doi:10.3772/j.issn.1006-6748.2018.02.013

A document-level model for tweet event detection①

Qin Yanxia (秦彦霞)*, Zhang Yue**, Zhang Min***, Zheng Dequan②*
(*School of Computer Science and Technology, Harbin Institute of Technology, Harbin 150001, P. R. China)
(**Information Systems Technology and Design, Singapore University of Technology and Design, Singapore 487372, Singapore)
(***School of Computer Science and Technology, Soochow University, Suzhou 215006, P. R. China)

Abstract

Social media like Twitter who serves as a novel news medium and has become increasingly popular since its establishment. Large scale first-hand user-generated tweets motivate automatic event detection on Twitter. Previous unsupervised approaches detected events by clustering words. These methods detect events using burstiness, which measures surging frequencies of words at certain time windows. However, event clusters represented by a set of individual words are difficult to understand. This issue is addressed by building a document-level event detection model that directly calculates the burstiness of tweets, leveraging distributed word representations for modeling semantic information, thereby avoiding sparsity. Results show that the document-level model not only offers event summaries that are directly human-readable, but also gives significantly improved accuracies compared to previous methods on unsupervised tweet event detection, which are based on words/segments.

Key words: social media, event detection, twitter, bursty, unsupervised, document-level

0 Introduction

Twitter has been recognized as an important source of news information, which can be faster than traditional media[1]. Large scale first-hand tweets motivate event detection on Twitter, which can be further used for tasks such as disaster event detection, event-based sentiment mining and stock analysis. Similar to Ref. [2], an event is defined as a group of features (e. g., words) obtained from a collection of tweets. Event detection based on tweets is challenging because tweets are in large scale and contain noise and mundane updates. While supervised learning approaches[3] can be useful for detection on specific types of events[4,5], open domain tweet event detection has been mostly addressed using unsupervised approaches.

The dominant approach for unsupervised event detection on tweets clusters similar words[6] or segments (n-grams)[2], detecting events based on burstiness of topics and difference from mundane posts. In particular, such feature-pivot methods select meaningful and informative features (e. g. words/segments) for representing tweets. Burstiness is calculated to select features that surge in frequency, which uses an essential source of information for detecting emerging events. However, one disadvantage of word and segment features is that they cannot fully convey event information. Manual links to tweets or search engines are necessary to recover the original event from word clusters.

In contrast, tweets are directly human-readable, and hence tweet clusters can be directly used as event summaries. However, little work in the literature has used tweets as features for unsupervised event detection. A major challenge is sparsity, which prevents tweets from being used for count-based features, which are used by word/segment-based methods. In addition, since tweet streams can be large, efficiency is necessary when mapping tweets into features. To address challenges above, the use of word embeddings for representing tweet semantic information, mapping each tweet into a low dimensional semantic space by averaging the embeddings of its words is empirically investigated. In contrast, traditional TF-IDF methods lead to vocabulary size dimensionality, which makes it intolerably slow to use for the task.

A two-stage model for event detection is built. First, tweets are clustered according to their semantic similarity using a density-based clustering algorithm

① Supported by the National High Technology Research and Development Programme of China (No. 2015AA015405).
② To whom correspondence should be addressed. E-mail: dqzheng2007@gmail.com
Received on Sep. 6, 2017

(DBSCAN[7]). Second, the resulting tweet clusters are further ranked using a newsworthiness heuristic, which considers burstiness of the tweet cluster, density of the cluster, number of related companies and distance to a few seed mundane posts and seed news.

The Document-level Burstiness model for Event Detection is called DBED. Results show that the document-level model not only gives directly readable summaries, but also outperforms word-level models. Compared with TF-IDF document representations, word embedding representations[8] are much more efficient due to the low-dimensionality of dense word representations. the code and data on https://github.com/qolina/DBED are released.

1 Related work

Both supervised and unsupervised methods have been used for event detection on Twitter. Supervised methods[10-12] are used mainly for specific-type event detection. Li, et al. [4] trained a classifier to determine whether a crawled tweet is crime and disaster related events. Ritter, et al. [5] extracted seed positive examples from tweets by defined keywords, and used them to train classifier for security tweet identification. Majority of event detection methods are unsupervised[13,14]. Chierichetti, et al. [15] observed that Twitter users like to tweet before or during an important event, and retweet after it. Identification of patterns is then applied to detect time of events. Kitagawa, et al. [16] predicted an epidemic of influenza through deep modality analysis of Twitter data. The work is in line with unsupervised methods.

Clustering-based methods include document-pivot clustering methods and feature-pivot clustering methods according to the processing unit. Document-pivot clustering methods[3,17,18] cluster tweets and select newsworthy clusters as events. Ifrim, et al. [18] applied token-level aggressive tweet filtering before fitting them in hierarchical clustering. The work is in line with two-stage document-pivot event detection methods. Existing methods are supervised. For example, Becker, et al. [3] used a SVM classifier with a set of cluster-level features to select event clusters. In contrast, the work uses an efficient and robust unsupervised method for cluster filtering. A novel document-level temporal feature (burstiness) is also proposed for distinguishing news events to constant mundane topics, using embeddings to overcome sparsity and efficiency challenges.

In contrast, feature-pivot methods[6,19-24] cluster a small number of bursty features (i. e. words), which indicate happening of events. In addition to improving clustering efficiency, detecting bursty features also perform an important task of feature selection, as social media are very noisy. A major disadvantage of feature-pivot clustering methods is hard-to-read output events. Inspired by feature selection through a word-level burstiness, it is proposed to extend word-level burstiness to document-level and thus used for clustering filtering.

2 Baseline methods

Two baseline methods are taken in this work, which both exploit temporal information for tweet event detection as the authors do. Different to the document-level temporal information, they use two traditional feature-based temporal information.

2.1 Segment-based baseline

Twevent[2] is a four-stage segment-based tweet event detection method, which includes segment-based tweet representation, feature (segment) selection, feature clustering and cluster filtering. Tweet Segmentation splits tweets into non-overlapping n-grams through an optimization process by considering Wikipedia and Microsoft Web Ngram Service as external resources. Segments that show bursty properties are selected as event-informative features, and then grouped into clusters. For cluster filtering in Twevent, a newsworthiness score is used to measure the probability of an event cluster being a news event rather than a mundane update or a constant topic. The newsworthiness of a cluster is evaluated through the following aspects: 1) average of all segments' newsworthiness in the cluster, which are the probabilities of segments being recognized as anchor texts in Wikipedia; and 2) density of the cluster, which is defined as average weight of edges between segments. Top ranking clusters by newsworthiness will be selected as event clusters.

2.2 N-gram-based baseline

Sensor[9] is a two stage event detection method, which consists of feature (n-grams) selection and feature clustering. It uses n-grams as features and applies a burstiness score calculated by *df- idf* for selecting important n-grams. The idea behind *df-idf* based burstiness, shown in Eq. (1), is that penalizing topics which began in the past and still popular in the present. In addition, the importance of named entities are boosted with a factor. N-grams are grouped into clusters with a hierarchical clustering algorithm. These clusters are directly ranked by the highest *df-idf* score of the n-grams contained in the cluster. And the top K_c

clusters are selected as output events.

$$df - idf = \frac{df_i + 1}{\log\left(\frac{1}{ltw}\sum_{j=i}^{ltw} df_{i-j} + 1\right) + 1} \quad (1)$$

where df_i represents an n-gram's document frequency in time window i, ltw is the number of history time windows, in which the n-gram's frequency is considered for estimating burstiness.

A comparison between two baseline methods and the proposed method is illustrated in Table 1. Both Twevent and Sensor consist of a burstiness-based feature selection before feature clustering and cluster filtering. However, Twevent and Sensor use different methods for burstiness calculation. Burstiness of Twevent is calculated as the difference between frequency in the current time window and the estimated average frequency in terms of standard deviation, while Sensor contrasts the frequency of the current time window with averaged history frequency without considering standard deviation. It is in line with Twevent by considering standard deviation. In addition, Twevent uses both history and future time window frequency for estimating burstiness, while Sensor only uses history information, which is more suitable for online event detection. The authors adopt online settings, using only history information.

For feature clustering, Twevent uses the k-Near-Neighbor algorithm and Sensor applies a hierarchical clustering method. A density-based clustering method is used for reasons explained in Section 3.3. For cluster filtering, Sensor takes the maximum burstiness of n-grams in a cluster directly as its newsworthiness score, while Twevent considers both newsworthiness and other features. They exploit Wikipedia as an external resource for calculating newsworthiness. A novel document-level burstiness model is developed and applied in multi-features cluster filtering, which greatly improves the performance of event detection.

Table 1 Comparison between different event detection methods

Setting	Twevent[2]	Sensor[9]	DBED (the proposed)
Feature	segment	n-gram	tweet
Tweet Representation	segment-based	-	-
Feature Selection	phrase-level burstiness	phrase-level burstiness	-
Clustering	kNearNeighbor	hierarchical	DBSCAN
Cluster Filtering	multiple features using external resouce	phrase-level burstiness	multiple features including document-level burstiness
Method	unsupervised	unsupervised	unsupervised

3 Document-level event detection

Fig. 1 shows a framework of the document-level event detection. Preprocessing (Section 3.1) includes non-English tweets removal, tweet normalization and keyword-based tweet filtering. Tweet clustering (Section 3.3) introduces the clustering algorithm used for yielding potential event candidates. Cluster filtering (Section 3.4) presents a proposed efficient heuristic to differentiate news events and mundane topics. Section 3.5 introduces event reporting procedure to get more readable events. Section 3.6 analyzes time cost of the document-level event detection model.

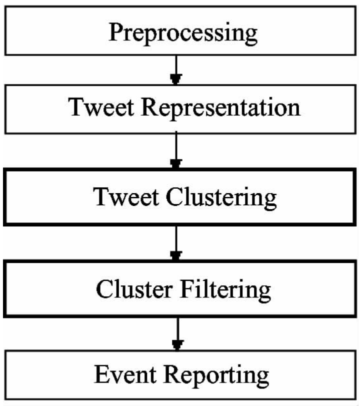

Fig. 1 Framework of proposed document-level Tweet event detection model

3.1 Preprocessing

Raw Twitter data are noisy and written in multiple languages. The event detection is focused on English tweets and following preprocessing is conducted.

Language filtering Non-English tweets are removed by a language detection method langid. py[25]. The tool firstly conducts tokenization and feature selection based on Aho-Corasick string matching algorithm, and then identifies language via a naive Bayes classifier. The tweets are kept whose language output by langid. py is English as targets.

Lexical normalization Non-standard forms of words in tweets influences results of NLP tasks in tweets. For example, 'earthqu', 'eathquake', 'earthquakeee', all appeared in a Twitter corpus. In this work, words are normalized in tweets via a lexical normalization dictionary extracted from microblogs[26]. The dictionary is built after extracting contextually similar OOV-IV (out-of-vocabulary, in-vocabulary) pairs and pair re-ranking via string similarity. OOV-IV pairs are selected by considering influences of context window size, n-gram order, KL divergence, JS divergence and etc. Re-ranking of pairs, which is conducted via string similarity algorithms like standard edit distance, is proven to be more effective than no-ranking or ranking by frequencies.

Tweet filtering A simple and efficient tweet filtering method is keyword-based filtering, which selects high quality tweets with a few predefined keywords. These keywords can be target entities in a specific domain such as company stock symbols '$AAPL' to select tweets related to Apple company or general words like '#breakingnews' to select potential breaking news in all domains. Here a domain-specific keyword-based filtering is adopted. Details are shown in Section 4.1.

3.2 Tweet representation

Two different text representation methods are tried to represent tweets.

Firstly, a traditional bag-of-words (BOW) method is used. The BOW method constructs a vocabulary of words V, and represents each document d with vocabulary size vector v. Each item of vector v_i indicates word w_i appearing in document d. Instead of simply using 1/0 to indicate the word w_i's appearance or not, some feature weighting methods can be used to improve the representation. A term frequency-inverse document frequency (TF-IDF) weighting method is used to distinguish different contribution of each word to the document. High term frequency words show more importance. High document frequency words, which appeared in most documents such as 'the', 'a' and 'an', will have little contribution to the document representation.

The main challenges with BOW include high sparsity of documents and difficulty to find semantically similar documents with very different and non-overlapping words. Recently, distributed representation[8] for words/texts is proposed to solve the problem. The proposed distributed representation method is denoted as Embedding. Embedding-based method represents a word with a low dimensional real-valued vector with dimension m. The word vectors are trained with continuous bag-of-words (CBOW) neural network with an input layer, a projection layer and an output layer. The neural network is considered to be a language model. It predicts a word given its context. Given a sentence $s = \{w_1, w_2, \cdots, w_i, \cdots, w_n\}$, w_i is the word to be predicted, context window size is $[-2, +2]$. The input layer firstly looks up each words' initial vector and gets $v = \{v_1, v_2, \cdots, v_i, \cdots, v_n\}$. In the projection layer, context vectors of w_i, $[v_{i-2}, v_{i-1}, v_{i+1}, v_{i+2}]$ are added together forming a new vector v_i'. A softmax output layer then classifies v_i' to vocabulary size $|V|$ probability vector, in which each item is the probability of corresponding word to be the predicted word. Hierarchical softmax classifier can be used to reduce classification complexity.

The trained word vectors can encode their semantic information making words' semantic similarity calculation possible. For example, string similarity of 'football' and 'soccer' is zero, while their semantic similarity is very high. Embedding-based method represents a document with the average embedding vector of all words within the document. Embedding-based representation also makes document-level semantic matching possible, which leads to opportunity for calculating document-level statistics. In practical, word embeddings used in this paper are 100-dimensional dense vectors, pre-trained with Word2Vec[8] on 20 million tweets.

3.3 Tweet clustering

Tweet vectors are grouped into clusters as event candidates. As focus is to investigate document-level burstiness feature, the traditional K-means algorithm and a density-based clustering method DBSCAN[7] are used for tweet clustering. More advanced optimization clustering methods will be left to future work.

K-means is a simple yet highly efficient clustering algorithm. Given a set of instances to be clustered, and K initially selected instances as centroids of K clusters, the algorithm iteratively assigns each instance to its nearest cluster centroid, and updates the centroids. The algorithm stops when centroids of clusters do not change. Noted that the number of clusters K is a hyper-parameter.

DBSCAN is a density-based clustering method. Given a set of points to be clustered, DBSCAN finds all high density core points and expands them to low density boundary points. Here, high density core points are defined to be those which have more than *minPt* nearby points within radius distance ϵ. Low density boundary points are non-core points which locate in core points' ϵ-distance area.

Compared to K-means, the advantage of DBSCAN is two-fold. Firstly, DBSCAN does not require the predefinition of the cluster number, which is more realistic because the number of topics at each time window can vary. Secondly, DBSCAN allows tweets to be isolated and not clustered. In contrast, K-means assigns each tweet to one of the k clusters.

For both methods, the hyper-parameters are tested on development data. In practical, implementation of K-means and DBSCAN is used in scikit-learn tool.

3.4 Cluster filtering

Besides news events, the resulting clusters also contain lots of mundane topics. Therefore cluster filtering is applied to differentiate news events from mundane topics. Different features are investigated for unsupervised cluster filtering. Generally speaking, widely discussed clusters with stronger inner connection, larger distance to seed mundane tweets, higher similarity to seed news have higher probability to be news events. In addition, a simple yet efficient temporal feature is proposed for tweet cluster filtering.

3.4.1 Burstiness: a semantic temporal feature

Both news events and mundane topics can be frequently discussed by Twitter users, which makes general statistic features difficult to differentiate them. A major difference between news events and mundane topics is the temporal information. In particular, news events are widely discussed only in a small time window near the time it happened, while mundane topics may always be popular. According to the observation, a temporal feature is designed for cluster filtering.

The bursty property is defined as a feature's abnormally high frequency in a time window compared to the average frequency in several time windows. For a formal definition, the traditional burstiness defined on word-level is first introduced.

Word-level burstiness Given a time period $D = \{d_1, d_2, \cdots, d_M\}$, word w's frequencies on each time window $\{f_{w,d_1}, f_{w,d_2}, \cdots, f_{w,d_M}\}$ are counted from corpus. Specifically, f_{w,d_1} is the frequency of w in all documents published in d_1. Then, z-score[27], $z(w,d)$ is exploited to measure the burstiness of feature w in time window d, which is defined in Eq. (2).

$$z(w, d) = \frac{f_{w,d} - \mu_w}{\sigma_w} \tag{2}$$

$$\mu_w = \frac{1}{M}\sum_{m=1}^{M} f_{w,d_m} \tag{3}$$

$$\sigma_w = \sqrt{\sum_{m=1}^{M} (f_{w,d_m} - \mu_w)^2} \tag{4}$$

$z(w, d)$ can be used to measure the difference between observation $f_{w,d}$ and the mean μ_w (shown in Eq. (3)) in units of standard deviation σ_w (shown in Eq. (4)). It is more general than metrics such as the one-sigma score of Li, et al.[2] (corresponding *z-score* = 1) and the three sigma rule of McMinn and Jose[21] (corresponding *z-score* = 3).

Tweet-level burstiness Similar to words, a tweet's burstiness measures how abnormal its frequency is in a time window. However, unlike words, it is infeasible to calculate tweet burstiness by exact match due to sparseness. Using a tweet's semantic neighbor frequency is proposed for calculation of a tweet's semantic burstiness.

Assuming that T_d represents a set of tweets published in a time window d. Each tweet t's burstiness is calculated as follows.

- Tweet representation. Each tweet in T_d is represented as a semantic vector.
- Tweet semantic frequency. For each target tweet t, the number of its near neighbors within distance is taken as t's semantic frequency in time window d, noted as $f_{t,d}$. An illustration is shown in Fig. 2. Noted, tweet t's near neighbors within are calculated by radius-based Nearest Neighbor. Locality-sensitive Hashing (LSH) is used to search for a tweet's near neighbors. In particular, the FALCONN package[28] is used as the LSH implementation.

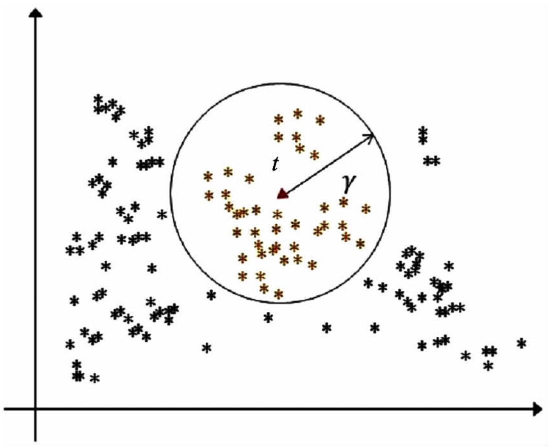

Fig. 2 Tweet semantic frequency

- Tweet t's z score $z(t,d)$ is calculated in Eq. (5). In particular, the word frequency $f_{w,d}$ in Eq. (2) is replaced with the tweet semantic frequency $f_{t,d}$.

$$z(t, d) = \frac{f_{t,d} - \mu_t}{\sigma_t} \tag{5}$$

$$\mu_t = \frac{1}{M}\sum_{m=1}^{M} f_{t,d_m} \tag{6}$$

$$\sigma_t = \sqrt{\sum_{m=1}^{M} (f_{t,d_m} - \mu_t)^2} \tag{7}$$

The tweet-level burstiness is expanded to cluster-level burstiness to estimate a cluster's temporal information. In particular, the centroid vector of a cluster, which is the averaged vector of tweet vectors within the cluster, is used to represent the cluster for calculating cluster-level burstiness. When calculating burstiness z_c of cluster c, the time period is defined as $(d - ltw, d)$, where d is the time window that cluster c is obtained and ltw is a left time window threshold. Thus only history information is used to estimate a cluster's burstiness. A cluster's frequency is presented in different time windows in Fig. 3, in which the cluster is bursty in the last time window.

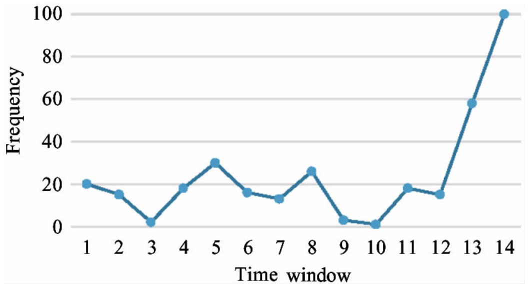

Fig. 3 A cluster's frequency in different time windows

3.4.2 General statistical features

In addition to burstiness, the following general statistical factors are considered for cluster filtering. Given cluster c and the set the tweets T_c in c.

Density of cluster, denoted as *density*, is the averaged similarities of all tweet pairs in T_c. Intuitively, the larger *density* is, the stronger inner connection the cluster has. And stronger inner connection ensures cluster c to be more pure and containing only one event. In particular, tweet similarity is calculated for each tweet pair and stored in $TPSim_c = \{sim_{t_i t_j} \mid i \in [1, \mid T_c \mid - 1], j \in [i + 1], \mid T_c \mid]\}$. *Density* is calculated as $average(TPSim_c)$.

Entity mention, denoted as *entity*, is calculated as the average number of entity names appearing in T_c. Higher quality clusters containing more meaningful entity names are suggested instead of referring to entities by hashtags or cashtags. *Entity* is calculated as $average(ENum_c)$, where $ENum_c$ contains the number of entities appearing in each tweet ($ENum_c = \{num_1, num_2, \cdots, num_{\mid T_c \mid}\}$).

Seed mundane tweets, denoted as *sTweet*, is estimated as average distance of tweets in T_c to a list of seed mundane tweets. The seed mundane tweet list consists of ten tweets manually selected from the development set. For example, a tweet 'our stock pick on \$thcz is up 638 . 15 for our subscribers get our next pick early \$tcb \$mck \$study' is a promotion tweet, which has high frequency in our finance dataset. For calculation, seed mundane tweet list is denoted to be $MT = \{t_1, t_2, \cdots, t_{10}\}$, distances of tweets in T_c to seed mundane tweets are stored in $MTDist_c = \{dist_{t_i t_j} \mid i \in [1, \mid T_c \mid], j \in [1, 10]\}$. *sTweet* is $average(MTDist_c)$.

Seed news headlines, *sNews*, is estimated as the average similarity of tweets in T_c to seed news headlines. Ten seed news headlines are manually selected from news published in Reuters. For example, an IPO news 'mcgraw hill education prepares for ipo'. Example mundane tweets and news headlines are shown in Table 2. For calculation, seed news headlines to be $NH = \{t_1, t_2, \cdots, t_{10}\}$, similarities of tweets denoted in T_c to seed news headlines are stored in $NHSim_c = \{sim_{t_i t_j} \mid i \in [1, \mid T_c \mid], j \in [1, 10]\}$. *sNews* is calculated as $average(NHSim_c)$.

Table 2 Examples of seed mundane tweets and news headlines

ID	Seed Mundane Tweet / Seed News
1	\$71000 in one trade by follwing their signals more info here \$cvc \$cvd \$cve
2	our penny stock pick on \$ppch closed up another 51.31 today huge news \$cce \$cadx \$grmn
3	our stock pick on \$thcz is up 638 . 15 for our subscribers get our next pick early \$tcb \$mck \$study
4	gains over 2500 in one trade subscribe here \$emo \$emq \$emr
5	largest food and staples retailing earnings 1 \$wmt 2 \$cvs 3 \$wba chart
6	dow chemical to sell agrofresh for \$860 mln in asset sale drive
7	oracle ceo sees benefit if rival buys salesforce. com
8	expedia inc first quarter profit tops expectations
9	cigna profit beats estimate as it adds more customers
10	mcgraw hill education prepares for ipo

3.4.3 Filtering score calculation

Given all features above, the newsworthiness $\rho(c)$ for cluster c is designed as in Eq. (8). Higher newsworthiness indicates the cluster would have higher probability to be a news event. After ranking by $\rho(c)$, top K_c ranked clusters are regarded as event clusters.

$$\rho(c) = density + S(entity) + sTweet + sNews + S(-Z_c) \quad (8)$$

Here $S(.)$ means the sigmoid function, used for normalization, Z_c is the burstiness of cluster c. This work assumes that a cluster with larger density, more entities included, larger distance to mundane tweets, higher similarity to news headlines and most important, higher bursty score, should have higher newsworthiness score.

3.5 Event reporting

After clustering filtering, a set of clusters which includes a list of tweets is got. Duplicate tweets are filtered for more readable outputs. In particular, semantic matching is used instead of exact string matching to detect a tweet's duplicate tweets to avoid sparsity. Similar to calculation of tweet semantic frequency in Section 3.3.1, a tweet t's radius distance neighbor is taken as its duplicate tweets. The radius is set to be 0.1. A set of unique tweets $\{t_1, t_2, \cdots, t_L\}$, and number of duplicate tweets $\{f_{t_1}, f_{t_2}, \cdots, f_{t_L}\}$ are obtained. Given a cluster, the top K_t tweets which have largest number of duplicates are selected to represent the event.

3.6 Time complexity

In this section, time cost of the document-level tweet event detection model is analyzed. The time cost is focused on two steps: tweet clustering and bursty cluster feature calculation. Suppose there are n tweets in current time window d to be clustered, time complexity of K-means algorithm is $O(n \times K \times t)$, where K is the number of clusters and t is the number of iterations. Time complexity of DBSCAN algorithm is $O(n \times \log(n))$, as it only needs one single pass on dataset.

According to Ref. [28], time cost of finding a query document's nearest neighbor is $O(L \times d + m \times \log m)$ via the used LSH variant, where L is the number of hash tables, d is the document vector dimension and m is the probing sequence length. As L, d and m are constant numbers, nearest neighbor search time cost is nearly $O(1)$. Suppose $|C|$ clusters are obtained and there are about $(ltw + 1) \times n$ tweets in current time window d's valid context time window $[d - ltw, d]$. Cluster's semantic frequency calculation dominates cluster burstiness feature calculation, which costs $O(|C| \times (ltw + 1) \times n)$. As clusters number $|C|$ and ltw are small number, this step's cost is $O(n)$.

In summary, like most other algorithms, the efficiency of proposed document-level event detection model relies on tweet clustering algorithm.

4 Experiments

4.1 Data

Instead of using general twitter dataset for evaluation, datasets on specific domains are chosen as they are allowed to conduct semi-automatic evaluation, by matching the resulting output events with ground truth news. The methods' effectiveness and robustness on two Twitter datasets are evaluated in different domains, namely the finance domain and the sports domain. Statistics of the datasets are shown in Table 3. '#TW' means the number of time windows.

Table 3 Statistics of dataset

Dataset	#TW	#Tweet	#Segment	#Word
Finance-Dev	3	72K	41K	40K
Finance-Test	5	130K	56K	55K
Finance	31	659K	141K	128K
Sports-Dev	4	4K	-	7K
Sports-Test	9	13K	-	14K
Sports	13	212K	-	102K

The finance dataset contains tweets published in one month (May, 2015), crawled using Twitter Streaming API by matching with a keyword list. Twitter defined a special kind of hashtag, cashtag, which starts with '$' and followed by a company's stock symbol, to indicate a company-related tweet. For example, the cashtag for Apple Inc. is '$AAPL'. In practical, a list of cashtags for S&P500 companies is used as keywords. Three days (May 6th-8th, 2015) are taken for development (Finance-Dev) and five days (May 11th-15th, 2015) for test (Finance-Test).

Financial news corresponding to tweets' published date are crawled from Reuters for automatic evaluation on finance dataset. Only news related to S&P 500 companies are kept. 492 news are obtained for 31 days in May, 2015. According to our observation, news published in every weekend or U.S. public holiday largely drops (less than 8). Due to this, these days are ignored when choosing twitter development and test set.

The Sports dataset is released by Aeillo, et al.[9], containing tweets for the 2012 FA Cup final (Chelsea V.S. Liverpool 2-1). The dataset is partitioned into one-minute time windows, 13 of which contain at least one ground truth event of FA Cup. Four time windows are treated as the development set (Sports-Dev) and nine for the test set (Sports-Test). Ground truth events for sports dataset, which include significant events like kick-off, score, yellow card, etc., are manually selected from published news in Ref. [9].

4.2 Evaluation metrics

Performance of different tweet event detection methods on the finance dataset is evaluated by three popular ranking metrics, namely Precision@K, mean average precision (MAP) and normalized discounted cumulative gain (NDCG)[29].

In particular, the clustering results are checked manually. Each method outputs K_c clusters as event candidates. For tweet-based methods, an event output is represented as K_t tweets. For Twevent, an event output is represented by a set of segments. An event is labelled as a true event only if there is a corresponding news event and the tweets talks about one event.

In addition, methods' recall by matching system outputs to financial news published in the time window is also evaluated. Each finance news headline is analyzed in Ref. [30] using off-the-shelf models for Part-of-Speech tagging and dependency parsing. S&P 500 companies appearing in the headlines are extracted. Important words are extracted if they are nouns, verbs and are involved in dependency arcs of 'root', 'sub', 'obj', 'vc', 'vmod', 'nmod'. An event output matches news only if they discuss the same company and at least one of the words is extracted from a news title appearing in one of the K_t tweets. Recall is calculated as the ratio of matched news out of all financial news events.

Note that the proposed recall measure is a rough approximation of the real recall, since not all events are covered by news articles, and the matching algorithm does not exactly match news headlines with tweets by the meaning of them. It can serve as a rough estimation of the real recall, which is similar to the estimation by Li et al.[2] according to the number of events that are returned. The proposed measure can be more accurate since it is based on news titles. Given the above, the precision measure is relatively more reliable.

For the sports dataset, the evaluation metrics of Sensor[9] is adopted, which are topic precision (T-Prec), keyword precision (K-Prec) and keyword recall (K-Rec). Similar to the experimental settings of Sensor[9], the top 2 clusters are chosen as output events. Due to the small size of time windows in the sports dataset, seed mundane tweets or seed news are not used for cluster filtering. Only burstiness, density and the size of clusters are used as filtering features.

4.3 Hyper-parameters

Parameters in the method are tuned on the finance development dataset. For DBSCAN, radius distance parameter ϵ is set to 0.2 by considering both performance and efficiency. For calculating cluster-level burstiness, the history time window size is set to 15, and radius distance threshold is set to 0.4. The number of output clusters K_c is set to 30. The number of output tweets K_t is set to 5.

4.4 Development experiments

4.4.1 Influence of tweet representation and clustering methods

The influence of different tweet representation and clustering methods on the Finance-Dev dataset, comparing BOW and embedding-based tweet representation are investigated. The clustering methods are compared with traditional K-means algorithm. Fig. 4 shows the results in MAP, Precision@K and NDCG.

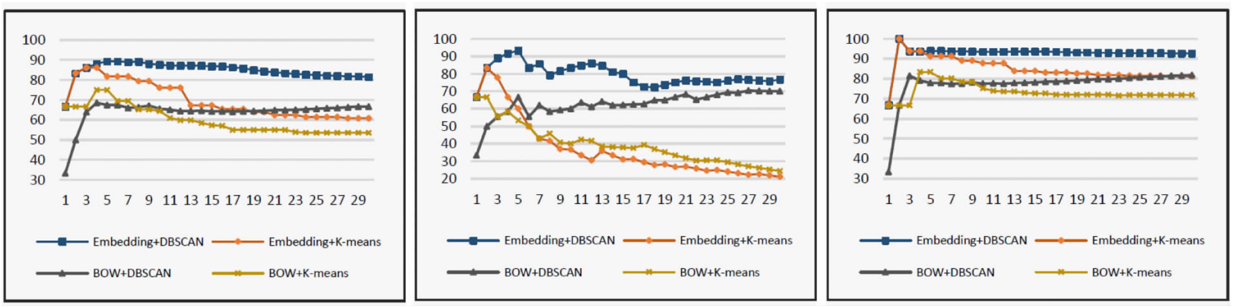

Fig. 4 Evaluation of different clustering methods and features on Finance-Dev. Three figures (from left to right) show results on MAP, Precision@K and NDCG respectively

The experimental setting of embedding-based representation and DBSCAN clustering method gives the best performance in all evaluation metrics, which is adopted in the remaining experiments. In terms of MAP and NDCG, BOW + K-means performs the worst in the four methods, while Embedding + K-means are slightly better than BOW + DBSCAN. In Precision@K, DBSCAN based methods greatly outperform K-means based methods, and the difference becomes larger with the increasing of K. The high precision@30 of DB-

SCAN shows that DBSCAN gives far more valuable events than K-means with larger K.

4.4.2 Influence of cluster filtering features

Table 4 shows results of using-single-feature tests and leave-one-out feature experiments for cluster filtering. From the single feature tests (lines 1-5), it is found that inconsistent results exist in the selection of the best feature. According to the performance in MAP, Precision@K and NDCG, comparable results for burstiness, entity and seedNews are obtained. In terms of recall, the two best features burstiness and entity with comparable performance (over 20%), seedNews and density give similar low recalls (below 10%). The worst feature is seedTweet in terms of all evaluation metrics.

From the leave-one-out feature tests, it is found that seedNews is the most important feature in terms of ranking metrics, followed by burstiness. entity and burstiness, which are the most useful features for recall. In conclusion, burstiness performs well consistently in all tests, which verifies the effectiveness of proposed document-level temporal feature.

Table 4 Results of single feature tests and leave-one-out feature tests of cluster filtering on Finance-Dev dataset

Feature	Prec@30	MAP	NDCG	Rec
density	36.67%	46.34%	67.17%	8.45%
entity	64.44%	64.16%	80.92%	**22.54%**
seedTweet	28.89%	21.69%	42.08%	4.23%
seedNews	58.89%	**68.46%**	**84.41%**	7.04%
burstiness	**65.56%**	65.04%	79.97%	21.13%
All	**76.67%**	**81.50%**	**92.73%**	**32.39%**
– density	0.00%	–0.22%	–0.12%	–1.40%
– entity	–4.45%	–1.29%	–4.4%	**–9.85%**
– seedTweet	–4.45%	–2.24%	–2.11%	–4.12%
– seedNews	**–14.45%**	**–18.07%**	**–13.45%**	–5.63%
– burstiness	–6.67%	–10.07%	–8.66%	–8.45%

4.5 Final test results

Table 5 shows the results of the method and baseline Twevent on Finance-Test dataset. Compared with Twevent, the method significantly improved Pre@K (26.00% →77.33%), MAP (45.74% →80.59%) and NDCG (63.57% →89.89%) because of the effectiveness of cluster ranking with tweet burstiness. Improvement on Rec (7.21%→10.81%) indicates that the method obtains more news events. Low recall of both Twevent and the method indicates the small overlap of events reported by traditional news media and social media.

Table 5 Results on Finance-Test

System	Prec@30	MAP	NDCG	Rec
Twevent	26.00%	45.74%	63.57%	7.21%
DBED	**77.33%**	**80.59%**	**89.89%**	**10.81%**

After analysis, it is found that a large fraction of incorrect Twevent outputs are those clusters with incomplete information. For example, lots of clusters include entities only, without important event trigger information. This is caused by the drawback of kNearNeighbor clustering in Twevent, in which each segment can belong to one cluster only. In addition, several events of one type may use same keywords. For example, if two events in the same day contain the same keyword “purchase”, they are likely treated as the same event by Twevent. In contrast, the method uses full document information, thereby alleviates this issue.

To evaluate the robustness of the method, the results (forth line) are compared with the baseline methods of Ref. [9] on the sports dataset, shown in Table 6. Sensor (third line) shows the proposed method in Ref. [9], which is discussed in Section 3. The first two lines in Table 6 show results of two strong baseline methods, which are selected from Ref. [9]. The LDA method is a topic model based on event detection method, which uses LDA (Latent Dirichlet Allocation) for clustering. Doc-p is a document-based topic detection method by Locality Sensitive Hashing. The method improves the topic recall from 76.92% by Sensor and Doc-p to 88.89%. Keyword precision and recall are also improved compared to all baseline methods. The improvement verifies the effectiveness of our method on the sports domain, which shows the robustness of the method.

Table 6 Results on the sports dataset

System	T-Rec@2	K-Pre@2	K-Rec@2
LDA	69.23%	16.37%	68.29%
Doc-p	76.92%	33.73%	58.33%
Sensor	76.92%	29.89%	57.78%
DBED	**88.89%**	**38.04%**	**70.83%**

4.6 Example output

Table 7 shows four example output event clusters (ID starting with ‘e’) by the proposed method and the baseline methods, respectively. The first two are outputs on the finance dataset and the last two are on the sports dataset. In addition, if the corresponding cluster matches a ground truth news/event, the ground truth is listed (with ID starting with ‘g’) in bold/italic after the cluster.

Table 7 Example output events

Method	**ID**	Tweets/**News**/*Ground Truth*
News Title	**g1**	**exclusive: microsoft has no plans to pursue salesforce sources**
	g2	**syngenta rejects $45 billion monsanto takeover offer**
DBED	e1	hot @ nadiaspeaks exclusive microsoft has no plans to pursue salesforce sources $crm $msft
	e2	$mon syngenta rejects unsolicited monsanto acquisition proposal 4th update
Twevent	e1	microsoft; salesforce; electronic; corporation; business; tv; arts; $hlf; $crm; $ea; 's; $msft
	e2	syngenta; monsanto; billion; breaking; offer; talks; room left; make initial; $45; rebuffed
Ground Truth	*g3*	mikel; [yellow card booking] gerrard; foul; agger
	g4	[salomon kalou]; run; box mazy
DBED	e3	a midfield of lampard mikel ramires beats one of gerrard spearing henderson who knew #lfc
	e4	great mazy run by kalou into the box but he gets ambushed by the liverpool defence before he can shoot #cfcwembley #facupfinal sl
Sensor	e3	mikel yellow card
	e4	liverpool gets ambushed kalou defence box mazy run before @ chelseafc great shoot #cfcwembley #facup

Compared with Twevent, the outputs of the document-level model are notably more human readable. It can be highly difficult to guess the real event without prior knowledge of the event, by reading the segment cluster of Twevent. In contrast, most tweet clusters from the document-level model are direct understandable.

Sensor selects tweets randomly, which contain most representative n-grams in a cluster for outputs. This can be useful when the time window is smaller, and thus ambiguity is limited for the matching above. In contrast, the method gives detected highly understandable tweet-based event clusters automatically.

5 Conclusion and future work

A two-stage document-level model is built for unsupervised tweet event detection, introducing a novel tweet-level burstiness measure for selecting newsworthy event cluster candidates, which demonstrates its usefulness empirically. The highly effective and efficient embedding-based representation and the duplication feature of tweets provides the foundation of our document-level burstiness. This work gives a hint of extending useful word-level statistics to document-level. In addition, the effectiveness and robustness of the method are verified in two different domain datasets, compared with two state-of-the-art baselines.

References

[1] Yom-Tov E, Diaz F. Location and timeliness of information sources during news events[C]. In: Proceedings of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval, Beijing, China, 2011. 1105-1106

[2] Li C, Sun A, Datta A. Twevent: segment-based event detection from tweets[C]. In: Proceedings of the 2012 ACM International Conference on Information and Knowledge Management, Maui, USA, 2012. 155-164

[3] Becker H, Naaman M, Gravano L. Beyond trending topics: real-world event identification on twitter[C]. In: Proceedings of the 2011 International Conference on Weblogs and Social Media, Barcelona, Spain, 2011. 438-441

[4] Li R, Lei K H, Khadiwala R, et al. Tedas: A twitter-based event detection and analysis system[C]. In: Proceedings of IEEE International Conference on Data Engineering, Arlington, USA, 2012. 1273-1276

[5] Ritter A, Wright E, Casey W, et al. Weakly supervised extraction of computer security events from twitter[C]. In: Proceedings of the 24th World Wide Web Conference, Florence, Italy, 2015. 896-905

[6] Lee S, Lee S, Kim K, et al. Bursty event detection from text streams for disaster management[C]. In: Proceedings of the 2012 World Wide Web Conference Companion, Lyon, France, 2012. 679-682

[7] Ester M, Kriegel H P, Sander J, et al. A density-based algorithm for discovering clusters in large spatial databases with noise[C]. In: Proceedings of the 1996 International Conference on Knowledge Discovery and Data Mining, Portland, USA, 1996. 226-231

[8] Mikolov T, Sutskever I, Chen K, et al. Distributed representations of words and phrases and their compositionality[J]. *Advances in Neural Information Processing Systems*, 2013, 26:3111-3119

[9] Aiello L M, Petkos G, Martin C, et al. Sensing trending topics in twitter[J]. *IEEE Transactions on Multimedia*, 2013, 15(6):1268-1282

[10] Li J, Ritter A, Cardie C, et al. Major life event extraction from twitter based on congratulations/condolences speech acts[C]. In: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing,

Doha, Qatar, 2014. 1997-2007
[11] Agarwal P, Vaithiyanathan R, Sharma S, et al. Catching the long-tail: Extracting local news events from twitter [C]. In: Proceedings of the 6th International AAAI Conference on Weblogs and Social Media, Dublin, Ireland, 2012. 379-382
[12] Benson E, Haghighi A, Barzilay R. Event discovery in social media feeds[C]. In: Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies, Portland, USA, 2011. 389-398
[13] Diao Q, Jiang J, Zhu F, et al. Finding bursty topics from microblogs [C]. In: Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics, Jeju Island, Korea, 2012. 536-544
[14] Zhou D, Chen L, He Y. An unsupervised framework of exploring events on twitter: Filtering, extraction and categorization[C]. In: Proceedings of the 29th AAAI Conference on Artificial Intelligence, Austin, USA, 2015. 2468-2474
[15] Chierichetti F, Kleinberg J, Kumar R, et al. Event detection via communication pattern analysis[C]. In: Proceedings of the 8th International AAAI Conference on Weblogs and Social Media, Ann Arbor, USA, 2014. 51-60
[16] Kitagawa Y, Komachi M, Aramaki E, et al. Disease event detection based on deep modality analysis[C]. In: Proceedings of the ACL-IJCNLP 2015 Student Research Workshop, Beijing, China, July 2015. 28-34
[17] Petrovic S, Osborne M, Lavrenko V. Streaming first story detection with application to twitter[C]. In: Proceedings of the 2010 North American Chapter of the Association for Computational Linguistics, Los Angeles, USA, 2010. 181-189
[18] Ifrim G, Shi B, Brigadir I. Event detection in twitter using aggressive filtering and hierarchical tweet clustering [C]. In: Proceedings of the SNOW 2014 Data Challenge, Seoul, Korea, 2014. 33-40
[19] Fung G P C, Yu J X, Yu P S, et al. Parameter free bursty events detection in text streams[C]. In: Proceedings of the International Conference on Very Large Data Bases, Trondheim, Norway, 2005. 181-192
[20] Qin Y, Zhang Y, Zhang M, et al. Feature-rich segment-based news event detection on twitter[C]. In: Proceedings of the the 6th International Joint Conference on Natural Language Processing, Nagoya, Japan, 2013. 302-310
[21] McMinn A J, Jose J M. Real-time entity-based event detection for twitter[C]. In: Proceedings of the Conference and Labs of the Evaluation Forum, Toulouse, France, 2015. 65-77
[22] Platakis M, Kotsakos D, Gunopulos D. Searching for events in the blogosphere [C]. In: Proceedings of the World Wide Web Conference, Madrid, Spain, 2009. 1225-1226
[23] Ou G, Chen W, Wang T, et al. Exploiting community emotion for microblog event detection[C]. In: Proceedings of the Conference on Empirical Methods in Natural Language Processing, Doha, Qatar, 2014. 1159-1168
[24] Zhao X, Shu B, Jiang J, et al. Identifying event-related bursts via social media activities[C]. In: Proceedings of the 2012 the Conference on Empirical Methods in Natural Language Processing, Jeju Island, Korea, 2012. 1466-1477
[25] Lui M, Baldwin T. langid. py: an off-the-shelf language identification tool[C]. In: Proceedings of the ACL 2012 System Demonstrations, Jeju Island, Korea, 2012. 25-30
[26] Han B, Cook P, Baldwin T. Automatically constructing a normalisation dictionary for microblogs[C]. In: Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, Jeju, Korea, 2012. 421-432
[27] Kreyszig E, Kreyszig H, Norminton E J. Advanced Engineering Mathematics[M], 10th ed. Hoboken, NJ: Wiley, 2011
[28] Andoni A, Indyk P, Laarhoven T, et al. Practical and optimal LSH for angular distance[C]. In: Proceedings of the 28th Annual Conference on Neural Information Processing Systems, Montreal, Canada, 2015. 1225-1233
[29] Jarvelin K, Kekalainen J. Cumulated gain-based evaluation of ir techniques[J]. *ACM Transactions on Information Systems*, 2002, 20(4): 422-446
[30] Zhang Y, Clark S. Syntactic processing using the generalized perceptron and beam search[J]. *Computational Linguistics*, 2011, 37(1): 105-151

Qin Yanxia, born in 1985. She is currently a Ph. D candidate of Harbin Institute of Technology. Her advisor is Prof. Zhang Min. She received her B. S. degree from Northeast Normal University and M. E. degree from Harbin Institute of Technology in 2010 and 2012, respectively. She interned at Institute for Infocomm Research in 2012, under supervision of Prof. Zhang Min. She interned at Singapore University of Technology in 2013 and 2017, under the supervision of Prof. Zhang Yue. Her research interests include natural language processing, deep learning, information extraction and social network.