Soc. Netw. Anal. Min. (2017)7:56
DOI 10.1007/s13278-017-0476-8

ORIGINAL ARTICLE

Incremental clustering with vector expansion for online event detection in microblogs

Ozer Ozdikis[1] · **Pinar Karagoz**[1] · **Halit Oğuztüzün**[1]

Received: 28 November 2016 / Revised: 16 August 2017 / Accepted: 22 October 2017

Abstract Identifying similarities in microblog posts for event detection poses challenges due to short texts with idiosyncratic spellings, irregular writing styles, abbreviations and synonyms. In order to overcome these challenges, we present an enhancement to the incremental clustering techniques by detecting similar terms in microblog posts in a temporal context. We devise an unsupervised method to measure the similarities online using co-occurrence-based techniques and use them in a vector expansion process. The results of our evaluation performed on a tweet set indicate that the proposed vector expansion method helps identify similarities in tweets despite differences in their content. This facilitates the clustering of tweets and detection of events with higher accuracy without incurring a high execution cost.

Keywords Online event detection · Clustering · Vector expansion · Statistical text analysis · Microblogs

1 Introduction

Microblogs contain a vast amount of data with millions of registered users. In particular, Twitter, one of the most common microblogging platforms, provides a practical way for people to post short text messages as tweets and share them with their followers. Since real-world events usually have a direct impact on the content of tweets, it is a useful medium to be informed about latest news, follow popular events, and keep up with hot topics.

Data mining techniques on content-rich textual documents have long been studied as a part of the field of Topic Detection and Tracking (TDT) (Allan 2002). The increasing usage of Twitter as a communication platform has led to TDT techniques that were previously applied for newspaper articles and blog posts being extended and adapted to perform event detection using tweets. In these studies, an event is defined as an activity that happens at a specific time and place (Allan 2002; Cordeiro and Gama 2016; Imran et al. 2015). There are interesting real-life applications of event detection systems (De Choudhury et al. 2008; Sakaki et al. 2013). Clustering documents according to their semantic similarity around topics is a widely adopted approach to detect events (Sankaranarayanan et al. 2009; Silva et al. 2013; Yin et al. 2012). In order to achieve effective clustering in Twitter, one of the basic challenges to be overcome is related to content. Twitter has distinct characteristics that differentiate it from text in newspapers and blogs (Kim et al. 2013). The limitation of the tweet length to 140 characters, and idiosyncratic spellings due to spontaneously generated and unedited content are two major reasons for the existing text mining methods to be enhanced. People make spelling mistakes, follow non-traditional writing conventions, and abbreviate long words because of the character length limitation. Even when they refer to the same event, they may express it in many different ways. Therefore, we claim that, in an event detection process where tweets are modeled in the standard vector space model based on their textual content, clustering performance can be improved if similar terms in tweets are discovered and utilized in a vector expansion process.

✉ Ozer Ozdikis
ozer.ozdikis@ceng.metu.edu.tr; e116501@ceng.metu.edu.tr

[1] Department of Computer Engineering, Middle East Technical University, Ankara, Turkey

The content in Twitter is predominantly driven by the user community rather than being based on a thesaurus such as WordNet or Wikipedia; therefore, a static thesaurus cannot effectively cover this user-generated content (Agirre et al. 2009; Varga et al. 2014). Moreover, the similarity of two terms may change depending on the time and context. As a consequence, we propose a method to identify similar terms using co-occurrence-based statistics on the tweet content and score term similarities online within a time window.

A clustered set of tweets may not necessarily indicate a new event in Twitter. A noticeable number of tweets have been reported not to be related to a real-world event but are concerned with the ordinary daily activities of people in the form of chat, personal updates, conversations, and spam (Weng and Lee 2011). Burst detection can be applied to distinguish event-triggered patterns in tweet traffic by analyzing temporal frequency distributions in specific features of tweets (e.g., frequency of a term, number of tweets per unit time) (Fung et al. 2005; Kleinberg 2002; Li et al. 2012; Xie et al. 2013). Therefore, as similar tweets are grouped around topics by clustering, we also apply a burst detection method to detect new events by analyzing surges in term frequencies in near real time. In our evaluations, we demonstrate the implications of the proposed methods on the clustering and burst detection performance. We aim to show that the proposed enhancement using online similarity analysis and vector expansion yields more coherent clusters and helps accurate identification of event-related tweets.

The contributions of this paper can be summarized as follows:

- We propose an enhancement for existing online incremental clustering methods by introducing a vector expansion process, which we call *Incremental Clustering with Vector Expansion* (*ICVE*).
- ICVE automatically extracts, scores, and utilizes term similarities in a temporal locality using statistical methods; therefore, it can be applied to any language without relying on an existing thesaurus.
- The proposed solution can be efficiently executed online, making a single pass on the incoming tweet stream without any post-processing. We make no *a priori* assumption about the number of clusters or their topics.

The remainder of the paper is organized as follows. First, a summary of related work is given in Sect. 2. This is followed by the description of our online event detection method in Sect. 3. We discuss the results of our evaluations and compare them with the results of the baseline method in Sect. 4. Finally, in the last section we summarize our findings and conclude with suggestions for further research.

2 Related work

We study the state of the art on event detection and similarity analysis for tweets in three categories. In Sect. 2.1, we present an overview of online event detection techniques proposed to detect events using tweets. In Sect. 2.2, we discuss the clustering methods in the literature, particularly focusing on the clustering of tweets and document streams. Finally, in Sect. 2.3 we address recent efforts to improve the accuracy of document clustering techniques and elaborate on the studies that aim to discover term similarities in short texts.

2.1 Event detection in microblogs

The techniques in the literature to perform event detection in Twitter have recently been reviewed in Atefeh and Khreich (2015), Cordeiro and Gama (2016). Clustering tweets according to their similarity is a widely used method for event detection. For example, the news processing system called TwitterStand employs an online clustering algorithm that measures the cosine similarity between the feature vectors of tweets and clusters them into topics (Sankaranarayanan et al. 2009). A similar approach is adopted in Yin et al. (2012) in which a single-pass incremental clustering algorithm is developed that automatically groups similar tweets into event-specific topics. Incremental clustering of tweets is also applied for online event detection in Shou et al. (2013), Zhou et al. (2016).

By taking tweet processing time into consideration, algorithms can be classified as retrospective or online. Retrospective algorithms process a corpus of tweets to identify events discussed therein, while the objective of online algorithms is to identify events as they happen. Due to a large volume of tweets arriving at a fast rate, online algorithms have to decide in a timely manner whether an event has just occurred. Burst detection is a widely adopted method for this purpose (Kleinberg 2002; Xie et al. 2013), which is also applied by Yin et al. (2012) to determine which topic clusters can be associated with a real-world event.

Regarding the subjects of events, event detection can be designed either to focus on a specific topic (usually implemented by querying specific terms in tweets) or applied to an open topic domain. For example, Sakaki et al. (2013) detect earthquakes by collecting tweets containing the terms *earthquake* or *shaking*. Cheong and Lee (2011) collect tweets about terrorism events using terror-related keywords. TwitInfo (Marcus et al. 2011) is another keyword-based solution, where tweets containing a given list of keywords are collected, and the peaks in their histograms are examined to detect events. Alternatively, TwitterStand uses tweet samples provided by Twitter services and the posts of a set of handpicked Twitter users without targeting a specific topic (Sankaranarayanan et al. 2009). Similarly, Yin et al. (2012) utilize the tweets returned by Twitter's location-based search API to apply an open-domain event detection.

Although there is an expanding literature on event detection, examination of the effect of vector expansion on online event detection performance has not been a focus of attention thus far. In this work, we cluster the tweets received from an online tweet stream and apply a burst detection method in order to detect events on an open topic domain in near real time. For burst detection, we analyze surges in the frequencies of terms in active clusters. By applying both clustering and burst detection, we can group similar tweets about a topic and detect events and event-related tweets on that topic in a timely manner.

2.2 Clustering

The clustering of textual items retrieved from data streams, particularly from social networks, is an active research area (Atefeh and Khreich 2015; Cordeiro and Gama 2016; Fang et al. 2014; Imran et al. 2015; Silva et al. 2013; Zhou et al. 2016). Incremental clustering is regarded as an appropriate method for grouping continuously received textual items (Shou et al. 2013; Yin et al. 2012; Zhou et al. 2016). Basically, in incremental clustering, if a received item is found to be sufficiently similar to an existing cluster, it is added to the most similar cluster. Otherwise, a new cluster is created with that item. There are several ways to decide whether a newly arriving data point should join the most similar cluster. Sankaranarayanan et al. (2009) and Yin et al. (2012) use constant thresholds to make this decision. Another approach is to calculate the mean and standard deviation of similarities between tweets and compare the similarity of a recently posted tweet with these values (Aggarwal and Subbian 2012). Shou et al. (2013) define a minimum bounding similarity measure as the weighted average of similarities between the cluster's centroid and the tweets in the cluster.

Constraints concerning memory size and processing capacity can become an issue to resolve in online clustering on continuous data streams (Sankaranarayanan et al. 2009; Shou et al. 2013). In particular, for high rates of streaming input data, a decision must be made to decide which stale clusters to delete from the memory. A straightforward method is to remove the least recently updated stale cluster when the capacity limit is reached (Aggarwal and Subbian 2012; Aggarwal and Yu 2006). In TwitterStand, the activity of clusters is periodically checked, and those with no recent change are removed (Sankaranarayanan et al. 2009). Similarly, the average timestamp of recent arrivals in each cluster can also be used to identify stale clusters. This is facilitated in Aggarwal et al. (2003) by the use of a micro-clustering approach, in which the definition of a cluster feature vector is extended with the timestamp components of the data points in the cluster. Alternatively, instead of keeping all the clusters in the memory and removing the stale ones when necessary, it is possible to select only the event-related clusters for maintenance at their creation. This approach is adopted in Yin et al. (2012) by allowing only the tweets that contain a bursty term to form clusters.

Fragmentation, i.e., duplicate clusters about a single topic, can be a major problem with online clustering algorithms (Sankaranarayanan et al. 2009). Merging active clusters in a post-processing stage can be a solution to this problem. However, comparing clusters with each other to find duplicates may not be suitable for online scenarios while tweets are still received continuously from the tweet stream. Therefore, because of its computational complexity, post-processing of clusters can only be performed offline or during less busy tweet traffic (Shou et al. 2013; Zhou et al. 2016).

The proposed enhancement for online clustering in this work aims to mitigate the fragmentation problem by discovering and utilizing term-level similarities in tweets online. In other words, ICVE reduces the number of duplicate clusters without requiring any post-processing or merging of clusters, as we demonstrate in Sect. 4. In deciding whether to add an incoming tweet to an existing cluster, we adopt an approach similar to Yin et al. (2012) and use constant thresholds. We use the method in Sankaranarayanan et al. (2009) to identify stale clusters. However, we would like to note that the vector expansion technique that we propose is not dependent on the particular techniques adopted for cluster processing.

2.3 Co-occurrence statistics and vector expansion

Among the efforts to tackle the problem of handling the spelling variances in documents to be clustered, a latent semantic analysis (LSA)-based *k*-means clustering method is described in Song and Park (2007). In that work, the authors aim to find relevant documents in the corpus that do not necessarily share any common words by reducing the dimension of document vectors. Similarly, Jun et al. (2014) build a clustering method by combining *k*-means clustering with dimension reduction through singular value decomposition (SVD) and principal component analysis (PCA). In these studies, the dimensions of document vectors are reduced in a preprocessing phase before applying the clustering algorithm. However, clustering an online stream of user-generated content with no prior assumption about the vocabulary and topic introduces certain challenges due to noisy data, diverse topics, and frequent spelling mistakes (Nguyen and Jung 2015). New terminology can emerge when different topics are discussed by users in their posts (Varga et al. 2014). This can result in each new post bringing about changes in the modeled vector space, which can limit the applicability of static training-based solutions. Furthermore, data preprocessing, formation of the vector representations and clustering algorithms must be executed in a timely manner to keep pace with the continuously incoming data items.

In order to resolve spelling variances in tweets, Kim et al. (2013) define a set of rules to handle abbreviations, minor typing errors, and terms separated by spaces. They apply these rules on terms that either start with a capital letter or are enclosed by quotation marks. Cotelo et al. (2015) and Kaufmann and Kalita (2010) propose methods to find and correct words in tweets that are not found in the vocabulary of a language. These method use language-specific dictionaries and handcrafted rules to normalize out-of-vocabulary words in tweets. Phuvipadawat and Murata (2010) identify the proper nouns in tweets by named entity recognition and boost the effect of these proper nouns. Zhou et al. (2016) calculate Jaro–Winkler distance between key terms in tweets and consider two key terms as the same if their similarity based on string distance is above a threshold. Fang et al. (2014) apply a weighing method using common phrases and hashtags in addition to the terms in tweets to better measure the similarity of tweets. Voorhees (1994) propose vector expansion as a solution to the issue of word mismatches in documents. The authors use a thesaurus, namely WordNet, to find the synonyms of the words in a query text. However, due to the noisy lexical nature of microblog posts with frequent irregularities and incorrect spelling, it may not be possible to identify entities in a pre-existing knowledge source (Varga et al. 2014).

As an alternative to using manually compiled thesauri, similar words can be identified by analyzing their distribution in large text corpora. Using first- and second-order co-occurrences of terms in documents to extract similarity information is discussed in Rapp (2002). A strong first-order relationship is observed when two terms appear frequently together in texts, whereas two terms have a second-order relationship if they frequently appear together with the same set of terms, i.e., having similar lexical neighborhood. A comparison of distributional and WordNet-based similarity measures is given in Agirre et al. (2009). The authors state that the distributional similarity was effectively used to cover the items that did not exist in the WordNet vocabulary. A similarity thesaurus, which includes term–term similarity values extracted from the co-occurrence statistics, is presented in Qiu and Frei (1993). In another study, Lin et al. (2003) focus on finding synonyms by analyzing the results obtained from search engines and evaluating the commonality of translations in bilingual dictionaries.

Vector expansion has been applied in various studies to solve similar problems. For example, Cao et al. (2008) use it to expand query terms in order to improve the query results. Applying query expansion to increase the performance of microblog searching is investigated in Chen et al. (2013), Cotelo et al. (2014), Magdy and Elsayed (2016). The earthquake prediction system in Okazaki and Matsuo (2010) employs a classifier to decide whether a tweet containing a query word really concerns an earthquake, using the words before and after the query words as the context. Thomas and Sindhu (2015) present a survey of recent studies on content-based semantic relations in tweets. Using vector expansion methods for retrospective event detection has been studied in Ozdikis et al. (2012a, b, 2014). However, extracting and measuring similarities for online event detection with no prior assumption about the topic brings new challenges (Cordeiro and Gama 2016).

ICVE does not require pre-compiled thesauri or language-specific analyses. We adopt statistical methods for similarity analysis of terms, since methods based on string distance can treat two unrelated terms as referring to the same concept only because of their spelling similarity. Moreover, statistical methods help identify terms that may refer to the same concept in a temporal locality even if they do not have similarity in their spelling. As demonstrated in Sect. 4, this can be achieved without remarkable cost that would hinder online processing of tweets.

3 Online event detection using ICVE

Our event detection method is composed of two major stages, namely a clustering stage and a burst detection stage. In order to enhance the accuracy of clustering and burst detection, we analyze the similarity of terms in tweets in a temporal context and use these similarities for vector expansion. This is achieved by an additional stage of similarity analysis prior to the clustering and burst detection stages.

The temporal context at time t can be described by the tweets posted before and after t. By keeping a number of tweets in a tweet history after they are processed, we can easily access the past portion of the time context. The future portion of the context requires "foreseeing" the tweets to be processed later. For this purpose, we introduce the concept of look-ahead cache. When a set of tweets is

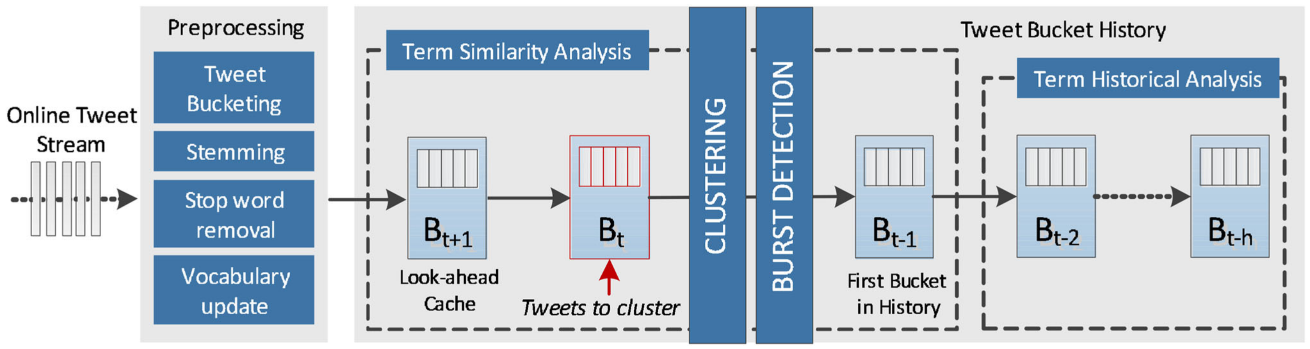

Fig. 1 Proposed online event detection process

received from the Twitter stream, rather than immediately applying the clustering and burst detection methods, we first keep these tweets in the look-ahead cache to be used in the context description. In other words, event detection follows the online tweet stream one-cache size behind.

The overall event detection process is depicted in Fig. 1. As new tweets are received from the stream, they are processed in sliding windows, which we call *tweet buckets* or simply *buckets*, denoted by B. Tweet bucketing is part of the preprocessing, along with tweet stemming, stop word removal, and vocabulary update, as shown in Fig. 1. As a specific amount of tweets accumulate in a bucket, the bucket is passed to the event detection pipeline to be used as the new look-ahead cache. The look-ahead cache is denoted by B_{t+1} in the figure. The previous bucket in the look-ahead cache now becomes the bucket to apply clustering and burst detection, represented by B_t. The flow proceeds by updating the other buckets in the history in a similar way. The figure shows B_{t+1}, B_t and B_{t-1} in a specific group labeled *Term Similarity Analysis*, since tweets in these three buckets are utilized in the word co-occurrence and vector expansion analyses. The buckets grouped in *Term Historical Analysis* are used as a memory of past tweets to calculate historical term-frequency statistics. The following section presents data collection and modeling strategy before moving on to the details of clustering, burst detection, and similarity analysis.

3.1 Data collection and preprocessing

Tweets posted in a region can be collected using Twitter's streaming API[1] and its geographical filtering. Specifically, in our implementation, to detect events occurring in Turkey, we define a geographical filter based on the boundary coordinates of the country in terms of latitude and longitude, and use the Java library of Twitter4j[2] for service connection and tweet collection tasks. We used the public streaming API as a free and commonly used method for tweet collection in the literature. Bounding box filtering helped us obtain a random sample of tweets posted in Turkey, which also enabled us to observe various writing styles due to language-specific characters in Turkish alphabet (i.e., *ı, ö, ü, ç, ş, ğ*).

In similar previous studies, we observe that the size of the sliding windows is determined either in terms of the clock time or in terms of the number of tweets (Fung et al. 2005; Marcus et al. 2011; Yang et al. 1998). In this work, we devise a hybrid bucket size model that employs both time and tweet count. The time window for a bucket is determined as 1 min; but if the tweet count in a bucket is below a certain threshold after 1 min since the receipt of the first tweet in that bucket, we wait for more tweets to accumulate. We set this threshold value as 200, which is approximately the average number of tweets per minute received by our streaming client in a day.

The reason for devising the hybrid bucket size model is twofold. Firstly, the flow of tweets received by the aforementioned streaming client considerably changes depending on the time of the day. As a result, if the bucket size was determined in terms of a fixed time interval, disparities in tweet rates would make it harder to detect surges in tweet counts and manage cluster lifetimes. Secondly, if buckets were generated from a fixed number of tweets, then tweets about a minor event could be dispersed at distant buckets, probably due to a major event occupying a larger portion of the buckets. This would inherently limit the number of concurrent events that can be detected. The hybrid bucketing approach using both time and tweet count yields a more homogeneous bucket size throughout the day and does not require any *a priori* assumption about the number of concurrent events.

When the time and tweet count requirements are satisfied to generate a bucket, a bucket B_i is formed as a list of n tweets denoted by $[B_{i,1}, \ldots, B_{i,n}]$. Tweets are considered bag-of-words in the vector space model (Yang et al. 1998). In the preprocessing phase, given a bucket B_i, the

[1] https://dev.twitter.com/overview/api.

[2] http://twitter4j.org/en/index.html.

stemming and stop word removal procedures are applied to its tweets, and the vocabulary statistics (i.e., term and document frequencies) are updated. We use a stop words list for Turkish[3] that is similar to its English equivalents. Space characters, punctuations, and parentheses are used for tokenization of the text. We also remove URLs and user mentions (terms that start with @) from the tweet text. We select only the words written in ASCII characters (characters ranging from 0–255) and Turkish alphabet, which clean emojis from the text. We do not apply any special handling for hashtags.

Associated with each tweet $B_{i,j}$, we keep its unique id, its posting time, the list of its stemmed terms with their frequencies, and a tweet vector $\mathbf{B_{i,j}}$. A tweet vector is a normalized tf–idf vector represented as the tuples of the form $\langle x, w \rangle$ where x is a term in the tweet and w is its normalized tf–idf value (Fang et al. 2014; Ozdikis et al. 2012b). We would like to note that except for the stemming library[4] and the stop words list in Turkish, the methods we present in this paper do not depend on the language, and they can be applied for other geographical regions.

3.2 IC: Incremental clustering

The clustering technique we employ in this work is an enhancement to incremental clustering, an approach widely employed for clustering streaming data (Sankaranarayanan et al. 2009; Shou et al. 2013; Yin et al. 2012; Zhou et al. 2016). In this section, we present a traditional incremental clustering algorithm. Then, in Sect. 3.3, we introduce how to extract term similarities in a time context and use them in the vector expansion process to develop ICVE.

The clustering algorithm is executed on the most recent bucket in a single pass, as the tweets are continuously received from the tweet stream and accumulated in a new bucket. The system maintains a list of active clusters and updates them incrementally. A cluster c is represented by a cluster centroid vector $\mathbf{c}$, its creation time, a list of term-frequency pairs, and a list of tweet *id*s added to the cluster grouped by their buckets. The term-frequency list for a cluster is generated using the terms and their total frequencies in the tweets of that cluster. The cluster centroid vector $\mathbf{c}$ is defined in a similar way as the tweet vector, i.e., as the normalized tf–idf values of the terms in the cluster. If a new tweet is added to the cluster, the term-frequency information and the centroid vector of the cluster are updated using the terms and their frequencies in the tweet. The set of tweet ids received in bucket B_i and added to cluster c is denoted as B_i^c. This information helps us obtain term-frequency histograms for the clusters and apply burst detection techniques presented in Sect. 3.4.

When a tweet is to be clustered with incremental clustering, its tweet vector is compared with the centroid vectors of the active clusters using cosine similarity (Sankaranarayanan et al. 2009). If the most similar cluster has a similarity score higher than a predefined threshold, called the *merge threshold*, then the tweet is assigned to that cluster. This assignment causes the cluster features to be updated using the features of this recently added tweet. If no similar cluster is found, a new cluster is created using the features of this tweet. After processing all tweets in a bucket, we check the active clusters whether they have been stale and terminate a cluster if no tweet has been added to it for the last three consecutive buckets, which we call *Cluster Maximum Idle Time*. The termination process can store the cluster data in a persistent storage for reporting purposes or simply discard it depending on the number of tweets collected in that cluster.

3.3 ICVE: Incremental Clustering with Vector Expansion

We propose an enhancement to the traditional incremental clustering algorithm described above by discovering and measuring the similarities between the terms, and using these similarities to perform a vector expansion prior to clustering. Considering the process flow in Fig. 1, where B_t represents the bucket of tweets to cluster, the proposed enhancement involves an expansion of tweet vectors in bucket B_t and the centroid vectors of active clusters. This is achieved by leveraging the contextual information in the neighboring buckets of B_t in the following four steps:

1. The terms in B_t to be compared for their pairwise similarities are identified.
2. Co-occurrence vectors are generated for the terms to compare using the co-occurrence statistics in B_{t+1}, B_t, and B_{t-1}.
3. Term similarities are calculated as real numbers in the range of [0,1] using the co-occurrence vectors.
4. Tweet and active cluster vectors are expanded using the extracted similarities.

These four steps are further explained in the following sections. After the expanded vectors are obtained for the tweets in B_t and for the active clusters, incremental clustering described in Sect. 3.2 is performed on these expanded vectors. Thus, we refer to this clustering technique as *Incremental Clustering with Vector Expansion* (*ICVE*).

[3] https://github.com/ahmetax/trstop/.

[4] http://coltekin.net/cagri/trmorph.

3.3.1 Identifying terms to compare

The significance of selecting discriminative terms for vector expansion has been discussed in Cao et al. (2008). In that work, the authors argue that not all expansion terms are useful for improvement. Moreover, in our case, the number of distinct terms in a bucket can be large, thus comparing all terms in B_t with each other in terms of their similarities can be a time-consuming task that is not suitable for online stream processing. Therefore, at this step, we aim to identify the most discriminative terms in B_t to be analyzed for their similarities. In this paper, we refer to these terms as *terms to compare*.

The terms to compare in B_t are determined by applying *concept decomposition*, in which we perform a separate clustering procedure only on the tweets in B_t with a high merge threshold to obtain coherent tweet collections (Aggarwal and Zhai 2012). Here we assume that the most descriptive terms for the detected concepts also represent the most discriminative terms in B_t. Therefore, the terms with high tf–idf values in the centroids of these clusters are selected as terms to compare. We would like to note that the clusters mentioned here are used only for concept decomposition to determine terms to compare. This means that they are not included in event detection process, and they are discarded after the similarity analysis.

We explain the procedures in finding terms to compare using an earthquake scenario. In case of an earthquake event, it is likely to observe a number of tweets in B_t that contain the terms *earthquake* or *shake*. Considering the tweets containing either of these terms as two separate sets, the concept decomposition process applied to B_t is expected to group these tweets in two separate clusters. Normally, these two terms would have the highest tf–idf scores in the corresponding cluster centroids, and thus *earthquake* and *shake* would be included in the set of terms to compare to be analyzed for their statistical similarities.

3.3.2 Generation of co-occurrence vectors

Temporal locality should be taken into account when extracting similarities between terms (Bansal and Koudas 2007). Two terms that are strongly related to each other in one context may be unrelated in another. As previously mentioned, temporal context is defined as the tweets posted before and after the bucket to be clustered. More specifically, before applying the event detection process to the bucket B_t, we discover the contextual similarities between the terms to compare in B_t using the tweets in B_{t+1}, B_t, and B_{t-1}, as shown in Fig. 1.

Term associations are useful in a vector expansion process since the expansion may add useful information to the vector that may not be explicitly specified in the corresponding text. Our primary focus is the second-order relations, which are supposed to identify the words that can be used interchangeably (Rapp 2002). By preparing co-occurrence vectors and measuring their similarities, we can obtain similarity scores in the interval of [0,1] between the term pairs. There are several ways to produce co-occurrence vectors. In this work, we experiment with the following two: (1) second order and (2) dimension-reduced using SVD. Once the co-occurrence vectors are generated using one of these methods, similarity of two co-occurrence vectors is measured using cosine similarity.

Second Order (SO): This method generates co-occurrence vectors based on the number of times that a term appears together with every other term in the same tweet. Using the terms and tweets in B_{t+1}, B_t, and B_{t-1}, we form a binary term-tweet matrix A where $A[i][j]$ is set to one if the term corresponding to the ith row appears in the tweet corresponding to the jth column. Otherwise, $A[i][j]$ is set to zero. Multiplying A with its transpose A^T gives the pairwise co-occurrence counts of the terms. In order to speed up the process, since we need the co-occurrence vectors only for the terms to compare as identified in Sect. 3.3.1, we process only the nonzero entries in matrix A, and obtain the co-occurrence vectors without applying matrix multiplication. The co-occurrence of a term with itself is discarded since it is trivial. The computational complexity of this process is then $\mathcal{O}(mnd)$, where mn is the dimension of A, and d represents the number of terms to compare (with $d \ll m$). While comparing the co-occurrence vectors of two terms for their similarity, their co-occurrence values with each other are set to zero and the vectors are finally normalized to be used in the calculation of the similarity score.

Dimensionally reduced vectors using SVD: SVD is a method that decomposes a matrix A into three matrices U, S, and V with specific properties (Berry et al. 1995; Deerwester et al. 1990). Matrix S has the property that if k is the rank of matrix A, only the first k values in the diagonal of S are nonzero. Moreover, S can be reduced by keeping the first few values in the diagonal and setting the remaining values to zero. If we call this reduced matrix S_R, a reduced approximation of A, called A_R, can be produced by taking the product $\mathrm{US_R}V^T$. By reducing the dimension of the term-tweet matrix A to A_R using SVD, trivial and incidental co-occurrences can be filtered out leaving only the significant ones. The reduced term co-occurrence matrix is then generated from the product of A_R with A_R^T, as given in Eq. (1). While comparing the co-occurrence vectors of two terms x_i and x_j, their co-occurrence values with each other are set to zero. The co-occurrence vectors are finally normalized to be used in the calculation of the similarity scores.

$$A_R A_R^T = \mathrm{U}S_R V^T \left(\mathrm{U}S_R V^T\right)^T = \mathrm{U}S_R^2 U^T \quad (1)$$

The extent of dimension reduction can be defined by changing the reduction ratio, i.e., the ratio of the rank of S_R to the rank of S. In this work, we empirically determined a reduction ratio of 80%, i.e., kept the top 20% of the diagonal in S. In our implementation, we used the JAMA[5] library for SVD operations. The computational complexity of SVD is $\mathcal{O}(m^2 n + mn^2)$ (Berry et al. 1995). The co-occurrence vector generation also includes the multiplication of decomposed matrices in $\mathrm{U}S_R^2 U^T$, which is $\mathcal{O}(m^2 n + mn^2)$.

3.3.3 *Calculation of statistical term similarities*

After generating the co-occurrence vectors, we find the cosine similarities between the co-occurrence vectors of each pair of terms to compare. The calculated cosine similarities represent the similarity scores between the corresponding terms. Given the normalized co-occurrence vectors for d terms to compare, the calculation of similarities between the term pairs is $\mathcal{O}(md^2)$. As a result, the output of this second step in term similarity analysis is the list of tuples $\langle x_i, x_j, s_{i,j} \rangle$, where x_i and x_j are two terms to compare, and $s_{i,j}$ represents their similarity score. For each term to compare, we only keep the top few tuples with the highest similarity scores above a certain threshold (i.e., $s_{i,j} >$ *term similarity threshold*). We refer to the number of similarity tuples we keep for a term as the *term similarity count*.

3.3.4 *Vector expansion*

Before performing the incremental clustering procedures, pairwise similarity scores between the terms to compare calculated in the previous step are used for the expansion of tweet vectors and centroid vectors of active clusters. The steps in vector expansion are given in Algorithm 1. This vector expansion algorithm first initializes an expanded tf–idf vector $\mathbf{v^e}$ with the values in $\mathbf{v}$, and for each $\langle x_i, w_i \rangle$ in $\mathbf{v}$, it finds other terms x_j that are similar to x_i with $s_{i,j} > 0$ and updates the weight of x_j in the expanded vector $\mathbf{v^e}$ by adding $w_i s_{i,j}$ to its previous weight w_j. After all terms in $\mathbf{v}$ are processed, the expanded vector is normalized. Given the similarity scores between term pairs to compare, the time complexity of this expansion operation is $\mathcal{O}(mnd^2)$ for all tweet vectors in B_t.

[5] http://math.nist.gov/javanumerics/jama.

Algorithm 1 Vector Expansion

```
1:  Input: Normalized tf-idf vector $\mathbf{v}$ as a list of tuples $\langle x_i, w_i \rangle$
2:  Input: List of pairwise similarity scores in the form of $\langle x_i, x_j, s_{i,j} \rangle$
3:  Output: Expanded tf-idf vector $\mathbf{v^e}$ as a list of tuples $\langle x_i, w_i^e \rangle$
4:  Initialize $\mathbf{v^e}$ with the vector $\mathbf{v}$
5:  for each $\langle x_i, w_i \rangle$ in $\mathbf{v}$ do
6:      for each $\langle x_i, x_j, s_{i,j} \rangle$ with $(s_{i,j} > 0)$ do
7:          Calculate $(w_i \times s_{i,j})$ for the term $x_j$
8:          if $x_j$ does not exist in $\mathbf{v^e}$ then
9:              Set its weight in $\mathbf{v^e}$ to $(w_i \times s_{i,j})$
10:         else
11:             Update the weight of $x_j$ in $\mathbf{v^e}$ as $w_j^e = w_j^e + (w_i \times s_{i,j})$
12:         end if
13:     end for
14: end for
15: Normalize the vector $\mathbf{v^e}$
```

Once expanded tweet and cluster vectors are obtained, the incremental clustering method described in Sect. 3.2 is applied to these vectors. In other words, when clustering a tweet $B_{t,i}$ using ICVE, we compare its expanded tweet vector with the expanded centroid vectors of the active clusters. The clusters generated as a result of this vector expansion are found to be more coherent and less fragmented, as demonstrated in the evaluation section (Sect. 4).

3.4 Burst detection

Burst detection stage aims to detect sudden surges in the frequency of terms received from the tweet stream. A term may be bursty in the tweet stream, but if it is mentioned in different clusters, it does not necessarily indicate a newsworthy event. Therefore, in order to mark an event about a topic, we also expect a bursty term in a tweet stream to be bursty in one of the active clusters. As a result, improving the accuracy of tweet clustering becomes specifically important for effective event detection, since fragmented clusters can make it less likely to detect a burst related to an event. The process of detecting bursts and marking new events in active clusters is performed in two steps explained below.

3.4.1 *Detection of bursty terms in clusters*

As presented in Fig. 1, after the tweets in bucket B_t are processed by the clustering algorithm, we execute the burst detection method on the active clusters. For the detection of bursty terms, we adopt a method similar to the selection of bursty keywords in Kim et al. (2013). Let $B_{i|x}$ represent the set of tweets in bucket B_i that contain the term x. The process starts with finding the average frequencies $\mu_t(x)$ of the terms in the oldest $(h-1)$ buckets of the bucket history. We define *burst ratio* of a term x in B_t as the ratio of $|B_{t|x}|$ to $\mu_t(x)$, and we designate x as a *bursty term* if its burst ratios in B_t and B_{t-1} are higher than a prescribed threshold Δ, called the *burst ratio threshold*.

This process finds the bursty terms in a tweet stream without considering the distribution of these terms in active clusters. However, tweets that contain a bursty term may be related to different topics. Therefore, for each bursty term identified in the tweet stream, we also find its distribution in active clusters. Reminding that B_t^c represents the set of tweet ids in bucket B_t that are added to the cluster c, and the tweet details for B_t and B_{t-1} are available in the memory, it is straightforward to identify the set of tweets in B_t^c and B_{t-1}^c that include the term x. These two tweet sets are denoted by $B_{t|x}^c$ and $B_{t-1|x}^c$, respectively. The burst ratios of x with respect to a cluster c can then be found in a similar way, i.e., by dividing the cardinalities of $B_{t|x}^c$ and $B_{t-1|x}^c$ by $\mu_t(x)$. If these ratios are also above the threshold, we signal the detection of a new event, and the cluster with a bursty term is marked as an *event cluster*. These bursty terms are used to extract event features from the event cluster.

3.4.2 Extraction of event features

Once a burst is detected in a cluster, we extract the corresponding event features to obtain a human understandable description of the event. If a term x is identified as a bursty term in cluster c, then the tweets $B_{t|x}^c \cup B_{t-1|x}^c$ are marked as the bursty tweets of that event. If multiple bursty terms are detected at time t in an event cluster, tweets that include at least one of these terms in the cluster constitute the set of the event's bursty tweets. This means that the bursty terms detected in the same bucket and in the same event cluster are treated as the descriptors of the same event. An event's bursty tweets are important since they are the first and probably the most relevant tweets describing the event.

An event is described with three features, namely the event time, event centroid vector, and the best tweet for the event. The posting time of the first bursty tweet is assigned as the *event time*. An *event centroid vector* is defined similarly to the cluster centroid vector, i.e., as normalized tf–idf values of the terms in the event's bursty tweets. In order to identify the best tweet for the event, we measure the similarities between the event centroid vector and the vectors of the event's bursty tweets. The tweet with the highest cosine similarity is designated as the *best tweet* to describe the event. Once an event is detected in a cluster, the tweets collected in that cluster for a specific duration after the detection of the event are considered to be mentioning that event.

4 Evaluation

We performed the proposed event detection method on tweets posted in Turkey between April 1 and April 10, 2014, collected by our streaming client. This tweet set is composed of 2,373,492 tweets posted by 249,168 distinct Twitter users. Using tweets in Turkey helped us observe various writing styles due to language-specific characters in an alphabet (e.g., the letters *ı, ö, ü, ç, ş, ğ* in Turkish alphabet are often typed as *i, o, u, c, s, g*, respectively, in tweets).

Table 1 presents the values for the constants and thresholds used in our implementation. Our evaluation focuses on the implications of the proposed vector expansion in terms of two aspects of event detection, namely clustering accuracy and event detection accuracy. Clustering accuracy refers to the precision and recall for the tweets in generated clusters. The analysis of event detection accuracy addresses the detection of events with minimum misses and false alarms. In Sect. 4.1, we describe how we prepare the ground truth for evaluation. Then the evaluations for clustering and event detection are presented in Sects. 4.2 and 4.3, respectively. We also discuss the threshold values in our implementation in Sect. 4.4 and present example clusters in Sect. 4.5.

4.1 Ground truth annotation

As a reference annotation, we select and annotate a set of target events and their associated target tweets in our tweet set. In order to minimize the effect of human interpretation while making these annotations, we devise a partially automated method based on the keyword-based tweet collection approach developed by Sakaki et al. (2013). We select earthquakes and goals in soccer games as two target topics in our evaluation for the following reasons: (1) they are real world events with specific time and place that can be described as ground truth, (2) the relevance of their tweets are often clear, and (3) their tweets are easier to select using keywords. Football is followed by millions of people in Turkey, and thus we could obtain a rich set of event tweets. Earthquakes also frequently happen and detecting them could be useful for situation awareness.

Considering earthquakes and goals as two specific topics, we first determine topic-specific keywords that best describe these topics. For earthquakes, we set *deprem* and *salla* (Turkish equivalents of *earthquake* and *shake*, respectively) as the topic-specific keywords. Goals are described by the terms *gol*, *gool*, and *goal* (*gol* is the Turkish spelling for *goal*). Ground truth annotation is carried out in three steps. In the first step, we count the tweets that contain topic-specific keywords in each bucket.

Table 1 Constants and thresholds in experiments

Name	Value
Minimum bucket size	200 tweets
Cluster maximum idle time	3 buckets
Number of buckets in history h	5
Merge threshold in concept decomposition	0.5
Term similarity threshold	0.5
Term similarity count	10

For the buckets with a noticeable number of such tweets, we search the news sources to learn about the details of these events. This helps us extract event-specific keywords and determine the event times. Event-specific keywords for an earthquake include the names of the city and town of the earthquake's epicenter. For a goal event, event-specific keywords include the names of the competing clubs, the name of the scoring player, and the new score in the game. The result of the first step is the list of target events with their times and descriptive keywords. In the second step of the annotation, a second pass is made on the tweets, in which a tweet is marked as a target tweet if it contains either a topic-specific or an event-specific keyword, and these tweets are assigned to the corresponding target events. The third and final step is the human controlled elimination of false positives in target tweets, with the same purpose of the classifier presented in Sakaki et al. (2013). As time passes after an event, the tweets can turn out to be *about news of the event*, rather than being *about the event* (Crooks et al. 2013), which makes it harder to decide the relevance of tweets with the event. Therefore, we performed our evaluation on the tweets posted within 10 min of the event time. As a result, by discarding minor events with few tweets, we obtained 3 earthquakes and 28 goals as the target events comprising 4783 tweets in our tweet set.

For example, one of these events was an earthquake in Bodrum, a town of Muğla, around the villages of Gümüşlük and Gümbet on April 6, 2014. It was a minor earthquake with a reported magnitude of 3.8 M_L. In addition to the topic-specific terms, we used the place names related to this event for annotation (e.g., *bodrum*, *muğla*, *mugla*, *gümüşlük*, *gumusluk*, *gümbet*, *gumbet*) and identified 34 tweets as our ground truth. An example goal event in our dataset is the goal of Cristiano Ronaldo in UEFA Champions League quarter-finals on April 2, 2014, making the score 3–0 for Real Madrid against Borussia Dortmund. Event-specific keywords that we used for annotation include *cristiano*, *ronaldo*, *real*, *madrid*, *halamadrid*, *dortmund*, *bvb*, *3–0*. As a result, we identified 36 tweets posted in Turkey related to this goal event. Another example for a goal event is the goal scored by Oğuzhan Özyakup in the game between Beşiktaş and Kayserispor on April 5, 2014. Since these teams are two major clubs highly followed in Turkey, the number of tweets annotated for this event is relatively higher than the others, specifically 248. Event-specific terms that we used for annotation are the name of the player, the names of the teams, and the score, which include *oğuzhan*, *oguzhan*, *özyakup*, *ozyakup*, *beşiktaş*, *besiktas*, *kayserispor*, *2–0*.

4.2 Clustering accuracy

In this section, we aim to explore whether tweets are correctly grouped in coherent clusters. Without considering burst detection, we focus on the accuracy of tweets in clusters and employ precision–recall analysis of the clustered tweets for evaluation (Li et al. 2012; Yang et al. 1998). Our intuition is that the cluster about an event should group together as many event-related tweets as possible, and with the minimum number of irrelevant tweets (false positives). For this analysis, we use the target events and their tweets that were determined as ground truths above. This requires matching each target event to one of the generated clusters. To this end, we define a matching strategy. For each target event e with the set of target tweets T_e, the cluster c with the highest number of tweets in T_e is designated as the best matching cluster for e. Let T_c represent the set of tweets in cluster c. Then, the precision and recall values for c are found as given in Eq. (2), and the overall accuracy for c is measured using the F_1 score in Eq. (3).

$$p(c) = \frac{|T_c \cap T_e|}{|T_c|} \qquad r(c) = \frac{|T_c \cap T_e|}{|T_e|} \tag{2}$$

$$F_1(c) = \frac{2 \times p(c) \times r(c)}{p(c) + r(c)} \tag{3}$$

This strategy can match multiple target events to the same cluster. In this case, union of the target tweet sets can be used in the analysis. For example, if e_1 and e_2 are matched with c, then T_e is substituted by $T_{e_1} \cup T_{e_2}$ in precision–recall calculations. We discuss this case on an example in Sect. 4.5. After each target event is matched with a cluster and the clusters are analyzed for their accuracy, we employ a macro-average approach and find the average values of the cluster accuracy scores to measure the overall performance of a clustering algorithm (Yang et al. 1998).

As a comparable baseline clustering method, we use the incremental clustering presented in Yin et al. (2012), because of its online processing of the tweet stream and applicability in open domain. Due to several implementation details not being clarified in that work (e.g., the cluster

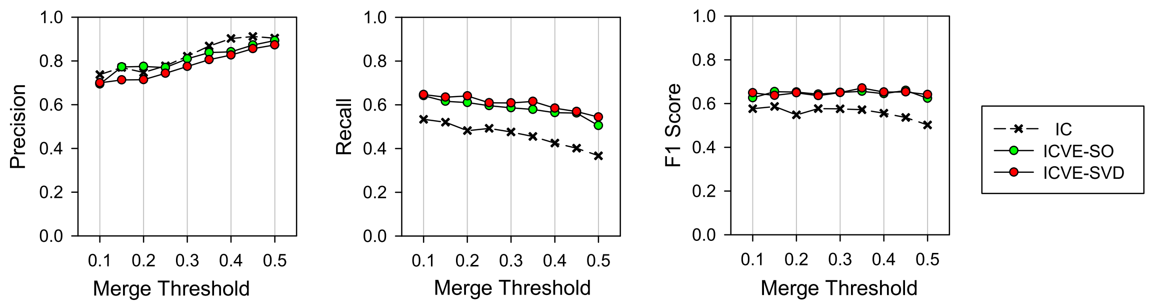

Fig. 2 Clustering accuracy for IC, ICVE-SO, and ICVE-SVD using different merge thresholds

termination condition, vocabulary management), we referred to other studies in the literature (Aggarwal and Zhai 2012; Atefeh and Khreich 2015; Sankaranarayanan et al. 2009; Yang et al. 1998) and implemented the clustering algorithm given in Sect. 3.2. We call this baseline algorithm IC. Our enhancement based on online similarity analysis and vector expansion builds on IC to culminate in ICVE. We use two methods to generate co-occurrence vectors as presented in Sect. 3.3.2. We call the enhanced clustering algorithms ICVE-SO and ICVE-SVD in accordance with the names of the co-occurrence vector generation methods.

The accuracy of the incremental clustering is sensitive to the merge threshold parameter. Since no specific merge threshold value was given in Yin et al. (2012), we analyzed the performance of the clustering algorithms under different merge thresholds ranging between 0.1 and 0.5. The results of the precision, recall, and F_1 score analysis are presented in Fig. 2. The graphs show that precision is usually similar in all algorithms. The slight decrease in precision by ICVE methods can be explained by possible false positives while finding relevant tweets using vector expansion. However, recall in both variations of ICVE is remarkably higher than the baseline. We imagine that vector expansion techniques would be most useful as a search aid for reporters who do not want to miss tweets about recent events. The improvement achieved by vector expansion is also remarkable on the F_1 scores. This analysis shows that both variations of ICVE outperform the baseline algorithm IC. It is also notable that regardless of the choice of the merge threshold, even the minimum F_1 scores obtained by ICVE-SO and ICVE-SVD are still higher than the maximum F_1 score that can be achieved by IC.

Another observation is that while high merge threshold values have a positive effect on precision, they cause tiny fragmented clusters and reduced recall. More importantly, fragmented clusters may not survive for a long time. Examples to the clusters with high threshold and low threshold will be given in Sect. 4.5. In order to determine the best merge threshold for an algorithm, we require all target events to be matched with a cluster with the highest precision possible. This condition was satisfied using 0.25 as the merge threshold for IC and ICVE-SVD. For ICVE-SO, the merge threshold to satisfy this condition was found to be 0.30. It is noteworthy that even with a higher merge threshold, ICVE-SO yields a higher recall than IC does.

As discussed in Sect. 3.3, the proposed vector expansion process introduces additional computational load on online clustering. Between the two ICVE methods, ICVE-SO is more efficient than ICVE-SVD in terms of computational cost. This is indicated by the time it takes to process a bucket. The total number of buckets in our tweet set is 8187, with 2469 being the maximum number of tweets in a bucket. We performed our evaluation on a personal computer with a 3.4 Ghz Intel Core i5 processor and 8 GB of DDR3 memory. The average and maximum execution times of a bucket are presented in Fig. 3a and b, respectively. These figures show that the enhancement introduced in ICVE-SO does not incur any remarkable cost in relation to the online processing of the tweet stream. On the other hand, ICVE-SVD may take much longer time to process a bucket, which can reach 470 s as shown in Fig. 3b. Since efficient and accurate clustering was achieved by ICVE-SO, for the remainder of our evaluation, we compare IC with ICVE-SO under their best merge thresholds (i.e., 0.25 and 0.30, respectively).

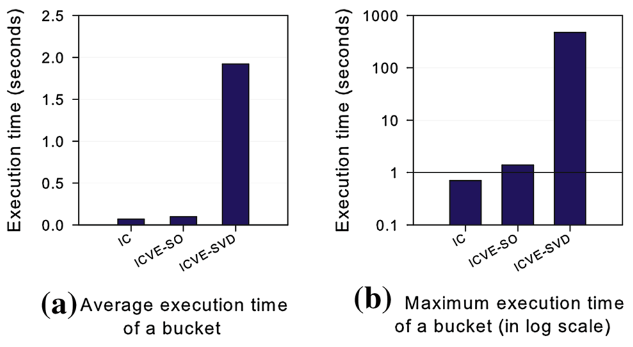

Fig. 3 Execution times of clustering algorithms. **a** Average execution time of a bucket, **b** maximum execution time of a bucket (in log scale)

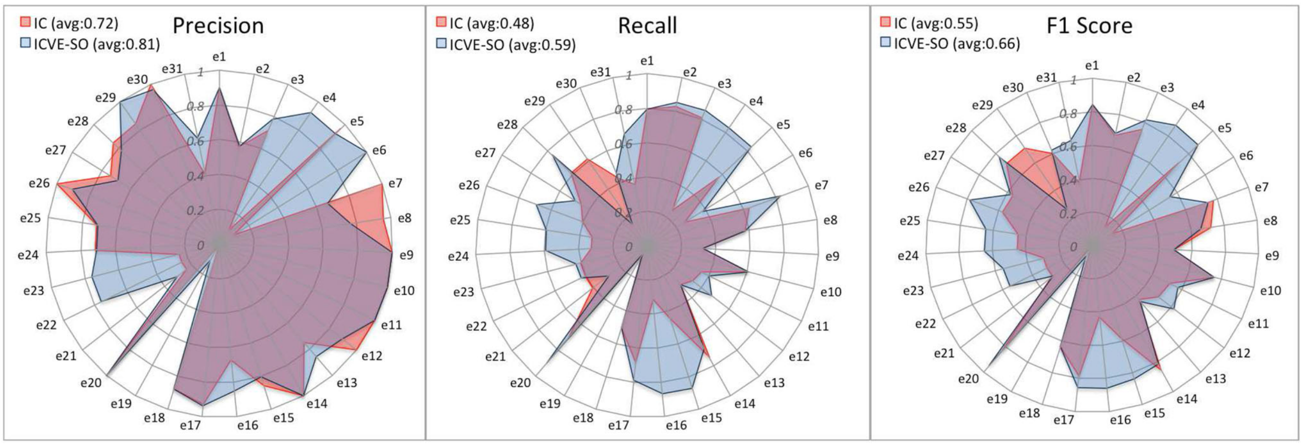

Fig. 4 Precision, recall, and F_1 scores for each event using IC and ICVE-SO algorithms

We would like to note that in these experiments we used tweets that we stored in a database and executed our algorithms under different settings in a single-thread process. In a real use-case scenario, the application can perform tweet collection and analysis tasks in two separate threads in parallel. In other words, while one thread collects tweets in a new bucket, another thread can process tweets in the previous bucket for clustering and event detection, as we have shown in the process flow in Fig. 1. The execution times in Fig. 3 show that our hybrid bucket size setting gives sufficient time for these analyses.

In addition to the analysis of macro-average accuracy (Fig. 2), we evaluate the precision, recall, and F_1 scores for each of the 31 target events and present the results in Fig. 4. The figure shows that for most target events, accuracy is higher with ICVE-SO. The significance of the improvement is analyzed using an unpaired *t* test. For both recall and F_1 scores, the *t*-test score yields a value lower than 0.050 (0.041 and 0.046, respectively), which indicates that the improvement is statistically significant.

We also analyzed the performance of IC and ICVE-SO by comparing their Adjusted Random Index (ARI) score using Scikit library.[6] We constructed the true labels array for annotated tweets using their events in the ground truth. Cluster ids for these tweets that are found by IC and ICVE-SO are used as the predicted labels by these algorithms. ARI scores for the assignments of IC and ICVE-SO are calculated as 0.32 and 0.45, respectively. The higher score for ICVE-SO indicates a better match for the ground truth assignments, and thus an improvement over the baseline IC method.

[6] http://scikit-learn.org/stable/modules/generated/sklearn.metrics.adjusted_rand_score.html.

4.3 Event detection accuracy

In this section, we analyze the results of the burst detection method presented in Sect. 3.4 when applied with different clustering methods, namely IC and ICVE-SO. In these analyses, we use the same tweet set with 31 ground truth events explained in Sect. 4.1. In our evaluations, we focus on the ground truth events by performing precision–recall analysis based on false alarms and missed target events.

For this evaluation, we need to determine which detected event corresponds to which target event in the ground truth. Therefore, we check the detected events within two buckets after the time of a target event. If the terms that describe the detected and target events have a match, then the target event is marked as *detected*. Otherwise, the target event is considered a *miss*. On the other hand, there can be multiple events detected for a target event, which are counted as *false alarms*. This would usually happen in case of fragmented clusters about an event, where bursty terms about the same event are dispersed in multiple clusters. It seems that with ICVE, fragmented clusters are less likely to form.

Since the accuracy of the burst detection algorithm depends on the burst ratio threshold Δ, we experiment with several Δ values on the same dataset. Figure 5 presents the results of the event detection accuracy analysis. These results show that the miss rates obtained by using IC and ICVE-SO are usually close to each other. On the other hand, for all Δ values in our experiments, using ICVE-SO significantly reduces the number of false alarms.

We present two examples to highlight the cases regarding the false alarms. The examples are selected from the burst detection results obtained using $\Delta = 9$. The first example in Fig. 6 shows the histograms for the frequency of term *gol* in the tweet stream and three clusters. In the figure, the histogram (A) represents the number of tweets

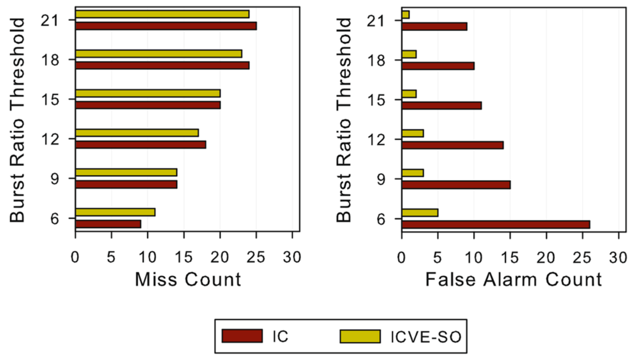

Fig. 5 Event detection accuracy analysis on annotated events

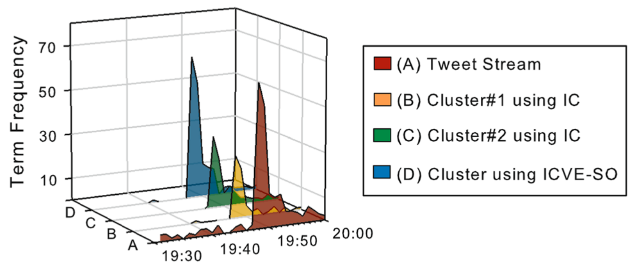

Fig. 6 Distribution of the term *gol* in tweet stream and clusters before and after the goal event

including the term *gol* in each bucket received from the tweet stream in 30 min. The goal event happens at around 19:45, which causes the burst in that histogram. The histograms (B) and (C) are for two event clusters associated with that goal event generated by the IC clustering method. One of these clusters has a centroid vector with a large weight for the term *gol*. The centroid vector of the second cluster has most of the weight on the name of the player. As a result of this fragmentation, the tweets that contain the term *gol* are grouped in two separate clusters, resulting in two separate alarms for the same event, one of which can be considered a false alarm. On the other hand, the event detection method using ICVE-SO generates one event cluster with histogram (D) in Fig. 6. The centroid vector of this cluster has close weights for the term *gol* and the name of the player. The similarity between histogram (A) and histogram (D) shows that almost all tweets containing the term *gol* could accurately be collected in a single cluster, resulting in the detection of the target event with no false alarms.

The second example is about the effect of cluster fragmentation on the results of event detection due to spelling variances. The target event in the example is a goal scored by a player named *oğuzhan*. It is observed that some Twitter users type the letter “g” instead of the letter “ğ” in the Turkish alphabet. Executing event detection using IC yields four separate event clusters about this event, with the centroid vectors mostly focusing on four different terms, namely *oguzhan*, *oğuzhan*, *gol*, and *ozi*. On the other hand, using ICVE-SO, we observe that a single event is detected by gathering relevant tweets in the same cluster with a centroid vector that has proportional weights for these event-related terms. This example suggests that ICVE-SO can successfully identify the differences in typing and group similar tweets about an event in the same cluster, resulting in increase in event detection accuracy.

4.4 Settings for constants and thresholds

In this section, we discuss the results of selected values for the constants and thresholds given in Table 1. In our hybrid bucket size model, we determined the time window for a bucket as 1 min in order to detect bursts even if they occur for a short period of time. Considering the event types in our experiments, namely earthquakes and goals, analysis of clusters every 1 min provides a near real-time detection without much delay. Accordingly, minimum number of tweets in the hybrid bucket model is determined as 200, since it is approximately the average number of tweets per minute that we received from the tweet stream.

Cluster maximum idle time in Table 1 is used to find stale clusters in online execution, as explained in Sect. 3.2. Setting its value to three means that we terminate a cluster if no tweet has been added to it for three consecutive buckets. In order to see its effect on the clustering accuracy, we run the baseline IC and the proposed ICVE-SO algorithms with different values. The precision, recall, and F_1 score that we obtained in these experiments are presented in Fig. 7. The merge threshold values that we used in these experiments are the best merge thresholds we found before (i.e., 0.25 for IC and ICVE-SVD, 0.30 for ICVE-SO).

The results show that smaller values for the threshold yield higher precision on our test data. This is reasonable since buckets closer to the event time would usually include a higher number of tweets about that event. However, setting the cluster maximum idle time to a very low value such as one or two buckets could be restrictive against minor events mentioned in a few tweets. On the other hand, high values for this threshold would require maintaining more clusters in the memory for online execution. Moreover, increasing its value does not necessarily yield higher F_1 scores, as shown in Fig. 7. Accordingly, we used three buckets as the maximum idle time for a cluster since it yields high accuracy and it gives reasonable time for a cluster to stay fresh. Finally, we would like to point out that the results in Fig. 7 underline the superiority of ICVE-SO over the baseline clustering method for all selected threshold values.

Another constant in Table 1 is *h*, the number of buckets in the history used for burst detection. We keep five

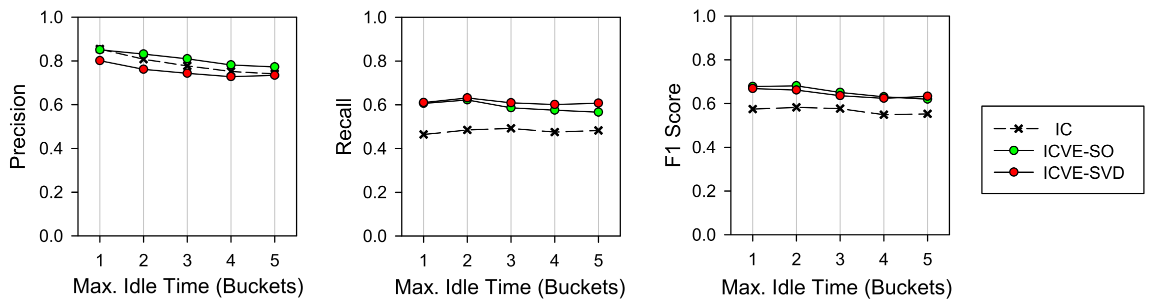

Fig. 7 Clustering accuracy for IC, ICVE-SO, and ICVE-SVD using different values for cluster maximum idle time

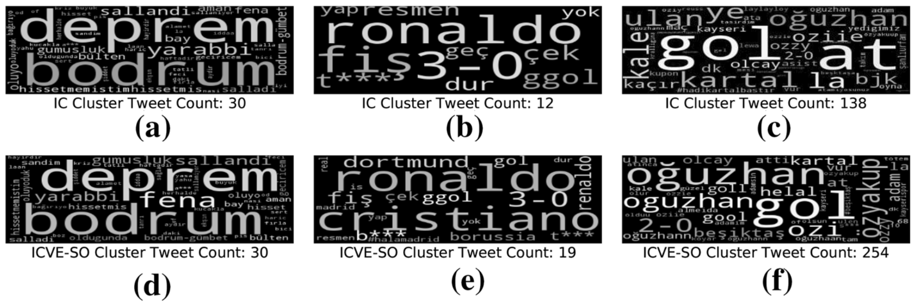
IC Cluster Tweet Count: 30
(a)
IC Cluster Tweet Count: 12
(b)
IC Cluster Tweet Count: 138
(c)
ICVE-SO Cluster Tweet Count: 30
(d)
ICVE-SO Cluster Tweet Count: 19
(e)
ICVE-SO Cluster Tweet Count: 254
(f)

Fig. 8 Comparison of sample clusters found by the baseline IC (top) and the proposed ICVE-SO (bottom) for three example events

buckets in the history in order to calculate the historical statistics for terms effectively for online processing. Other values in Table 1 are used in the similarity analysis of terms for vector expansion. A high merge threshold in concept decomposition is selected in order to group tweets in coherent clusters. Similarly, the values for the term similarity threshold and term similarity count are determined to prevent the search space to grow unwieldy and compare only distinctive terms in a bucket. These values are selected by intuition, and their adaptive selection is considered as future work.

4.5 Sample clusters and discussion

In this section, we present sample clusters generated by IC and ICVE-SO in order to exemplify the improvement that we achieved by using vector expansion. We use the example events in the ground truth that we explained in Sect. 4.1, i.e., one earthquake and two goal events. The three pairs of cluster summaries in Fig. 8 show the terms with highest tf–idf scores in cluster centroids that we obtained by IC and ICVE-SO for each of the three events. The word clouds are generated by the Python word cloud library[7] using tf–idf scores in cluster centroids (we replaced parts of offensive words by *** in the figures).

Figure 8a and d represent clusters generated by IC and ICVE-SO, respectively, for the earthquake. These figures show that both methods resulted in similar clusters for this event, having the highest tf–idf values for *deprem(earthquake)* and *bodrum*. The improvement with ICVE-SO is more remarkable when we compare Fig. 8b with e, the clusters generated for the first goal event. The cluster found by IC in 8b has 12 tweets, and it has the highest tf–idf values for *ronaldo* and *3–0*. The cluster generated by ICVE-SO in 8e has 19 tweets. It grouped more tweets about the event, yielding a centroid vector that also shows *cristiano* and *dortmund*. Similarly, for the third example, the cluster generated by ICVE-SO in Fig. 8f contains more tweets than the cluster generated by IC in 8c. In addition to the increase in accuracy, the terms in the cluster centroid for ICVE-SO noticeably have a higher weight for terms that describe the event. Terms such as *2–0*, *oguzhan*, *özyakup*, and *beşiktaş* are more noticeable in the word cloud in Fig. 8f.

As we explained in Sect. 4.2, multiple target events in the ground truth can be matched to the same cluster. In this case, union of the target tweet sets is used in accuracy analysis. We exemplify this case using two goal events in our ground truth that occurred within 3 mins in the same game, which was played between the teams of Sivas and Bursa. The first goal, scored by *Burhan Eşer*, has 33 tweets in the ground truth, and the second goal, scored by *José Fernando*, has 98 annotated tweets in our dataset. As a result of running IC on our dataset, we observe that the cluster that includes the highest number of tweets for the first goal has 8 tweets related to this event (call this cluster

[7] https://github.com/amueller/word_cloud.

c_x). Other tweets about this event are scattered in several other clusters in smaller amounts. We observe that c_x is also the same cluster that includes the highest number of tweets about the second goal. Specifically, c_x includes 35 tweets related to the second goal event. The reason is presumably that the second event happens soon after the other, and they happen in the same game, thus having similar terms in tweets. As a result, c_x is composed of 60 tweets in total, including the ones related to these two goals. According to these results, while calculating the precision and recall for c_x, we use $|T_e| = 33 + 98 = 131$, $|T_{c_x}| = 60$, and $|T_{c_x} \cap T_e| = 8 + 35 = 43$. Using these values in Eqs. (2) and (3), we obtain 0.72 for precision, 0.33 for recall, and 0.45 for the F_1 score of c_x generated by the baseline IC.

Based on the above example, we also analyze the result of ICVE-SO for these two events. ICVE-SO also finds one cluster (call c_y) having the highest number of tweets for these two events. c_y has totally 110 tweets, 10 of which are related to the first goal and 68 of which are related to the second goal. Thus, $|T_{c_y}| = 110$ and $|T_{c_y} \cap T_e| = 10 + 68 = 78$. As a result, its accuracy analysis yields 0.71 for its precision, 0.60 for recall, and 0.65 for F_1 score. The slight decrease in precision compared to the precision of c_x means that c_y has several other tweets that are not related to these events (false positives). However, tweets related to these two events are remarkably higher in c_y, yielding a remarkable increase in recall and overall F_1 score.

We noted in Sect. 4.2 that high merge threshold in clustering usually has a positive effect on precision, whereas it can cause tiny fragmented clusters and reduced recall. We exemplify the effect of merge threshold on an example event, namely the earthquake in Bodrum. We would like to remind that this event has 34 tweets in the ground truth. The lowest and highest merge thresholds that we experimented in our analyses were 0.1 and 0.5, respectively. Accordingly, we executed the baseline IC clustering on our dataset using these thresholds and observed its effect on the generated clusters for this earthquake. Figure 9a shows the cluster that includes the highest number of tweets about this event generated by the low threshold. Our detailed analyses revealed that 82% of annotated tweets were successfully collected in this cluster. Other tweets that are not included in this cluster mostly started new clusters by themselves and did not survive for a long time. It is noteworthy that the cluster summary in Fig. 9a displays many relevant terms about this event. Figure 9b and c display the result of using a high threshold value of 0.5. We show the two largest clusters that include event-related tweets. The cluster in Fig. 9b is apparently grouped around a centroid with terms such as *deprem (earthquake)*, *salla (shake)*, *ekşın (action)* and terms that indicate panic or emotions. Figure 9c, on the other hand, has more weight for *bodrum*, i.e., the town for the earthquake's epicenter. Because of the high merge threshold, most of the event-related tweets could not be collected in a single cluster. This resulted in more fragmented clusters with lower recall. On the other hand, high merge threshold made it less likely to collect false positives in these clusters, which improved their precision.

In accordance with the definition of the event, example cases in our evaluation dataset are related to things that happened at a specific time and place, and they were discussed relatively in a short period of time in Twitter. In order to observe the effect of vector expansion in different situations, we also performed further experiments with tweets related to a popular TV show in Turkey, named *Beyaz Show*, which is a media event that lasts for a longer period of time. We collected and processed public tweets posted in Turkey for 1 h before and after the start of the show. We annotated tweets posted within 1 h after the show started using the same keyword-based annotation method. Specifically, the list of keywords we used for annotation are the terms that appear in name of the show (e.g., *beyazshow*, *beyaz*, *show*), the name of host (e.g., *beyazıt*, *beyazit*, *öztürk*, *ozturk*), and the names of the guests (e.g., *zara*, *hayko*, *cepkin*). We executed IC and ICVE-SO clustering on this dataset using the same constants and thresholds stated above. The cluster summaries along with the calculated accuracy values are given in Fig. 10. Both clusters have the highest tf–idf score for the name of a guest (*hayko*) in the show. Despite a slight decrease in precision, the cluster in Fig. 10b that we obtained by ICVE-SO has remarkably higher recall than the cluster in Fig. 10a. As a result, the cluster found by ICVE-SO includes other important terms about this TV show, such as *#beyazshow* and *zara*.

5 Conclusion

In this work, we propose Incremental Clustering with Vector Expansion (ICVE) method, which is developed in order to improve the online clustering and event detection accuracy in microblogs. We demonstrated that our vector expansion process based on unsupervised temporal analysis of term similarities in tweets facilitates collecting relevant tweets in the same cluster even if they do not share common terms. The evaluations performed using a variety of threshold values illustrate that ICVE-SO yields statistically significant improvements in clustering accuracy and reduces the false alarm counts for events when compared with the results obtained by the baseline incremental clustering algorithm. The proposed enhancement does not incur any remarkable cost that would hinder online processing of the

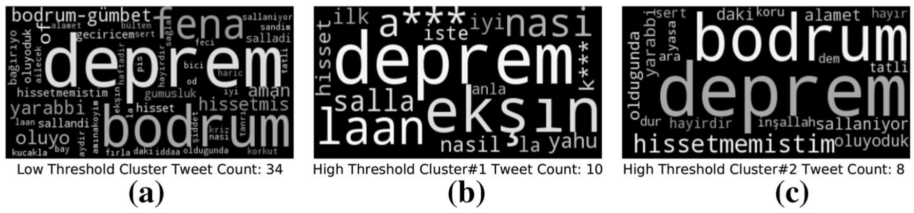
Low Threshold Cluster Tweet Count: 34
(a)
High Threshold Cluster#1 Tweet Count: 10
(b)
High Threshold Cluster#2 Tweet Count: 8
(c)

Fig. 9 Sample cluster summaries found by different merge thresholds. **a** With low threshold: 0.1, **b** and **c** with high threshold: 0.5

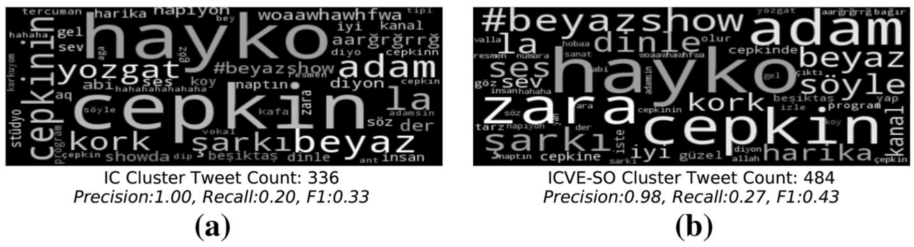
IC Cluster Tweet Count: 336
Precision:1.00, Recall:0.20, F1:0.33
(a)
ICVE-SO Cluster Tweet Count: 484
Precision:0.98, Recall:0.27, F1:0.43
(b)

Fig. 10 Clusters with highest tweet counts found by IC (**a**) and ICVE-SO (**b**) for TV show

stream. We believe this enhancement is also applicable to other implementations of incremental clustering, for example, those using different similarity metrics or different thresholds. Accurately grouping similar tweets in coherent clusters is significant not only for the event detection purposes, but also for performing further analyses on tweets, such as extraction of user profiles and location analysis for events.

In our future work, we plan to investigate the adaptive selection of the parameter values in the algorithms. For even better performance, adopting a language-specific perspective and using natural language processing techniques for similarity analysis seem to have potential. In addition, exploring the relationship between the detected events and building a hierarchical view to present these aspects at various abstraction levels would be beneficial to end users. Further analysis on the benefit of vector expansion on a wider range of events is also considered as future work.

Acknowledgements This work was financially supported by TUBITAK with the Grant number 112E275 and ICT COST Action IC1203.

References

Aggarwal C, Zhai C (2012) A survey of text clustering algorithms. In: Aggarwal CC, Zhai C (eds) Mining text data. Springer, New York, pp 77–128

Aggarwal CC, Subbian K (2012) Event detection in social streams. In: SDM. SIAM/Omnipress, pp 624–635

Aggarwal CC, Yu PS (2006) A framework for clustering massive text and categorical data streams. In: Ghosh J, Lambert D, Skillicorn DB, Srivastava J (eds) SDM. SIAM, Philadelphia, pp 479–483

Aggarwal CC, Han J, Wang J, Yu PS (2003) A framework for clustering evolving data streams. In: Proceedings of the 29th international conference on very large data bases—volume 29, VLDB Endowment, VLDB '03, pp 81–92

Agirre E, Alfonseca E, Hall K, Kravalova J, Paşca M, Soroa A (2009) A study on similarity and relatedness using distributional and wordnet-based approaches. In: Proceedings of human language technologies: the 2009 annual conference of the North American chapter of the association for computational linguistics, Association for Computational Linguistics, Stroudsburg, NAACL'09, pp 19–27

Allan J (ed) (2002) Topic detection and tracking: event-based information organization. Kluwer Academic Publishers

Atefeh F, Khreich W (2015) A survey of techniques for event detection in Twitter. Comput Intell 31(1):132–164

Bansal N, Koudas N (2007) Blogscope: a system for online analysis of high volume text streams. In: Proceedings of the 33rd international conference on very large data bases, VLDB Endowment, VLDB'07, pp 1410–1413

Berry MW, Dumais ST, O'Brien GW (1995) Using linear algebra for intelligent information retrieval. SIAM Rev 37(4):573–595. doi:10.1137/1037127

Cao G, Nie JY, Gao J, Robertson S (2008) Selecting good expansion terms for pseudo-relevance feedback. In: Proceedings of the 31st annual international ACM SIGIR conference on research and development in information retrieval, SIGIR'08, pp 243–250

Chen L, Chun L, Ziyu L, Quan Z (2013) Hybrid pseudo-relevance feedback for microblog retrieval. J Inf Sci 39(6):773–788

Cheong M, Lee VCS (2011) A microblogging-based approach to terrorism informatics: Exploration and chronicling civilian sentiment and response to terrorism events via Twitter. Inf Syst Front 13(1):45–59

Cordeiro M, Gama J (2016) Online social networks event detection: A survey. In: Michaelis S, Piatkowski N, Stolpe M (eds) Solving Large Scale Learning Tasks. Challenges and Algorithms. Lecture Notes in Computer Science, vol 9580. Springer, Cham, pp 1–41

Cotelo JM, Cruz FL, Troyano JA, Ortega FJ (2015) A modular approach for lexical normalization applied to spanish tweets. Expert Syst Appl 42(10):4743–4754

Cotelo JM, Cruz FL, Troyano JA (2014) Dynamic topic-related tweet retrieval. J Assoc Inf Sci Technol 65(3):513–523

Crooks A, Croitoru A, Stefanidis A, Radzikowski J (2013) #Earthquake: Twitter as a distributed sensor system. Trans GIS 17(1):124–147

De Choudhury M, Sundaram H, John A, Seligmann DD (2008) Can blog communication dynamics be correlated with stock market activity? In: Proceedings of the nineteenth ACM conference on hypertext and hypermedia, HT'08, pp 55–60

Deerwester S, Dumais ST, Furnas GW, Landauer TK, Harshman R (1990) Indexing by latent semantic analysis. J Am Soc Inf Sci 41(6):391–407

Fang Y, Zhang H, Ye Y, Li X (2014) Detecting hot topics from Twitter: A multiview approach. J Inf Sci 40(5):578–593

Fung GPC, Yu JX, Yu PS, Lu H (2005) Parameter free bursty events detection in text streams. In: Proceedings of the 31st international conference on very large data bases, VLDB Endowment, VLDB'05, pp 181–192

Imran M, Castillo C, Diaz F, Vieweg S (2015) Processing social media messages in mass emergency: a survey. ACM Comput Surv 47(4):67:1–67:38

Jun S, Park SS, Jang DS (2014) Document clustering method using dimension reduction and support vector clustering to overcome sparseness. Expert Syst Appl 41(7):3204–3212

Kaufmann M, Kalita J (2010) Syntactic normalization of Twitter messages. In: International conference on natural language processing, Kharagpur

Kim D, Kim D, Rho S, Hwang E (2013) Detecting trend and bursty keywords using characteristics of Twitter stream data. Int J Smart Home 7(1):209–220

Kleinberg J (2002) Bursty and hierarchical structure in streams. In: Proceedings of the eighth ACM SIGKDD international conference on knowledge discovery and data mining, KDD'02, pp 91–101

Li C, Sun A, Datta A (2012) Twevent: segment-based event detection from tweets. In: Proceedings of the 21st ACM international conference on information and knowledge management, CIKM'12, pp 155–164

Lin D, Zhao S, Qin L, Zhou M (2003) Identifying synonyms among distributionally similar words. In: Proceedings of the 18th international joint conference on artificial intelligence, IJCAI'03, pp 1492–1493

Magdy W, Elsayed T (2016) Unsupervised adaptive microblog filtering for broad dynamic topics. Inf Process Manage 52(4):513–528

Marcus A, Bernstein MS, Badar O, Karger DR, Madden S, Miller RC (2011) Twitinfo: aggregating and visualizing microblogs for event exploration. In: Proceedings of the SIGCHI conference on human factors in computing systems, CHI'11, pp 227–236

Nguyen D, Jung J (2015) Real-time event detection on social data stream. Mob Netw Appl 20(4):475–486

Okazaki M, Matsuo Y (2010) Semantic Twitter: analyzing tweets for real-time event notification. In: Breslin J, Burg T, Kim HG, Raftery T, Schmidt JH (eds) Recent trends and developments in social software, lecture notes in computer science, vol 6045. Springer, Berlin, pp 63–74

Ozdikis O, Senkul P, Oguztuzun H (2012a) Semantic expansion of hashtags for enhanced event detection in Twitter. In: Proceedings of VLDB 2012 Workshop on Online Social Systems (WOSS)

Ozdikis O, Senkul P, Oguztuzun H (2012b) Semantic expansion of tweet contents for enhanced event detection in Twitter. In: IEEE/ACM international conference on Advances in Social Networks Analysis and Mining (ASONAM), pp 20–24

Ozdikis O, Senkul P, Oguztuzun H (2014) Context based semantic relations in tweets. In: Can F, Özyer T, Polat F (eds) State of the art applications of social network analysis, lecture notes in social networks. Springer International Publishing, pp 35–52

Phuvipadawat S, Murata T (2010) Breaking news detection and tracking in Twitter. In: IEEE/WIC/ACM international conference on Web Intelligence and Intelligent Agent Technology (WI-IAT), vol 3. pp 120–123

Qiu Y, Frei HP (1993) Concept based query expansion. In: Proceedings of the 16th annual international ACM SIGIR conference on research and development in information retrieval. SIGIR'93, pp 160–169

Rapp R (2002) The computation of word associations: comparing syntagmatic and paradigmatic approaches. In: Proceedings of the 19th international conference on computational linguistics—volume 1, Association for Computational Linguistics, Stroudsburg, COLING'02, pp 1–7

Sakaki T, Okazaki M, Matsuo Y (2013) Tweet analysis for real-time event detection and earthquake reporting system development. IEEE Trans Knowl Data Eng 25(4):919–931

Sankaranarayanan J, Samet H, Teitler BE, Lieberman MD, Sperling J (2009) TwitterStand: News in tweets. In: Proceedings of the 17th ACM SIGSPATIAL international conference on advances in geographic information systems. GIS'09, pp 42–51

Shou L, Wang Z, Chen K, Chen G (2013) Sumblr: Continuous summarization of evolving tweet streams. In: Proceedings of the 36th international ACM SIGIR conference on research and development in information retrieval, SIGIR'13, pp 533–542

Silva JA, Faria ER, Barros RC, Hruschka ER, de Carvalho ACPLF, Gama J (2013) Data stream clustering: a survey. ACM Comput Surv 46(1):13:1–13:31

Song W, Park SC (2007) A novel document clustering model based on latent semantic analysis. In: Proceedings of the third international conference on Semantics, knowledge and grid, pp 539–542

Thomas A, Sindhu L (2015) A survey on content based semantic relations in tweets. Int J Comput Appl 132(11):14–18

Varga A, Basave AEC, Rowe M, Ciravegna F, He Y (2014) Linked knowledge sources for topic classification of microposts: a semantic graph-based approach. J Web Semant Sci Serv Agents World Wide Web 26:36–57

Voorhees EM (1994) Query expansion using lexical-semantic relations. In: Proceedings of the 17th annual international ACM SIGIR conference on research and development in information retrieval, SIGIR'94, pp 61–69

Weng J, Lee B (2011) Event detection in Twitter. In: Proceedings of the fifth international conference on weblogs and social media, ICWSM'11, pp 401-408

Xie W, Zhu F, Jiang J, Lim EP, Wang K (2013) TopicSketch: Real-time bursty topic detection from Twitter. In: IEEE 13th international conference on Data mining (ICDM), pp 837–846

Yang Y, Pierce T, Carbonell J (1998) A study of retrospective and on-line event detection. In: Proceedings of the 21st annual international ACM SIGIR conference on research and development in information retrieval, SIGIR'98, pp 28–36

Yin J, Lampert A, Cameron M, Robinson B, Power R (2012) Using social media to enhance emergency situation awareness. IEEE Intell Syst 27(6):52–59

Zhou Y, Kanhabua N, Cristea AI (2016) Real-time timeline summarisation for high-impact events in Twitter. In: 22nd European conference on artificial intelligence, ECAI'16, pp 1158–1166