Improving Text Stream Clustering using Term Burstiness and Co-burstiness

Argyris Kalogeratos
CMLA, ENS Cachan, CNRS
Université Paris-Saclay,
France
kalogeratos@cmla.ens-cachan.fr

Panagiotis Zagorisios
Dept. of Computer Science
and Engineering,
University of Ioannina, Greece
pzagoris@cs.uoi.gr

Aristidis Likas
Dept. of Computer Science
and Engineering,
University of Ioannina, Greece
arly@cs.uoi.gr

ABSTRACT

In text streams, documents appear over time and their timestamps can be used to improve typical approaches for text representation and clustering. A way to exploit temporal information is through the detection of *bursty terms* in such streams, i.e. terms that appear in many documents during short time period. Research efforts so far have shown that utilizing the burst information in the text representation can improve the performance of text clustering algorithms. However, most attempts take into account the bursty terms individually, without investigating the relation between them. In this work, we take advantage of the fact that most of the important documents of a topic are published during the period in which the 'main' topic terms are bursty. Therefore, we focus on both *term burstiness* and *co-burstiness* by determining groups of terms that are simultaneously bursty at a time period and also *co-occurring* in the same documents. Next, the documents that contain co-bursty terms from those groups, are considered as important for the respective topics and are used to construct robust *synthetic prototypes* following an agglomerative process. These prototypes are finally used to initialize in a deterministic fashion the spherical k-means clustering algorithm. Experimental results validate empirically the quality of the solutions provided by the proposed approach which seems to efficiently overcome the initialization problem of spherical k-means.

CCS Concepts

•Information systems → Clustering; Data stream mining; Document representation;

1. INTRODUCTION

Clustering is one of the corner-stone unsupervised learning problems for the data mining and machine learning field [6]. In a clustering task the aim is to discover groups of similar objects and separate them from other dissimilar groups. Text clustering, on static collections or text streams, is a challenging problem that receives increasing attention in recent years due to its apparent applications on the textual content that dominates the communication on the Web. Sources such as social networks (e.g., `Twitter` and `Facebook`) or newswire sites and blogs, produce continuous data streams and in huge quantities.

This work investigates the problem of *text stream clustering* which is closely related to the *topic detection* problem [2]. The goal is the identification of document clusters in text streams, where each document is marked by a *timestamp* denoting the time at which it appears in the stream. Each output cluster should contain documents that refer to the same real-life *topic*, which is considered as a content abstraction since it may include multiple semantically similar events that occur over time and produce temporal increase of interest in the topic.

Conventional clustering neglects the timestamp information of documents, hence the question regarding what kind of improvement can be achieved to existing approaches by taking into account this temporal information is straightforward. The most common way to do this, is through the detection and exploitation of the *bursty terms*, which are terms appearing in many documents during a short time period in the stream. In particular, this involves the reweighting of the feature vectors that represent the documents in the vector space model (VSM). The basic idea is the following: if a term is bursty during a specific time interval, its contribution (term weight) should be increased in all documents that contain the term and are published during that term's burst interval. In [8] a VSM extension with *bursty term representation* of documents is proposed that considers a bursty term interval if an unusual large number of documents containing that term appears in the stream. In the same spirit, other VSM extensions have also been proposed [9, 25]. In these approaches, the *spherical k-means* (sp$k$-means) is employed to cluster the enriched document vectors representations. Note, however, that sp$k$-means inherits the sensitivity to its initialization from the k-means family, and hence may get stuck into poor local maxima of the objective function.

In this work, we elaborate on the terms' temporal information and suggest a novel way to use it for improving text stream clustering. We attempt to exploit both *term burstiness* and co-burstiness over time. The burstiness is incorporated using existing related approaches that, as mentioned, increase the importance of bursty terms individually for certain documents. As for the *co-burstiness*, we seek for groups of terms that are *co-bursty* at the same time period and also *co-occur* in the same documents. We then consider that

SETN '16, May 18 - 20, 2016, Thessaloniki, Greece

ISBN 978-1-4503-3734-2/16/05…$15.00

DOI: http://dx.doi.org/10.1145/2903220.2903229

the documents containing bursty terms are representatives for their topic. Specifically, we use such characteristic documents, to build cluster representatives by combining the concepts of *synthetic prototypes* [11] and agglomerative clustering. Finally, the obtained cluster representatives are used to initialize the sp$k$-means algorithm that provides the final text stream clustering solution.

The rest of the paper is organized as follows. Sec. 2 refers to the term burst detection problem and ways to integrate the burst information to construct improved vector representations for documents. It also provides a description of the idea of synthetic prototypes. Sec. 3 presents the proposed method, called *Correlated Bursty Term Clustering* (CBTC). In Sec. 4, comparative experimental results are presented, and finally, Sec. 5 provides our conclusions and directions for further research.

2. BACKGROUND AND RELATED WORK

2.1 Text Representation and Spherical k-means

Representation. As text stream S, we consider a set of documents D that appear on a timeline which is split in T non-overlapping time windows:

$$S = [s_1, ..., s_T], \tag{1}$$

where if $|D| = N$ is the total number of documents of the stream, then $s_t \in D$ represents a batch of documents published at the t-th time window. The documents of a batch are all attributed with one timestamp among $\{1, ..., T\}$.

The most widely-used text representation in the field of text mining is the vector space model (VSM). Under this model, a document is represented by a vector of weights corresponding to text features of a vocabulary $\mathcal{V}$ of size $V = |\mathcal{V}|$. The weight of each term quantifies its relevance to the document content. As features, according to the popular bag-of-words (BOW) approach, distinct terms are considered, and finally, a document $d_i \in \mathbb{R}^V$ is represented as a V-dimensional vector:

$$d_i = [d_{i1}, ..., d_{iV}]^\top = [tf_{i1} \cdot idf_1, ..., tf_{iV} \cdot idf_V]^\top, \tag{2}$$

where tf_j is the frequency, and $idf_j = log\frac{N}{df_t}$ is the inverse document frequency for term f_j that is higher for more discriminative terms. The common characteristics between two documents are measured using the cosine similarity that computes the cosine of the angle between two vectors:

$$sim_{cos}(d_i, d_j) = \frac{d_i^\top d_j}{||d_i||_2 \, ||d_j||_2} \in [0, 1]. \tag{3}$$

We denote the matrix that stores the vector representations of the documents, as stacked rows, as $\mathrm{X} \in \mathbb{R}^{N \times V}$.

Clustering. Spherical k-means (sp$k$-means) is a k-means variant that utilizes the *cosine similarity* as a proximity measure, which ignores the vector magnitude, and the clustering *cohesion* as maximized objective function. The optimal representative for a cluster is its L_2-normalized centroid, denoted as r_j, and the overall clustering cohesion of a partition C is given by:

$$Cohesion(C) = \sum_{j=1}^{k} \sum_{d_i \in c_j} r_j^\top d_i. \tag{4}$$

Given a dataset and the number of desired clusters k, this algorithm initially selects k cluster representatives (centroids), usually by picking documents at random. Then, it iterates until convergence between two steps: one that assigns each document to the cluster with the closest representative, and a second one that updates the cluster representatives by considering the updated cluster assignments. This variant inherits the simplicity of its family and achieves fast monotonic convergence. However, it suffers from the same weaknesses, e.g. it converges to a local optimum of the objective function which largely depends on the initialization.

2.2 Synthetic Cluster Prototypes

The concept of *synthetic cluster prototypes* originates from [11], where the k-synthetic prototypes (k-sp) clustering algorithm was proposed. In fact, k-sp is a sp$k$-means variant that challenges the use of centroids as cluster prototypes in favor of the synthetic prototypes, although the former are optimal w.r.t. to Eq. 4 or the standard mean squared error function. The key-idea that justifies this choice is that the usual randomized initialization approaches provide inhomogeneous clusters to sp$k$-means. If such impure clusters are represented *optimally* by the centroids, then due to the high-dimensional and sparse space, the inhomogeneity will be retained by the cluster representation and the procedure will converge to a particularly bad solution. Instead, more robust and selective prototypes should be considered during the early clustering iterations, and the centroids should be used later as a final refinement step on already quite homogeneous clusters.

On a technical level, a synthetic prototype of a cluster is constructed as follows. Starting from the *medoid* object m, a percentage of the p_{docs} documents in the cluster that are nearest to m are selected and their centroid is set as the synthetic representative. In addition, a feature filtering is optionally applied on that representative, where only the non-zero entries of its vector that correspond (in sum) to a percentage of p_{terms} of the initial vector magnitude is retained. Intuitively, the synthetic prototype favors the representation of documents belonging to the same class to that of the medoid object. Therefore, if there are documents of two or more classes in a cluster, then its synthetic prototype will tend to represent the dominant class and suppress the representation of documents farther from the medoid which may belong to minor classes mixed in the cluster.

Note that it can be decided for the construction of synthetic prototypes to be more refined, by applying an incremental approach. In this case, instead of selecting directly the set corresponding to the p_{docs} from the cluster members, this set grows gradually around the cluster medoid [11]. However, this improvement comes with the cost of additional parameters and was not used in this work.

2.3 Burst Detection

The rapid increase of a term's frequency of appearance, defines a *term burst* in the text stream. Thus, a term is considered as bursty when its frequency is encountered at an unusual high rate. The identification of bursts is known as *burst detection* procedure. Among the proposed methodologies [23, 26, 4], a widely-used one is the two-state automaton proposed by Kleinberg [13]. It assumes that the documents arrive in T consecutive batches, as in Eq. 1. Let $|s_t|$ express the total number of documents in the batch arriving at the t-th time window (i.e. $\sum_t^T |s_t| = N$), and $|s_{tj}|$ be the number of documents of that batch which contain the term f_j (i.e.

$\sum_t^T |s_{tj}| = |D_j|$). The automaton has two states, one with low emission rate $p_0 = |D_j|/T$, which is actually an expectation under the uniform distribution hypothesis, and another one with higher rate $p_1 = \alpha \cdot p_0$, where $\alpha > 1$ is a parameter and makes the detection more selective.

In this model, each (s_t, s_{tj}) is considered to be an output symbol that is produced probabilistically according to the internal state of a hidden Markov model. Let $q_t \in [0, 1]$ the state of the automaton at time t, and a conversion cost for state transition, which is $\gamma \log(T)$ when $q_t < q_{t+1}$, and zero otherwise. In other words, the model considers a cost for passing in more bursty states, while the opposite is cost-free. Consecutive appearances of state 1 (bursty state) are considered as bursty moments of the term. Provided a sequence of (s_t, s_{tj}) tuples, the goal of burst detection is to find the optimal state sequence $(q_1, ..., q_T)$ that minimizes the cost for each q_t to be at the i-th state ($i \in [0, 1]$) of the automaton:

$$\sigma(i, |s_{tj}|, |s_t|) = -ln \left[\binom{|s_t|}{|s_{tj}|} p_i^{|s_{tj}|} (1 - |p_i|)^{|s_t| - |s_{tj}|} \right]. \quad (5)$$

The state transition sequence that minimizes the above cost function is derived using the Viterbi approach for dynamic programming. The *length* of a burst is the time distance between the first and the last consecutive bursty moment, i.e. $t_2 - t_1 + 1$. As for the *burst weight* of a term f_j which occurs in a specific time interval $[t_1, t_2]$, that is:

$$w_j^{[t_1, t_2]} = \sum_{t=t_1}^{t_2} (\sigma(0, |s_{tj}|, |s_t|) - \sigma(1, |s_{tj}|, |s_t|)). \quad (6)$$

The weight is non-negative and expresses the reduction in the cost incurred by the use of state p_1 instead of p_0. Note that there is no efficient automatic way provided for tuning the two parameters α and γ, which should be set manually.

To simplify our notations, we denote as $w_j^{(t)}$ the burst weight of term f_j for the time interval that includes a point in time, t (also as $\overline{w}_j^{(t)} = w_j^{(t)} / \max_{i,t} w_i^{(t)}$, the normalized burst weight). Also, we denote the set of bursty terms as $\mathcal{B}$ and we define as τ_j the bursty time period of term f_j, i.e. the union of its bursty time intervals. Apparently it holds: $w_j^{(t)} > 0, \forall t \in \tau_j$, and also $f_j \in \mathcal{B}$ iff $\tau_j \neq \emptyset$.

2.4 Bursty Document Representations

In [9] the bursty feature representation has been introduced that extends the typical VSM-BOW approach with the inclusion of a binary weight for each term. The basic idea is that the weight of a bursty term should increase for the vectors of documents appearing during that term's burst period. Let $d_i^{(t)}$ a document that appears at time t in the stream, then its bursty vector representation (B-VSM) is:

$$d_{ij}^{(t)} = \begin{cases} \mathbb{1}\{tf_{ij} > 0\} + \delta w_j^{(t)}, & \text{if } t \in \tau_j; \\ \mathbb{1}\{tf_{ij} > 0\}, & \text{otherwise,} \end{cases} \quad (7)$$

where $\mathbb{1}\{\cdot\}$ is the indicator function, and $\delta > 0$ is a constant.

In the follow-up work [8], the performance of the previous idea is tested using the TFIDF weighting scheme by considering both the normalized $\overline{w}_j$ and unormalized burst weight w_j, ending up with five B-VSM variants. The first two variants modify the document vectors while keeping them in the original V-dimensional space:

$$\textbf{(SAB)}: \; d_{ij}^{(t)} = \begin{cases} tfidf_{ij} + \overline{w}_j^{(t)}, & \text{if } f_j \in \mathcal{B}; \\ tfidf_{ij}, & \text{otherwise,} \end{cases} \quad (8)$$

$$\textbf{(SMB)}: \; d_{ij}^{(t)} = \begin{cases} tfidf_{ij} \cdot w_j^{(t)}, & \text{if } f_j \in \mathcal{B}; \\ tfidf_{ij}, & \text{otherwise.} \end{cases} \quad (9)$$

However, the rest B-VSM models use only the bursty feature space $\mathcal{B}$, i.e. $d_i = [d_{i1}, ..., d_{iB}]^\top$, to represent a document:

$$\textbf{(BAB)}: \; d_{ij}^{(t)} = tfidf_{ij} + \overline{w}_j^{(t)}, \quad (10)$$

$$\textbf{(BMB)}: \; d_{ij}^{(t)} = tfidf_{ij} \cdot w_j^{(t)}, \quad (11)$$

$$\textbf{(BT)}: \; d_{ij}^{(t)} = tfidf_{ij}. \quad (12)$$

The experimental results in [8] revealed that the B-VSM could improve significantly the clustering result in terms of recall and precision measures.

The work in [25] follows a similar concept with [8, 9], aiming to retrospectively mine events from text streams by means of clustering. Its major contribution is the Burst-VSM representation that filters out all non-bursty terms and represents the documents in the B-dimensional bursty feature space using the original term weights:

$$\textbf{(Burst-VSM)}: \; d_{ij}^{(t)} = \begin{cases} tfidf_{ij}, & \text{if } t \in \tau_j; \\ 0, & \text{otherwise.} \end{cases} \quad (13)$$

In that work, the bursty terms were inferred using a variation of Kleinberg's two-state automaton on a sliding window.

Finally, a closely related method to the aforementioned works is [10]. Assuming that traditional VSM 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 [14]. Their 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 term co-occurrence.

Employed B-VSM representation. In our method which is presented next, we used the Kleinberg's two-state automaton to detect the set of bursty terms $\mathcal{B}$ that returns the burst weight w_j and burst time period τ_j (union of time intervals) of each term. Based on the results of comparative experiments, we concluded that the following burst-reweighing scheme works better for the text streams we used:

$$d_{ij}^{(t)} = \begin{cases} tfidf_{ij} \cdot w_j^{(t)}, & \text{if } t \in \tau_j; \\ tfidf_{ij}, & \text{otherwise,} \end{cases} \quad (14)$$

After applying the above transformation to the initial corpus representation X, we obtain the representation XB that takes into account term burstiness.

3. CBTC: CORRELATED BURSTY TERM CLUSTERING

The burst period of a term implies the time intervals during which the events associated with it are *trendy* and a lot of related documents are being published around those points in time. All these events can be part of the same topic. Furthermore, and according to Kleinberg's burst detection

Algorithm 1 Initialization of sp$k$-means with the CBTC.

function CBTC ($\hat{X}$, p_{docs}, p_{terms}, k, k', A)

input : $\hat{X}$ is the document matrix with row vectors, p_{docs}, p_{terms} are parameters for the synthetic prototype construction, k and k' the starting and desired number of clusters ($k' \geq k$), and A the bursty term correlation matrix

output : $R = \{r_1, ..., r_k\}$ the set of final cluster prototypes, $C = \{c_1, ..., c_k\}$ the sets of documents assigned to each cluster

1: $C^{(f)} \leftarrow$ SegmentTermGraph (A, k') // see Alg. 2
2: $\{SP, C^{(b)}\} \leftarrow$ ConstructBurstySP ($C^{(f)}$, $\hat{X}$, p_{docs}, p_{terms}) // see Alg. 3
3: $\{SP\} \leftarrow$ MergeClusters ($C^{(b)}$, SP, k, p_{docs}, p_{terms}) // see Alg. 4
4: $\{R, C\} \leftarrow$ spkmeans (SP, $\hat{X}$, k) // see Sec. 2.1
5: **return** (R, C)

method, the larger the number of documents containing the candidate term, the larger the probability for detecting it as bursty. Thus, bursty terms could indicate the most important documents for the topic. We regard as important a document that is close to the centroid (or medoid) of documents that belong to the same topic (ground truth cluster). Such documents are in essence good quality initial centroids that could improve the performance of algorithms of the k-means family which depend on the selection of the initialization.

Our contribution to the discussed text stream clustering problem is packed in a method called *Correlated Bursty Term Clustering* (CBTC) for the selection of the k initial centers, from which, spherical k-means (sp$k$-means) may then search for a local optimum. The proposed method is deterministic and, thus, not sensitive to random initialization on which sp$k$-means relies.

The CBTC approach stems from the philosophy of feature-based methods that define clusters by grouping the vocabulary terms. Suppose we wish to cluster the documents of a text stream into k clusters, each one being a topic for which one or more semantically similar events create bursts of interest over time. The pseudocode of our method is presented in Alg. 1 where its main steps are: i) Initially, we create k' groups of bursty terms ($k' > k$, e.g. $k' = 2k$) by segmenting the so-called *bursty term correlation graph* using the spectral clustering algorithm [16] (Sec. 3.1). ii) Then, we compute the k' representatives of the document sets related with each of the term clusters, in the original V-dimensional space, using the *synthetic prototypes* procedure (Sec. 3.2). The representatives are computed from the set of documents ($Docs_B$) that contain the bursty terms and are published during their burst intervals. iii) Next, we reduce the number of clusters from k' to k using an *agglomerative* approach that merges iteratively the most similar clusters according to the cosine similarity of their synthetic prototypes (Sec. 3.3). iv) Finally, the synthetic prototypes are used to initialize sp$k$-means and get the final clusters and their centroids.

3.1 Bursty Term Correlation Graph

In order to cluster the bursty terms, we should first define the similarity between them. To this end, we create a *term correlation graph* based on the co-occurrences of bursty terms. In literature, there are several works following this concept, either for event detection or topic summarization. In [17] an event detection algorithm was proposed using a keyword co-occurrence graph. In [19] a graph is created, from noun phrases and name entities, and then a community detection algorithm was applied to detect events. However, none of these works considers term burstiness that we focus herein. Furthermore, in [15] a network is built for a specific trending topic, where nodes denote bursty and non-bursty terms and each edge denotes a co-occurrence relation. In addition, in [21] the problem of bursty event tagging was studied, where an event is described by a set of tags. The authors observed that tags from various Web sources reflect the users' interests over time, thus by applying graph segmentation techniques one could detect bursty events. Finally, a more sophisticated method for event detection from social text streams (e.g. blogs and exchanges of emails) is presented in [24].

Our *bursty term correlation graph* is represented by an adjacency matrix $A \in \mathbb{R}^{B \times B}$, where nodes correspond to bursty terms while each edge a_{ij} between two nodes, f_i and f_j, is added under two conditions: i) *Co-burstiness*: their burst time periods, τ_i and τ_j, do overlap, i.e. $\tau_i \cap \tau_j \neq \emptyset$. ii) *Co-occurrence*: during the overlapping period, the terms f_i and f_j do co-occur in at least one document. Let h be a function that maps a set of documents to the set of their timestamps, then, the above conditions are satisfied if $h(D_i \cap D_j) \cap (\tau_i \cap \tau_j) \neq \emptyset$. Similarly to [5, 20], we define edge weights by the following formula:

$$a_{ij} = \begin{cases} \frac{1}{2}\left(\frac{|D_i \cap D_j|}{|D_i|} + \frac{|D_i \cap D_j|}{|D_j|}\right), & \text{if } h(D_i \cap D_j) \cap (\tau_i \cap \tau_j) \neq \emptyset; \\ 0, & \text{otherwise,} \end{cases} \quad (15)$$

where D_i is the set of documents that all include the term f_i and are published during the time period in which f_i is bursty. Zero degree vertices are eliminated from the final graph on which we apply the standard spectral clustering algorithm [16] to segment the bursty terms into $k' \geq k$ non-overlapping clusters. From the result, we discard any groups with less than two terms.

3.2 Cluster Representatives

The second stage of our algorithm starts by determining the set of documents that are associated with each of the k' groups of bursty terms. More specifically, given a group of bursty terms, a document is included in the corresponding document set *Docs* if it contains at least one of the bursty terms of that group and has been published during the burst time period of any of those terms. Next, a *cluster representative* is computed for each of the k' sets *Docs*, using the robust *synthetic prototypes* approach [11] instead of the traditional approach that is based on centroids or medoids. This process is presented in Alg. 3.

Algorithm 2 Segmentation procedure on the bursty terms.

function SegmentTermGraph (A, k')

input : A is the bursty term correlation matrix, k' the desired number of groups

output : $C^{(f)} = \{c_1^{(f)}, ..., c_{k'}^{(f)}\}$ the segmentation solution with $k' \geq k$ groups of bursty terms

1: $C^{(f)} \leftarrow$ SpectralClustering (A, k')
2: $C^{(f)} \leftarrow C^{(f)} \setminus \{\bigcup c_i^{(f)},\ \forall i \in [1, k']$ s.t. $|c_i^{(f)}| < 2\}$
3: **return** ($C^{(f)}$)

Algorithm 3 Construction of bursty synthetic prototypes.

function `ConstructBurstySP` ($C^{(f)}$, $\hat{X}$, p_{docs}, p_{terms})

input : $C^{(f)}$ is the segmentation of `SegmentTermGraph()`, $\hat{X}$ the document matrix with row vectors, p_{docs}, p_{terms} are the parameters for the synthetic prototype construction

output : $SP = \{sp_1, ..., sp_{k'}\}$ the set of synthetic prototypes, $C^{(b)} = \{c_1^{(b)}, ..., c_{k'}^{(b)}\}$ the documents clusters corresponding to the groups of bursty terms $C^{(f)}$

let : f_j the j-th term (here $f_j \in \mathcal{B}$), $k' = |C^{(b)}|$ the number of clusters, D_j the set of documents containing the term f_j, $\hat{X}_{Docs}$ the submatrix of $\hat{X}$ with the rows that correspond to the documents in the set $Docs$, `ConstructSP()` constructs a synthetic prototype, `AssignToClosest()` assigns the documents of a set to the closest of the prototypes provided

1: $Docs_B \leftarrow \oslash$
2: **for** $i = 1...k'$
3: $\quad Docs \leftarrow \oslash$
4: $\quad$ **for each** $f_j \in c_i^{(f)}$
5: $\quad\quad Docs \leftarrow Docs \cup D_j$
6: $\quad$ **end for**
7: $\quad Docs_B \leftarrow Docs_B \cup Docs$
8: $\quad sp_i \leftarrow$ `ConstructSP` $(\hat{X}_{Docs}, p_{docs}, p_{terms})$
9: **end for**
10: $C^{(b)} \leftarrow$ `AssignToClosest` $(\hat{X}_{Docs}, SP)$
11: **return** $(SP, C^{(b)})$

The reasoning behind the choice of synthetic prototypes is that: i) the set $Docs$ may contain a small number 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 need a robust method able to deal with the overlap and to compute the representative for documents of the dominant class in $Docs$ set. All the above fit well with the setting for which the synthetic prototypes approach has been proposed and shown to be more efficient compared to traditional text cluster prototypes [11]. At the end of Alg. 3, k' synthetic prototypes are constructed representing the k' document clusters that were previously formed by the graph segmentation of Sec. 3.1.

3.3 Agglomerative Cluster Merging Step

At the last stage of CBTC method in Alg. 1, and before applying sp$k$-means, the number of clusters k' in which the document set $Docs_B$ has been split, is reduced to the desirable final number k using agglomerative clustering. At each iteration of Alg. 4, the pair of clusters with the most similar synthetic representatives are merged forming a new cluster. Each time, the new synthetic prototypes of the clusters are computed, and the procedure terminates when the number of formed clusters becomes equal to the desired k.

Algorithm 4 Agglomerative cluster merging step.

function `MergeClusters` ($C^{(b)}$, SP, k, p_{docs}, p_{terms})

input : $C^{(b)}$, SP are the output of `ConstructBurstySP()`, k is the final number of clusters to reduce set $C^{(b)}$, p_{docs}, p_{terms} are for the SP construction

output : SP the synthetic cluster prototypes

let : `ClosestPrototypes()` that returns the indexes of the two most similar prototypes in a given set

1: $k' \leftarrow |C^{(b)}|$
2: **repeat**
3: $\quad \{s, u\} \leftarrow$ `ClosestPrototypes` (SP)
4: $\quad c_{su}^{(b)} \leftarrow c_s^{(b)} \cup c_u^{(b)}$
5: $\quad (C^{(b)} \leftarrow C^{(b)} \setminus \{c_s^{(b)}, c_u^{(b)}\}) \cup c_{su}^{(b)}$
6: $\quad sp_{su} \leftarrow$ `ConstructSP` $(c_{su}, p_{docs}, p_{terms})$
7: $\quad SP \leftarrow (SP \setminus \{sp_s, sp_u\}) \cup sp_{su}$
8: $\quad k' \leftarrow k' - 1$
9: **until** $k' == k$
10: **return** (SP)

4. EXPERIMENTS

4.1 Datasets

In our experiments, we used 5 different text datasets. D1 and D2 are subsets of the 20-Newsgroups (20NGs) created by choosing randomly 100 documents from each of the selected categories. D3 is a version of the Reuters-21578 (Reuters) benchmark collection that contains 100 randomly selected documents from each of the 9 top-sized categories, and all the documents of the 10th largest category as they are less than 100. D4 is a subset of TDT5 text collection [7] that originally 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). From this dataset we keep the English documents having single category labels. From the resulting categories, we consider only those with more than 50 documents. D5 is a subset of GoogleNews dataset [1] that contains English articles from the *Technology* category. D5 contains the classes with more than 20 documents and we extract the main content from each article. Details about this dataset and the way it was originally annotated can be found in [12].

For the preprocessing of each raw text collection, we use a standard two-step protocol: i) first, stop-words, numbers and alphanumeric terms are eliminated, and then, ii) Porter's stemming algorithm [18] is applied, in order to map word terms to its canonical stems. The derived stems constitute the final vocabulary of the text collection. This process is performed using the toolkit in [22].

The size of the vocabularies is reduced using a thresholding approach on the term document frequency (term support). Specifically, for datasets D1-D3, terms appearing in less than 5 documents are discarded, while the respective threshold for D4 and D5 is set to 3. For each of the latter two datasets, we compute three quantiles dividing the distribution of terms document frequency in three equal parts. We then exclude the two side parts that correspond to terms with very high and very low document frequency and we keep the rest of the terms. After the previous preprocessing, documents with no terms are also discarded. The characteristics of the final datasets are presented in Tab. 1 and details about the selected topics are presented in the Appendix.

For D4 and D5, the provided timelines of the TDT5 and GoogleNews datasets are used, respectively; documents published on the same day are assigned in the same batch. However, for 20NGs and Reuters datasets we do only make use of the original recorded time-order of documents. Then, we implement a simulation method for the generation of the final streams. As we describe next, this allows for controlling the detection difficulty in the generated stream (i.e. the overlap of topic bursts over time).

Name	Text characteristics					Stream characteristics			
	Classes	N	*Balance*	V	$\overline{V_i}$	T	B	$\overline{\lvert s_i \rvert}$	H_s
D1	10	1000	1	2352	45.89	30	354	33.3	3.030 ± 0.918
D2	10	1000	1	2310	44.54	30	381	33.3	3.030 ± 0.918
D3	10	993	0.93	1566	44.16	30	350	33.1	3.028 ± 0.831
D4	30	4972	0.06	4717	21.54	183	4020	23.8	2.053 ± 0.581
D5	11	268	0.43	1298	59.07	31	400	8.6	0.237 ± 0.543

Table 1: Text and stream characteristics of the datasets used.

– N denotes the number of documents, *Balance* the ratio of the smallest to the largest class, V the size of the vocabulary, and $\overline{V}_i$ the average document vocabulary size.
– T is the number of time windows, B the number of bursty terms, $\overline{|s_i|}$ the average number of documents per window, and H_S the temporal topic entropy.

Quantifying stream complexity. Moreover, we introduce a supervised metric, the *temporal topic entropy* H_S of the stream S, to quantify the "*stream complexity*" of real or artificially generated streams. In particular, H_S measures the mean entropy of the topic labels observed in each of the T non-overlapping time windows (batches):

$$H_S = \frac{1}{T}\sum_{t=1}^{T}\left[-\sum_{i}\frac{n(C_i^{*t})}{N^t}\cdot log_2\frac{n(C_i^{*t})}{N^t}\right]. \quad (16)$$

In the above formula, N^t is the number of documents in the t-th batch and $n(C_i^{*t})$ the number of documents out of those that belong to class C_i^*. H_S gives large values when there is a uniform mixture of topic labels in each batch, which would imply high stream complexity since the timestamps would indeed carry very little useful information, and a zero value when only one topic appears in each batch.

Text stream generator. In order to artificially generate streams with realistic topic bursts, we have at our disposal the document-to-class labeling and the time ordering in which these documents were originally recorded. Hence, our task is not to create text content, but only to assign proper random timestamps to the documents while respecting their original time ordering. The major considerations of our approach are: i) Due to the duality of documents-terms space, it is expected for bursts of relevant documents to also create term bursts over time. ii) It is reasonable to assume that if a topic presents a number of bursts of attention at distinct points in time, then, starting from the beginning of such a burst, the time interval after which each document appears follows an inverse exponential law. Note that this latter, simulates bursts of *peak-and-fade out* shape whereas, more generally, one may consider a *fade in* phase as well.

The stream generation process described next is simple and straightforward. Part of the user-specified parameters are the desired length of the stream timeline T (i.e. time windows) and the number of topics k. For each topic, the generator first decides about the number of bursts to create in a range [1, $bursts_{max}$], their points in time [1, $0.8\,T$] which are not let be very close to the end of the stream, the number of documents in [rd_{min}, rd_{max}] each burst has, and the mean value of the exponential distribution following each burst in the range $[1/\lambda_{min}, 1/\lambda_{max}]$. The parameter initialization uses a uniform random selection of values in the respective valid ranges. Then, a timestamp is assigned to the documents according to the exponential distribution characterizing its topic. As for the rest of the documents that do not belong to any burst, those are spread uniformly in the time period between the first burst of their topic and the end of the stream. For each of the D1-D3 datasets, we generated one stream using the setup presented in Tab. 2. Noteworthy, we observed relatively small differences when comparing the clustering results on multiple generated streams per dataset. Two of the generated streams are illustrated in the Appendix.

Parameter	Value / Selection range
T	30
λ	[0.2, 0.9]
#bursts per topic	{1, 2}
%docs in bursts	[0.7, 0.9]

Table 2: Parameters used in our stream generator.

4.2 Experimental Protocol

For the evaluation of the CBTC method, we conduct experiments on the five text streams described previously. We consider both representations X and XB (see Sec. 2), where X is the traditional BOW representation, and XB is the bursty representation of the corpus. For each representation, sp$k$-means is initialized either randomly or using the proposed approach. The final number of clusters k is set equal to the known number of topics in each dataset.

4.2.1 Evaluation Metrics

The evaluation is based on three supervised measures, with values in the range $[0, 1]$, that are positively correlated with the clustering quality (indicated by the ↑ in Tab. 3), i.e. higher values indicate better solution. For the notations, let N be the total number of documents in the dataset, $C = \{c_1, ..., c_k\}$ be the clustering solution, $C^* = \{c_1^*, ..., c_k^*\}$ be the ground truth documents classes, n_i be the number of documents in c_i, n_i^* be the number of documents in c_i^*, n_{ij} be the number of documents that are clustered into c_j and also belong to c_i^*.

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:

$$Purity = \frac{1}{N}\sum_{j=1}^{k}\max\{n_{ij}\}. \quad (17)$$

F1-measure. This is the harmonic mean of precision P and recall R. Specifically, $P = \frac{TP}{TP+FP}$ and $R = \frac{TP}{TP+FN}$, where TP, FP, and FN denote the True Positive, False Positive, and False Negative observations, respectively:

$$F1 = 2\frac{P \cdot R}{P + R}. \quad (18)$$

Dataset	VSM representation (X)				B-VSM representation (XB)			
		***Purity*↑**	***F1*↑**	***NMI*↑**		***Purity*↑**	***F1*↑**	***NMI*↑**
D1	X (avg.)	0.419	0.423	0.365	XB (avg.)	0.444	0.479	0.410
	(best)	0.510	0.524	0.457	(best)	0.562	0.573	0.490
	X-3k	0.580	0.596	0.578	XB-3k	0.602	0.603	0.558
	X-2k	**0.628**	**0.658**	**0.594**	XB-2k	**0.626**	**0.653**	**0.576**
D2	X (avg.)	0.503	0.515	0.439	XB (avg.)	0.508	0.546	0.451
	(best)	0.571	0.580	0.491	(best)	0.611	0.622	0.535
	X-3k	0.684	0.712	**0.633**	XB-3k	0.684	0.700	0.618
	X-2k	**0.714**	**0.714**	0.619	XB-2k	**0.711**	**0.730**	**0.628**
D3	X (avg.)	0.661	0.649	0.645	XB (avg.)	0.710	0.710	0.686
	(best)	0.771	0.774	0.745	(best)	**0.796**	**0.805**	**0.768**
	X-3k	0.719	0.744	0.703	XB-3k	0.751	0.759	0.745
	X-2k	**0.774**	**0.787**	**0.765**	XB-2k	0.774	0.792	0.766
D4	X (avg.)	0.500	0.457	0.545	XB (avg.)	0.518	0.473	0.584
	(best)	0.564	0.511	0.587	(best)	0.614	0.556	0.641
	X-3k	**0.689**	**0.635**	0.704	XB-3k	**0.701**	**0.638**	0.718
	X-2k	0.678	0.622	**0.712**	XB-2k	0.688	0.625	**0.722**
D5	X (avg.)	0.444	0.441	0.369	XB (avg.)	0.720	0.713	0.710
	(best)	0.557	0.566	0.474	(best)	0.794	0.793	0.772
	X-3k	**0.716**	**0.742**	**0.650**	XB-3k	**0.828**	**0.837**	**0.791**
	X-2k	0.522	0.531	0.504	XB-2k	0.623	0.647	0.658

Table 3: RandInit vs. CBTC initializations – results on the D1-D5 datasets using sp$k$-means.

Normalized Mutual Information. This metric divides the Mutual Information measure to the maximum between cluster and class entropy:

$$NMI = \frac{\sum \frac{n_{ji}}{N} log_2 \frac{\frac{n_{ij}}{N}}{\frac{n_i^*}{N} \cdot \frac{n_j}{N}}}{\max\{H(C), H(C^*)\}}. \quad (19)$$

4.3 Results

In this section, we develop a comparative evaluation for the proposed CBTC method (see Alg. 1 in Sec. 3). Our aim is to study the performance of CBTC in both the traditional BOW representation X (Eq. 2), as well as the bursty term representation XB (Eq. 14). We compare against the randomly initialized sp$k$-means (*RandInit*), where we conduct 100 runs for each dataset from random initializations and report the average metric values (avg.) for all 100 solutions, as well as the metric values (best) corresponding to the solution with the highest cohesion in the 100 runs. Recall that for our method a single run suffices for each dataset, since there is no aspect of randomized initialization.

Apart from the desired final number of clusters k, the overall CBTC method that we propose depends on two kinds of parameters: i) The parameters of the burst detection procedure, namely Kleinberg's two-state automaton, $\alpha = 1$ (as suggested in [13]) and $\gamma = 3$ in order to identify short rather long bursts. ii) The parameters k', p_{docs}, p_{terms} of the CBTC initialization approach. The first out of the latter parameters is set to either $2k$ or $3k$. The other two are involved in the construction of synthetic prototypes (the `ConstructSP()` in Alg. 3) and are set to 0.5 and 0.4, respectively, in all experiments. This actually means that when constructing a synthetic prototype, only 50% of the documents of the cluster will be used, and similarly only the terms corresponding to about the 40% of the magnitude of the initial vector representation. Note also that the construction of each synthetic prototype can be decided to be more robust and refined by applying an incremental construction approach around the cluster medoid (Sec. 2.2). However, this feature comes with the cost of additional parameters which is here avoided as we select in just one step the p_{docs}% of the documents in the cluster that are around its medoid to form the synthetic prototype.

At this point, we should note that for D4, we used all the term bursts identified by Kleinberg's two-state automaton. Contrary, for the rest of the text streams which are also shorter, the CBTC method performed better when for each term only the largest burst weight over the whole stream duration T was considered.

In the results presented in the table Tab. 3 for each dataset, the first two rows correspond to the randomly initialized sp$k$-means (average and best), while the last two rows correspond to CBTC method that use either $k' = 2k$ or $k' = 3k$ starting clusters obtained from spectral graph segmentation. As it is observed from the results for D1 and D2 datasets, the superiority of our method is clear and gives relatively similar and stable results for the two representations X and XB. Using CBTC initialization, X seems slightly better in D1, whereas XB in D2. In D3, and only in terms of the bursty term representation XB, there is the single case where our method does not achieve better results than the best solution of sp$k$-means with XB representation. However, we observe that the deterministic CBTC approach performs much better, according to all three metrics, when compared to the average result of the RandInit sp$k$-means runs.

In D4, for both representations, we can see that our method provides much better results than sp$k$-means. Similarly in D5, which we should underline that is a smaller collection with less imbalanced clusters than D4, and with the lowest stream entropy H_S among the tested datasets (see Fig. 1 in the Appendix). In any case, CBTC has the important advantage of deterministic initialization and seems to provide solutions of good quality in a single run of sp$k$-means.

Additional findings. We also comment on experiments we conducted as part of our empirical validation, but decided not to report them in detail. First, we tried *hierarchical ag-*

glomerative clustering as main procedure (note that CBTC uses a modified such procedure internally). However, as it has been observed in series of papers of the field (e.g. [11]), this approach was quite not competitive in our datasets' high-dimensional and sparse setting as the merging error at low level is usually non-negligible and cannot be corrected at higher level as clusters are nested.

In addition, we tested k-means++ [3] that performs careful randomization using data objects as seeds. Although this consistently improved the average performance in most cases, it did not manage to outperform the best solution already found by RandInit with 100 restarts. This is not unexpected as, for the scale of the datasets we used (determined by N, V, and k values), the 100 random initializations should be already enough to explore large part of the solution space. Important to clarify the difference that CBTC does seek for object seeds, instead, it follows a sophisticated deterministic constructive approach using multiple data objects for each initial centroid.

5. CONCLUSIONS

In this work, we proposed a novel approach for using term burst information in text stream clustering. In addition to the reweighting of independent bursty term weights (*term burstiness*), we aimed at improving the document representation by taking advantage of the fact that most of the important documents for a topic are published during time periods in which many of the main topic terms are simultaneously bursty (*co-burstiness*). This is a key feature to discover characteristic documents for the topics. These documents are subsequently exploited in order to deterministically create efficient initial cluster representatives using the robust synthetic prototypes approach in combination with agglomerative clustering. Comparative experimental results on five text stream datasets indicate that the proposed CBTC approach achieves good clustering performance for both non-bursty (X) and bursty (XB) document representations, despite that it requires only a single run of spherical k-means.

Future work could focus on the one hand on the automated estimation of the number k of clusters (topics), which is now given in advance. Additionally, it would be interesting to test the performance of our method on larger datasets derived from social networking platforms.

6. REFERENCES

[1] A subset of the googlenews dataset, available at: `http://www.db-net.aueb.gr/GoogleNewsDataset/`.

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

[3] D. Arthur and S. Vassilvitskii. K-means++: The advantages of careful seeding. In *Proc. of the ACM-SIAM Symposium on Discrete Algorithms*, SODA '07, pages 1027–1035, 2007.

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

[5] 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.

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

[7] M. Glenn, S. Strassel, J. Kong, and K. Maeda. TDT5 topics and annotations, available at: `http://catalog.ldc.upenn.edu/LDC2006T19`, 2006.

[8] Q. He, K. Chang, and E.-P. Lim. Using burstiness to improve clustering of topics in news streams. In *Proc. of the IEEE Int. Conf. on Data Mining*, pages 493–498, 2007.

[9] Q. He, K. Chang, E.-P. Lim, and J. Zhang. Bursty feature representation for clustering text streams. In *Proc. of the SIAM Int. Conf. on Data Mining*, pages 491–496, 2007.

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

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

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

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

[14] T. Lappas, B. Arai, M. Platakis, D. Kotsakos, and D. Gunopulos. On burstiness-aware search for document sequences. In *Proc. of the SIGKDD Int. Conf. on Knowledge Discovery and Data Mining*, pages 477–486, 2009.

[15] Y. Mitsuishi, V. Novácek, and P.-Y. Vandenbussche. A method for building burst-annotated co-occurrence networks for analysing trends in textual data. In *Proc. of the Conf. on Language Resources and Evaluation*, 2014.

[16] 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.

[17] Y. Ohsawa, N. E. Benson, and M. Yachida. Keygraph: Automatic indexing by co-occurrence graph based on building construction metaphor. In *Proc. of the IEEE Int. Conf. on Research and Technology Advances in Digital Libraries*, pages 12–18, 1998.

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

[19] H. Sayyadi, M. Hurst, and A. Maykov. Event detection and tracking in social streams. In *Proc. of the Int. Conf. on Weblogs and Social Media*, 2009.

[20] M. Vlachos, C. Meek, Z. Vagena, and D. Gunopulos. Identifying similarities, periodicities and bursts for online search queries. In *Proc. of the SIGMOD Int. Conf. on Management of Data*, pages 131–142, 2004.

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

[22] D. Zeimpekis and E. Gallopoulos. TMG: A matlab toolbox for generating term-document matrices from

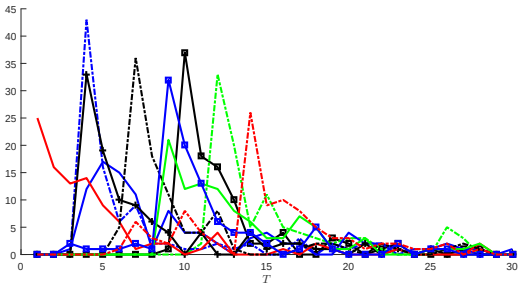
text collections. In J. Kogan, C. Nicholas, and M. Teboulle, editors, *Grouping Multidimensional Data*. Springer, 2006.

[23] X. Zhang and D. Shasha. Better burst detection. In *Proc. of the Int. Conf. on Data Engineering*, pages 146–146, 2006.

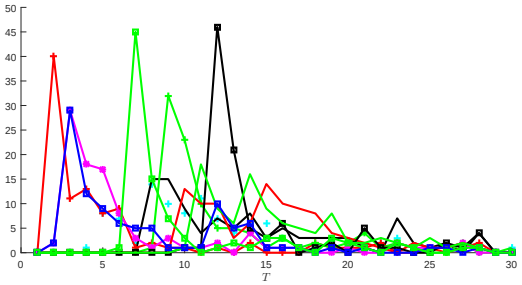
[24] Q. Zhao and P. Mitra. Event detection and visualization for social text streams. In *Proc. of the Int. Conf. on Weblog and Social Media*, 2007.

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

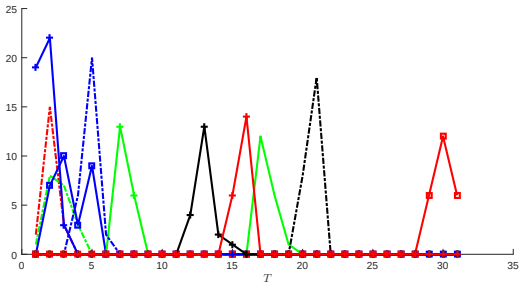
[26] Y. Zhu and D. Shasha. Efficient elastic burst detection in data streams. In *Proc. of the SIGKDD Int. Conf. on Knowledge Discovery and Data Mining*, pages 336–345, 2003.



(a) Generated stream for D1 dataset



(b) Generated stream for D3 dataset



(c) Original stream for D5 dataset

Figure 1: Indicative text streams: the generated streams for D1 and D3 datasets (note: the stream of D2 is similar to that of D1), and the original timeline of D5. Different colors indicate the topics over time.

APPENDIX

Here, we provide details for topics we sampled to create the datasets D1-D5. Moreover, Fig. 1 visualizes the original timeline of D5, and the topic timelines generated (see Sec. 4.2) for datasets D1 and D3 that do not have document timestamps.

Table 4: D1-D3 datasets – Selected topics.

Dataset	Source	Topic description
D1	20NGs	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	acq, corn, crude, earn, grain, interest, interest, money-fx, ship, trade, wheat

Table 5: D4 dataset – Selected topics.

id	Topic ID	Topic description (Source: TDT)
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 PM
28	55200	Iraq: Protection of antiquities
29	55227	Bin Laden videotape
30	55240	US troops fire on Mosul crowd

Table 6: D5 dataset – Selected topics.

id	Topic ID	Topic description (Source: Googlenews)
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