0,62771	0,55516	0,652	0,6777	0,58435
XB·SIM$_{GVSM}$	0,63691	**0,65713**	**0,58709**	0,66	0,68877	0,60598
X·SIM$_{GVSM}^{bursty}$	0,63297	0,6446	0,56822	0,6588	0,66877	0,58325
XB·SIM$_{GVSM}^{bursty}$	**0,64419**	0,65414	0,5761	0,6806	0,68242	0,59549
X·SIM1$_B^{cor}$	0,62474	0,63165	0,55026	0,7076	**0,70546**	**0,6075**
XB·SIM1$_B^{cor}$	0,64196	0,64793	0,55513	0,6916	0,6856	0,58247
X·SIM2$_B^{cor}$	0,62178	0,62906	0,54796	0,6986	0,69727	0,59996
XB·SIM2$_B^{cor}$	0,64223	0,6479	0,55593	**0,7096**	0,70136	0,60215
X·SIM1$_B$	0,51282	0,53095	0,43706	0,572	0,57295	0,47098
XB·SIM1$_B$	0,49357	0,51196	0,40729	0,5524	0,55505	0,43814
X·SIM2$_B$	0,51912	0,53693	0,44164	0,584	0,58396	0,48088
XB·SIM2$_B$	0,49595	0,51328	0,40817	0,5518	0,5518	0,44174
			T=90			
X	0,57099	0,58035	0,49093	0,686	0,67073	0,56485
XB	0,66339	0,66576	0,60469	0,7424	0,73902	0,66036
X·SIM$_{GVSM}$	0,60616	0,62771	0,55516	0,652	0,6777	0,58435
XB·SIM$_{GVSM}$	0,67288	0,69339	0,6483	0,726	0,75616	0,68227
X·SIM$_{GVSM}^{bursty}$	0,65327	0,66459	0,59129	0,6956	0,68812	0,60852
XB·SIM$_{GVSM}^{bursty}$	0,69089	0,70393	0,65602	0,7582	0,75575	0,69054
X·SIM1$_B^{cor}$	0,66211	0,67312	0,59251	0,733	0,71921	0,63036
XB·SIM1$_B^{cor}$	0,71492	0,72561	0,66898	0,781	0,77358	0,70545
X·SIM2$_B^{cor}$	0,66068	0,6713	0,5904	0,7332	0,7182	0,63024
XB·SIM2$_B^{cor}$	**0,71563**	**0,72611**	**0,66904**	0,781	0,77322	0,70519
X·SIM1$_B$	0,64078	0,67199	0,57726	0,7358	0,73954	0,63166
XB·SIM1$_B$	0,69354	0,72217	0,6511	**0,7988**	**0,79643**	**0,70785**
X·SIM2$_B$	0,64269	0,67262	0,57738	0,7348	0,73653	0,63
XB·SIM2$_B$	0,69311	0,72112	0,65084	0,7972	0,79387	**0,70792**

Maybe this is an explanation why the XB·SIM$_{GVSM}$ representation achieved the

highest average F1/NMI and the XB·SIM$^{bursty}_{GVSM}$ the highest Purity score. Finally as reported by the best solution, only the SIM^{bursty}_{GVSM} (F1,NMI) kept performing well again, while the max purity owned to the $SIM2^{cor}_{B}$.

We conclude that $SIM1^{cor}_{B}$ and $SIM2^{cor}_{B}$ generally, perform better than $SIM1_B$ and $SIM2_B$ matrices. The latter are not appropriate solutions because in many cases they failed to gain even the initial representation. All the proposed matrices depend on the burst detection step. Thus, in the matrices there are noisy co-bursty relations, proportional to the number of false detected bursty terms. We believe, that this noise is eliminated enough due to the correlation matrix COR. A more effective burst detection method would probably boost more the performance of all aforementioned matrices.

Shading more light to the second experiment of D3 dataset, it is noticeable that the X·SIM$_{GVSM}$ is the worst representation. Any additional weight that refers to terms co-occurrence wasn't not able to beat the XB representation. However, from the experiment-B in D4 and D5 datasets, as well as from results in D1 collection, where the stream entropy is low enough, the proposed similarity matrices seem to perform pretty well.

Our initial argument that the co-bursty terms could enhance text clustering performance is verified not only from the experiments on streams with low stream entropy, but also from the stable performance of XB·SIM$^{bursty}_{GVSM}$, which in most cases achieved better results than the XB representation.

- **Correlated Bursty Term Clustering**

The execution of the spherical K-means algorithm with initial centers the synthetic prototypes produced by our method in section 3.3.3 is compared with the average and best solution of 100 runs of spk-means with random initialization. In the first row of X and XB representations the average value of each metric is displayed, while in the second row the value of the best solution.

Our method depends on four parameters : *Pdocs, Pterms,* λ, β, *K, K'*. The first four are important for the synthetic prototype (*ConstructSP*) method. The λ and β were set to 0, while the *Pdocs* and *Pterms* to 0.5 and 0.4 respectively. We chose these values so that half documents of each cluster and less than 50 percentage of the words to be taken into consideration for the creation of the representatives. The K parameter was set to the numbers of the the underlying classes. Finally, K' was

set to *2K* and *3K*.

At this point, we should notice that for the evaluation of CBTC algorithm in D4 data set, we regarded all the bursty information that has been extracted from the Kleinberg's two-state automaton. Because the rest text streams (D1,D2,D3,D5) spanned only 30-31 time units, we observed after extensive experimental tests, that our method performed better when we adopted the period with the largest bursty weight of a feature as its bursty period over the whole duration T. Probably, this happens because we deal with more complicated streams (Figures 4.1, 4.2) contrary to the D5 dataset (Figure 4.3).

Table 4.12: Random Initializationation vs CBTC - results on the D5, D4 using spherical K-means

	Purity	**F1**	**NMI**		**Purity**	**F1**	**NMI**
			D5				
X	0,55694	0,56595	0,47404	XB	0,79388	0,79319	0,77183
	0,56343	0,5928	0,5156		**0,89552**	**0,89624**	**0,84097**
X-2K	**0,78358**	**0,81382**	**0,73753**	XB-2K	0,89179	0,89504	0,83692
X-3K	0,66791	0,69642	0,64545	X-3K	0,82463	0,80893	0,80116
			D4				
X	0,56383	0,51125	0,58706	XB	0,61354	0,55601	0,6412
	0,61826	0,5767	0,62348		0,66895	0,63935	0,6714
X-2K	0,68866	0,62495	**0,71205**	XB-2K	**0,70555**	**0,64479**	**0,72803**
X-3K	**0,69308**	**0,63978**	0,70533	X-3K	0,70032	0,6536	0,71984

From Table 4.12 we could see that in the D4 dataset our method gave much better results than the randomly initialized spk-means for both representations. In D5 dataset, things are a little different. First of all, we should remind that it is a small text collection, less imbalanced and with the lowest stream entropy. Moreover it is known that K-Means works better in datasets that have equally sized clusters. For these reasons, especially the first two, the clustering problem in the D5 dataset is much easier. For the X representation, the initialization with CBTC performs the best, contrary to the XB representation where the random initialization that maximizes the cohesion of the clusters is dominating. On the other the D3 (Table 4.13) was the only dataset, that our method didn't achieve better results than the best solution of X and XB representation. However, we observe that CBTC algorithm performs much better for all three metrics than the average of 100 spk-means runs.

Table 4.13: Random Initializationation vs CBTC - results on the D3 using spherical K-means

	Purity	**F1**	**NMI**
X	0,77058	0,77449	0,74477
	0,8711	**0,86384**	**0,82431**
X-2K	0,83787	0,85119	0,81308
X-3K	0,86103	0,85392	0,81263
XB	0,77915	0,78156	0,72218
	0,87412	**0,87378**	**0,78662**
XB-2K	0,81873	0,81737	0,75674
XB-3K	0,84693	0,8362	0,77201

The superiority of our method in all metrics against the average value after running the spherical K-means 100 times, was expected as the representatives of the clusters are produced from the combination of documents that are close to the medoid of each class. The crucial point in our method is that the most important documents for an event are published when this event is bursty.

Table 4.14: Random Initializationation vs CBTC - results on the D1,D2 using spherical K-means

	Purity	**F1**	**NMI**		**Purity**	**F1**	**NMI**
			D1				
	Purity	F1	NMI		Purity	F1	NMI
X	0,51049	0,52423	0,45747	XB	0,58167	0,5905	0,53521
	0,593	0,59331	0,51949		**0,639**	**0,64696**	0,5645
X-2K	0,587	0,59707	0,55897	XB-2K	0,623	0,63372	**0,60387**
XB-3K	**0,649**	**0,6629**	**0,61582**	XB-3K	0,631	0,64531	0,59339
			D2				
X	0,57099	0,58035	0,49093	XB	0,59065	0,60314	0,50202
	0,686	0,67073	0,56485		0,653	0,66345	0,54625
X-2K	**0,694**	**0,72323**	**0,62019**	XB-2K	**0,679**	**0,70549**	**0,58885**
X-3K	0,598	0,60933	0,55441	XB-3K	0,587	0,60784	0,53174

As it is observed from the results presented in Table 4.14 for the D2 data collection, the superiority of our method is clearly. Moreover it performs better for the X representation for data set D1. Regarding the XB representation, the CBTC initialization method gave the highest NMI measure.

Figure 4.1: Text stream distribution of D1 and D2.

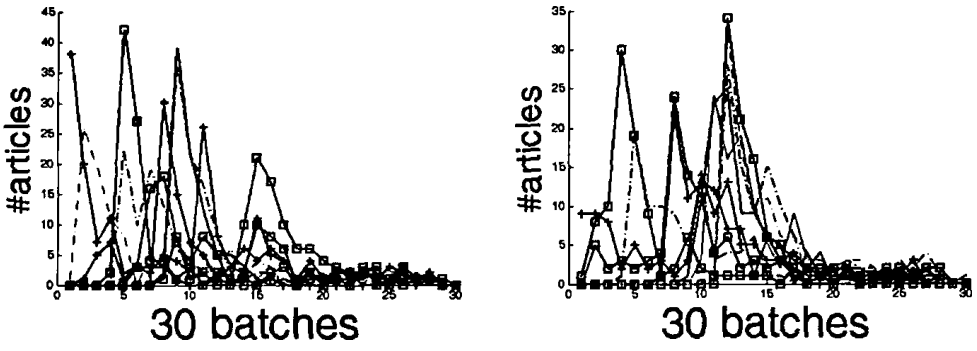


Figure 4.2: Text stream distribution of D3.

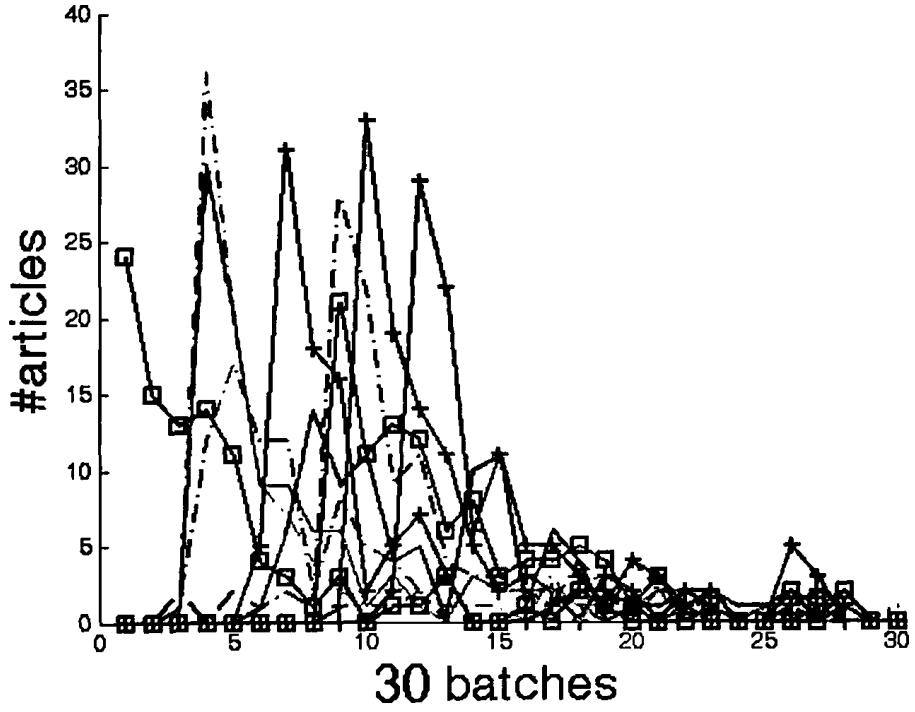
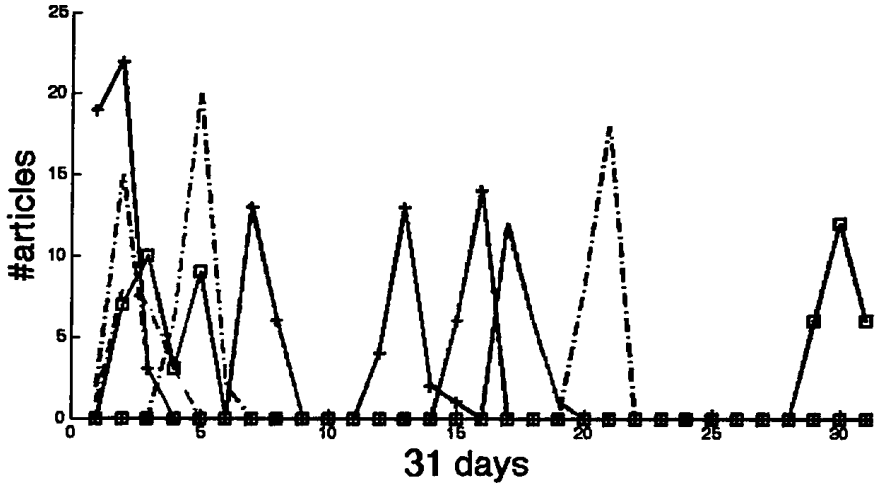


Figure 4.3: Text Stream distribution of D5.



CHAPTER 5

CONCLUSIONS AND FUTURE WORK

5.1 Conclusions

5.2 Future Work

5.1 Conclusions

In this thesis, we studied the problem of text stream clustering, exploiting the bursty information. In order to discover the busty features we used the Kleineberg's two-state automaton. We have, also, presented several term-similarity matrices, that are constructed from bursty features and were evaluated against a bursty-feature and GVSM representations using the spherical K-means algorithm.

Furthermore, we introduced the CBTC approach, which is used for the initialization of spherical K-means when this is applied to text stream collections where the notion of burst is crucial. In order to find the initial representatives, we construct a graph using the bursty terms and then we partition the graph using Spectral Clustering. In the next step, we assign documents to each cluster of terms and we compute representatives of each cluster using the synthetic prototype method. At the end we merge the most similar clusters.

The experiments were conducted on several benchmark datasets. From the clustering results, we observed that in most cases the CBTC method performs much better than random initialization. Moreover, because of the lack of timestamps for 20NGs and Reuters-21578 datasets we used an artificial method to create topic bursts. Moreover, the SIM^{bursty}_{GVSM} term similarity matrix seems to be superior in most cases to the GVSM and B-VSM representations. Finally, in text streams where events

overlap in few time slots, the XB·SIM2$_B^{cor}$ tends to perform better than GVSM and SIM_{GVSM}^{bursty}.

5.2 Future Work

Our plans for future work are to test the CBTC algorithm against other methods that are used for the initialization of the spherical K-means like K-means++ [3]. We could also investigate further ways to improve our algorithm by building a graph with bursty and non-bursty features or testing other methods in order to partition the graph into groups of features. It is also interesting to assess the behavior of our algorithm on short and long-running events.

Regarding the term correlation matrices, we could study ways to calculate the corresponding semantic kernels. Moreover, the proposed methods should be evaluated in the future using other clustering algorithms. We have, also, the intuition, that if graph clustering is successfully we could extract groups of terms from the bursty keywords graph, that would be able to construct several term similarity matrices corresponding to different topics of a text stream. Finally, it would also be interesting to check the performance of our method on data derived from social platforms like Twitter.

BIBLIOGRAPHY

[1] J. Allan, J. G. Carbonell, G. Doddington, J. Yamron, and Y. Yang. Topic detection and tracking pilot study final report. 1998.

[2] J. Allan, R. Papka, and V. Lavrenko. On-line new event detection and tracking. In *Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 37–45. ACM, 1998.

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

[4] N. Bansal and N. Koudas. Blogscope: spatio-temporal analysis of the blogosphere. In *Proceedings of the 16th international conference on World Wide Web*, pages 1269–1270. ACM, 2007.

[5] H. Billhardt, D. Borrajo, and V. Maojo. A context vector model for information retrieval. *Journal of the American Society for Information Science and Technology*, 53(3):236–249, 2002.

[6] D. Bogard and W. Tiederman. Burst detection with single-point velocity measurements. *Journal of Fluid Mechanics*, 162:389–413, 1986.

[7] N. BuzzMetric. Blogpulse, 2008.

[8] F. Can, S. Kocberber, O. Baglioglu, S. Kardas, H. C. Ocalan, and E. Uyar. Bilkent news portal: A personalizable system with new event detection and tracking capabilities. In *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 885–885. ACM, 2008.

[9] C. C. Chen, M. C. Chen, and M.-S. Chen. An adaptive threshold framework for event detection using hmm-based life profiles. *ACM Transactions on Information Systems (TOIS)*, 27(2):9, 2009.

[10] C. C. Chen, Y.-T. Chen, Y. Sun, and M. C. Chen. Life cycle modeling of news events using aging theory. In *Machine Learning: ECML 2003*, pages 47–59. Springer, 2003.

[11] W. Chen, C. Chen, L.-j. Zhang, C. Wang, and J.-j. Bu. Online detection of bursty events and their evolution in news streams. *Journal of Zhejiang University SCIENCE C*, 11(5):340–355, 2010.

[12] X. Cheng, D. Miao, C. Wang, and L. Cao. Coupled term-term relation analysis for document clustering. In *Neural Networks (IJCNN), The 2013 International Joint Conference on*, pages 1–8. IEEE, 2013.

[13] G. Du, J. Guo, W. Xu, and Z. Yang. Maximizing the reliability of two-state automaton for burst feature detection in news streams. In *Progress in Informatics and Computing (PIC), 2010 IEEE International Conference on*, volume 1, pages 229–233. IEEE, 2010.

[14] A. K. Farahat and M. S. Kamel. Statistical semantics for enhancing document clustering. *Knowledge and information systems*, 28(2):365–393, 2011.

[15] U. Fayyad, G. Piatetsky-Shapiro, and P. Smyth. From data mining to knowledge discovery in databases. *AI magazine*, 17(3):37, 1996.

[16] D. Fisher, A. Hoff, G. Robertson, and M. Hurst. Narratives: A visualization to track narrative events as they develop. In *Visual Analytics Science and Technology, 2008. VAST'08. IEEE Symposium on*, pages 115–122. IEEE, 2008.

[17] T. Fujiki, T. Nanno, Y. Suzuki, and M. Okumura. Identification of bursts in a document stream. In *First International Workshop on Knowledge Discovery in Data Streams (in conjunction with ECML/PKDD 2004)*, pages 55–64. Citeseer, 2004.

[18] G. P. C. Fung, J. X. Yu, P. S. Yu, and H. Lu. Parameter free bursty events detection in text streams. In *Proceedings of the 31st international conference on Very large data bases*, pages 181–192. VLDB Endowment, 2005.

[19] S. Havre, B. Hetzler, and L. Nowell. Themeriver: Visualizing theme changes over time. In *Information Visualization, 2000. InfoVis 2000. IEEE Symposium on*, pages 115–123. IEEE, 2000.

[20] Q. He, K. Chang, and E.-P. Lim. Analyzing feature trajectories for event detection. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 207–214. ACM, 2007.

[21] Q. He, K. Chang, and E.-P. Lim. Using burstiness to improve clustering of topics in news streams. In *Data Mining, 2007. ICDM 2007. Seventh IEEE International Conference on*, pages 493–498. IEEE, 2007.

[22] Q. He, K. Chang, E.-P. Lim, and A. Banerjee. Keep it simple with time: A reexamination of probabilistic topic detection models. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 32(10):1795–1808, 2010.

[23] Q. He, K. Chang, E.-P. Lim, and J. Zhang. Bursty feature representation for clustering text streams. In *SDM*, pages 491–496. SIAM, 2007.

[24] A. Hoonlor, B. K. Szymanski, M. J. Zaki, and V. Chaoji. Document clustering with bursty information. *Computing and Informatics*, 31(6+):1533–1555, 2013.

[25] A. Kalogeratos and A. Likas. Document clustering using synthetic cluster prototypes. *Data & Knowledge Engineering*, 70(3):284–306, 2011.

[26] A. Kalogeratos and A. Likas. Text document clustering using global term context vectors. *Knowledge and information systems*, 31(3):455–474, 2012.

[27] A. Kaltenbrunner and D. Laniado. There is no deadline: time evolution of wikipedia discussions. In *Proceedings of the Eighth Annual International Symposium on Wikis and Open Collaboration*, page 6. ACM, 2012.

[28] M. Karkali, F. Rousseau, A. Ntoulas, and M. Vazirgiannis. Efficient online novelty detection in news streams. In *Web Information Systems Engineering–WISE 2013*, pages 57–71. Springer, 2013.

[29] J. Kleinberg. Bursty and hierarchical structure in streams. *Data Mining and Knowledge Discovery*, 7(4):373–397, 2003.

[30] G. Kumaran and J. Allan. Text classification and named entities for new event detection. In *Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 297–304. ACM, 2004.

[31] W. Lam, H. Meng, K. Wong, and J. Yen. Using contextual analysis for news event detection. *International Journal of Intelligent Systems*, 16(4):525–546, 2001.

[32] T. Lappas, B. Arai, M. Platakis, D. Kotsakos, and D. Gunopulos. On burstiness-aware search for document sequences. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 477–486. ACM, 2009.

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

[34] D. Luo, J. Yang, M. Krstajic, J. Fan, W. Ribarsky, and D. Keim. Eventriver: An event-based visual analytics approach to exploring large text collections with a temporal focus. *IEEE Transactions on Visualization and Computer Graphics*, 18(1):93–105, 2012.

[35] G. Luo, C. Tang, and P. S. Yu. Resource-adaptive real-time new event detection. In *Proceedings of the 2007 ACM SIGMOD international conference on Management of data*, pages 497–508. ACM, 2007.

[36] J. Makkonen, H. Ahonen-Myka, and M. Salmenkivi. Simple semantics in topic detection and tracking. *Information Retrieval*, 7(3-4):347–368, 2004.

[37] Y. Mitsuishi, V. Nováček, and P.-Y. Vandenbussche. A method for building burst-annotated co-occurrence networks for analysing trends in textual data.

[38] A. Y. Ng, M. I. Jordan, Y. Weiss, et al. On spectral clustering: Analysis and an algorithm. *Advances in neural information processing systems*, 2:849–856, 2002.

[39] Y. Ohsawa, N. E. Benson, and M. Yachida. Keygraph: Automatic indexing by co-occurrence graph based on building construction metaphor. In *Research and Technology Advances in Digital Libraries, 1998. ADL 98. Proceedings. IEEE International Forum on*, pages 12–18. IEEE, 1998.

[40] M. Platakis, D. Kotsakos, and D. Gunopulos. Discovering hot topics in the blogosphere. In *Proc. of the Panhellenic Scientific Student Conference on Informatics, Related Technologies and Applications EUREKA*, pages 122–132, 2008.

[41] M. F. Porter. An algorithm for suffix stripping. *Program*, 14(3):130–137, 1980.

[42] J. Ratkiewicz, M. Conover, M. Meiss, B. Gonçalves, S. Patil, A. Flammini, and F. Menczer. Detecting and tracking the spread of astroturf memes in microblog streams. *arXiv preprint arXiv:1011.3768*, 2010.

[43] H. Sayyadi, M. Hurst, and A. Maykov. Event detection and tracking in social streams. In *ICWSM*, 2009.

[44] M. Vlachos, C. Meek, Z. Vagena, and D. Gunopulos. Identifying similarities, periodicities and bursts for online search queries. In *Proceedings of the 2004 ACM SIGMOD international conference on Management of data*, pages 131–142. ACM, 2004.

[45] A. Vural. *On-line new event detection and clustering using the concepts of the cover coefficient-based clustering methodology*. PhD thesis, Department of Computer Engineering and the Institute of Engineering and Science of Bilkent Univ., 2002.

[46] S. M. Wong, W. Ziarko, and P. C. Wong. Generalized vector spaces model in information retrieval. In *Proceedings of the 8th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 18–25. ACM, 1985.

[47] R. Xie, F. Zhu, H. Ma, W. Xie, and C. Lin. Clear: A real-time online observatory for bursty and viral events. *Proceedings of the VLDB Endowment*, 7(13), 2014.

[48] W. Xie, F. Zhu, J. Jiang, E.-P. Lim, and K. Wang. Topicsketch: Real-time bursty topic detection from twitter. In *Data Mining (ICDM), 2013 IEEE 13th International Conference on*, pages 837–846. IEEE, 2013.

[49] Y. Yang, J. Zhang, J. Carbonell, and C. Jin. Topic-conditioned novelty detection. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 688–693. ACM, 2002.

[50] J. Yao, B. Cui, Y. Huang, and Y. Zhou. Bursty event detection from collaborative tags. *World Wide Web*, 15(2):171–195, 2012.

[51] J. Yi. Detecting buzz from time-sequenced document streams. In *e-Technology, e-Commerce and e-Service, 2005. EEE'05. Proceedings. The 2005 IEEE International Conference on*, pages 347–352. IEEE, 2005.

[52] Z. Yuan, Y. Jia, and S. Yang. Online burst detection over high speed short text streams. In *Computational Science–ICCS 2007*, pages 717–725. Springer, 2007.

[53] D. Zeimpekis and E. Gallopoulos. Text to matrix generator user's guide. *Department of Computer Engineering and Informatics, University of Patras, Greece*, 2007.

[54] K. Zhang, J. Li, G. Wu, and K. Wang. A new event detection model based on term reweighting. *Journal of Software*, 19(4):817–828, 2008.

[55] K. Zhang, J. Zi, and L. G. Wu. New event detection based on indexing-tree and named entity. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 215–222. ACM, 2007.

[56] X. Zhang and D. Shasha. Better burst detection. In *Data Engineering, 2006. ICDE'06. Proceedings of the 22nd International Conference on*, pages 146–146. IEEE, 2006.

[57] Q. Zhao and P. Mitra. Event detection and visualization for social text streams. In *ICWSM*, 2007.

[58] W. X. Zhao, R. Chen, K. Fan, H. Yan, and X. Li. A novel burst-based text representation model for scalable event detection. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Short Papers-Volume 2*, pages 43–47. Association for Computational Linguistics, 2012.

[59] Y. Zhu and D. Shasha. Efficient elastic burst detection in data streams. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 336–345. ACM, 2003.

Short Vita

Panagiotis Zagorisios was born in Ioannina in 1988. He was admitted at Computer Science Department of the University of Ioannina in 2006, and he received his BSc degree in Computer Science in 2011. Then, he attended the postgraduate program in University of Ioannina, Computer Science and Engineering Department. From September to December 2013, he did an Erasmus internship in the Social Media research group of Fundacio Barcelona Media. His academic interests are in the field of data mining and machine learning.