IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received February 17, 2019, accepted February 28, 2019, date of current version March 25, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2903095

Burst Hotspots Dynamic Detection and Tracking on Large-Scale Text Stream

JIANYI HUANG[1,2], JIANJIANG LI[1,2], YINGYING CHEN[1,2], JIANKUN SUN[1], AND PENG SHI[2,3]

[1]School of Computer and Communication Engineering, University of Science and Technology Beijing, Beijing 100083, China
[2]Beijing Key Laboratory of Knowledge Engineering for Materials Science, Beijing 100083, China
[3]National Center for Materials Service Safety, University of Science and Technology Beijing, Beijing 100083, China

Corresponding author: Peng Shi (shipengustb@sina.com)

This work was supported in part by the National Key R&D Program of China under Grant 2017YFB0803302, in part by the National Nature Science Foundation of China under Grant U1836106, and in part by the Fundamental Research Funds for the Central Universities under Grant 06116104.

ABSTRACT The online social network has become an important communication tool for people and forms a virtual society interacting with the real world. The numerous events rapidly spread through social networks and may become hotspots in a short period of time. Especially, the negative events vibrate national security and social stability, potentially causing a series of social problems. Therefore, detection and tracking burst hotspots on social networks are of great significance. However, this problem is non-trivial because of the challenges of massive noise, sparsity, high-dimensionality, and dynamic changing. To address these challenges, this paper proposes a distributed method of burst hotspots dynamic detection and tracking based on Map/Reduce. To judge the relevance of the current text and previous texts, the keyword similarity matrix is calculated by using Word2Vec and context information. The keyword weights are modified according to the association model to further reduce noise. The proposed method is not only robust to the sparsity of short text but also overcomes the curse of dimensionality. Finally, it can dynamically detect and track burst hotspots from large-scale short text stream on the Hadoop platform. The experiments have shown that the proposed method outperforms state-of-the-art algorithms.

INDEX TERMS Dynamic detection and tracking, Map/Reduce, online social network, text clustering, text stream.

I. INTRODUCTION

The vigorous development of online social networks has gradually penetrated into all aspects of people's life, including politics, education, economy and culture. On the one hand it makes people's life more convenient, but on the other hand it brings some negative effects to society. Negative information such as violence and rumors on social networks can cause panic and even affect social stability.

There are a large number of topics generated every day in social networks, and the evolution of these topics presents different characteristics [1]. Some topics gradually gain popularity and then slowly decline. And some other topics gain a lot of popularity in a short period of time, reach the peak and then decline quickly, called burst topics. Crane *et al.* [2] defined a topic with peak popularity above 20% of its total popularity as burst topic. Burst topics are usually triggered by breaking news, real world events, malicious rumors, etc. [3], [4]. From a practical perspective, outbreak period [5] would be an important quantity for many online problems. There are some predicting applications in different areas [6], but to predict the burst of Web contents is still a challenging topic. How burst topics diffuse and what public opinions are in the outbreak period are very important. Burst topics detection and tracking is of great significance for emergency monitoring.

However, in the era of mobile Internet, as the pace of life continues to speed up, simple and fast interaction with phrases are important to people with busy lifestyles. For example, people often use simple keywords to query information on search engines and communicate with friends by Twitter, Facebook and others. Although Chen *et al.* [7] recognized the human activity by multilayer extreme learning machine, human activity online is hard to analyze. The paper only

The associate editor coordinating the review of this manuscript and approving it for publication was Wenbing Zhao.

2169-3536 © 2019 IEEE. Translations and content mining are permitted for academic research only.
Personal use is also permitted, but republication/redistribution requires IEEE permission.
See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

focuses on the contents published on the Internet. Nowadays enormous amounts of texts are constantly being produced. According to the recent statistics [8], around 500 million tweets created on Twitter every day. These texts are noisy, various, continuous, infinite and dynamic [9]. Their text content and attention degree dynamically change [10]. The central idea of topic also changes over time [11]. The correlation of context is very strong. The timeliness of information is getting higher and higher. These problems make it increasingly difficult to detect and track topics timely and accurately. TF-IDF (term frequency-inverse document frequency) measure, VSM (vector space model), and normal text clustering methods may not work well when applied to short texts.

Based on the above problems, this paper proposes a distributed method of burst hotspots dynamic detection and tracking on large-scale text stream (BHDDT). The texts are processed one by one. Each topic is a sequence of texts which are closely related. The latest text of each topic reflects the latest developments of incident and public opinion. The latest text set of existing topics is used as the context to process the current text. First of all, the algorithm preprocesses the short texts, then calculates the keyword similarity matrix by using context information to judge the relevance of current text and previous texts, overcoming the sparsity caused by less keywords in short texts. Secondly, each new text does not need to be compared with all existing texts, but only needs to calculate the association degree with the latest text set of existing topics, thereby avoiding the increasing complexity caused by the ever-increasing data size. And then, the keyword weights are adjusted to highlight important keywords and suppress secondary keywords for the associated texts. At the same time, to detect irrelevant information, a threshold is set to adjust the deviation degree on each keyword, and filter out a large amount of redundant information. Finally, through these measures, the performance is guaranteed to be stable, efficient and accurate. The contribution of this work is mainly three-fold:

1) This paper calculates the keyword similarity matrix by using Word2Vec and context information to judge the relevance of current text and previous texts. It can cope with the sparsity and high-dimensionality problems of short texts, and can obtain the representative words of each cluster.
2) This paper sets a threshold to adjust the deviation degree on each keyword, highlights important keywords and suppresses secondary keywords for associated texts. A large amount of redundant information is further reduced.
3) Our proposed algorithm is parallelized based on Map/Reduce, which greatly improves the efficiency of the program operation and has good performance on large-scale text stream, avoiding the increasing complexity caused by the ever-increasing data size.

The rest of this paper proceeds as follows. Section II describes related work. Section III presents preliminaries. Section IV introduces our solution for burst hotspot dynamic detection and tracking. Experiments are conducted in Section V to evaluate our solution. Section VI concludes the paper.

II. RELATED WORK

A. TRADITIONAL TEXT CLUSTERING METHODS

Traditional long-text clustering methods generally calculate similarity by text feature vectors, such as typical partition-based clustering methods [12], hierarchical-based clustering methods [13], and density-based clustering methods [14].

The overall idea of partition-based clustering methods [15] is finally dividing a text set (N records) into K ($K << N$) packets, where each group representing a category. The evaluation standard of clustering results is that the correlation of records within a group is as close as possible, and the correlation of records between different groups is as far as possible. Furthermore, each record must belongs to a unique group, and each group contains at least one record. K-means [16], K-medoids [17] and CLARANS [18] are based on partitioning ideas. The advantages of such algorithms are simple and efficient. The disadvantages are described as follows. First, the clustering effect has a great dependence on the initial value selection; besides, there is no good way to choose the appropriate initial value. These problems cause the clustering time to be very unstable. Second, the noises and isolated points are not well handled. Finally, when data set is large, the processing time is difficult to be required, and the results are easy to fall into the local optimum.

Hierarchical-based clustering methods determine the relevance of the processed text set layer by layer until reaching the exit condition [19]. According to different processing orders, there are two ways of bottom-up and top-down. The following takes bottom-up hierarchical clustering as an example. It initially treats each element as a separate class, then analyses the association between classes, combines the classes that reach association threshold, and makes the same decision on the newly obtained class until the exit condition is reached [20]. There are many algorithms based on hierarchical clustering [21], such as BIRCH [22], CURE [23] and CHAMELEON [24]. Hierarchical clustering methods are relatively more flexible than partition-based methods. However, hierarchical clustering methods have higher computation complexity and time complexity. As less characteristic information, the cluster similarity is judged in large error, which will affect the subsequent judgments.

Density-based clustering methods are not based on the similarity between elements and classes, but on the density of elements distributions. Therefore, it can avoid its over-dependence on the shape of text distribution that determines similarity based on distance. If the density of points in a region is greater than the threshold, the point is assigned to the cluster adjacent to it [25]. DBSCAN [26], OPTICS [27] and DENCLUE [28] are all density-based clustering algorithms. DBSCAN is extremely sensitive to the scan radius and the minimum number of points included. Subtle changes in these

two parameters will have a great effect on clustering results. Unfortunately, the choices of these two parameters do not have a fixed rule to follow, and can only be artificially set by experience.

These methods use TF-IDF (Term Frequency-Inverse Document Frequency) to calculate the weights of keywords, and calculate the distance or similarity of two texts based on feature vector [29], [30]. A good clustering result can be obtained for long texts because a long text generally has many feature words. Even if the extracted feature vectors have error, there is no significant effect on clustering results. Short texts on social networks have the simplification characteristic [31]. At the same time, the weights of keywords calculated by TF-IDF are close to 0. As a result, the similarity between texts is calculated with a big error. Processing time and memory resource are wasted. When topics are varied and text data is large-scale, performance will drop rapidly and it will be difficult to complete in a short time. Therefore, this global range of text vector representations is not suitable for social network short texts.

B. SHORT TEXT CLUSTERING METHODS

In recent years, algorithms for short text clustering are emerging in an endless stream, but most of them are improved on traditional clustering algorithms [32], [33], like WR-Kmeans [12] and spherical K-means [34], which partly improve the clustering results of short text. However, to solve the problem of massive data processing, their precisions are still low; and calculations cost lots of time.

Shen *et al.* [35] proposed five Single-Pass clustering algorithms: SPB, SP-WC, SP-NN, SP-WNN, and SP-LF. Single-Pass clustering algorithm is a classic method of data stream clustering, processing datasets in real-time or offline. It can effectively cope with dynamic changes of text content. Single-Pass clustering algorithm is relatively sensitive to the order of input text, but it has little effect on the clustering results of datasets organized by time. For each new text, the similarities between the text and existing texts are calculated. The calculation complexity rapidly increases as data size grows.

The clustering algorithms improved on traditional clustering algorithms have poor performance because of the sparsity of short text. Some methods were proposed to solve the sparsity problem from the theoretical encoding [36]. In this paper, two new ways are proposed to solve the sparsity. One way is to expand short texts to long texts by using external knowledge sources such as Wikipedia [37], WordNet [20], HowNet [38], peripheral information sources [39], etc. However, the creation and maintenance of such resources can be very expensive and complex. The expanding methods significantly increase calculation. The other way is word embedding such as Word2Vec [40] and Doc2Vec [41]. Word2Vec analyzes semantic based on the dimensions of words, without considering paragraphs and sentences. Doc2Vec adds a paragraph vector based on Word2Vec. It can be used for text clustering and text categorization. However, Doc2Vec separately trains each local context window without taking advantage of the statistics contained in the global co-occurrence matrix. Doc2Vec cannot represent and process polysemous words.

Term co-occurrence clustering algorithms [42], [43] work well on scientific literature dataset, but they are very sensitive to the selected terms. The short texts on social networks have various irregular terms, resulting in uncertainty issues. WordCom [44] combines K-means and co-occurrence clustering for short text clustering by identifying word communities. WordCom is robust to sparse short texts. But it has the problems of high dimensionality and difficulty dealing with long texts.

C. TOPIC MODELS

The research on topic model was originally derived from Latent Semantic Indexing (LSI) [45] proposed by Christos H. Papadimitriou *et al.* Compared with VSM, LSI has smaller dimension and clearer semantic relationship. The disadvantages are that LSI lacks rigorous mathematical statistics foundation; the complexity of SVD decomposition is high; LSI cannot solve the problem of polysemous words; LSI cannot calculate the relevance degree between topics on time series.

Thomas Hofmann proposed Probabilistic Latent Semantic Indexing (PLSI) [46]. PLSI can quantitatively analyze the contribution of each word to the topic, so that the topics in different semantic spaces are connected. By means of the probability distribution of words within topics, PLSI can analyze the relevance degree between topics. The disadvantages are that PLSI is difficult to handle dynamically added documents; overfitting can occur in some datasets; the parameters linearly increase, and the performance of processing large-scale text data is difficult to meet the requirements.

Blei *et al.* [47] proposed Latent Dirichlet Allocation (LDA). LDA has been widely used and has derived many improved versions, such as BTM [48], GSDMM [49]. By introducing Dirichlet distribution, LDA is a complete probability generation model with strong mathematical theory. The parameter settings are relatively fixed. LDA is regardless with the size of training text dataset, and has good generalization ability. So in the large-scale text dataset, LDA performs better in target topic mining. However, LDA does not consider the order of words, assumes that the number of topics is fixed, and ignores topics' declining, dividing, and transferring.

III. PRELIMINARIES

A. TEXT STREAM

A text sequence is a sequence of large, fast, and contiguous data sequences that can be thought as a dynamic dataset growing indefinitely over time. The following firstly describes text sequence in a formal way. On this basis, sliding time window and the texts collected within it are defined. We summarize all of our notations in Table 1.

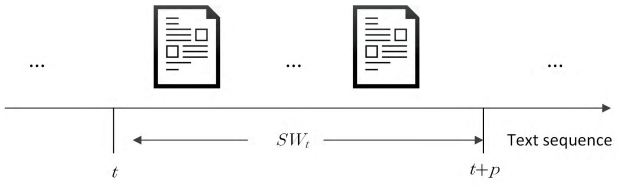

FIGURE 1. Sliding time window.

1) TEXT SEQUENCE

In the time-series text data, temporally consecutive text data items and their time windows form a text sequence $TS = \{x_1, x_2, \ldots, x_k, \ldots\}$, where x_k is the kth arriving text.

2) MICRO-TEXT

A micro-text on social networks is a short text message represented as

$$x_k = (u_k, W_k, t) \tag{1}$$

where W_k is the keywords set in x_k, u_k is the author of micro-text x_k, and t is its posting time.

3) SLIDING WINDOW TEXT

Sliding window text is defined as a set of text on $(t, t + p]$:

$$SW_t = TS_t^{t+p} \tag{2}$$

where t denotes time, p denotes time interval, time series is divided by $(t, t+p]$. The principle of sliding window is shown in Figure 1.

Assuming that v is the speed of the text sequence, S_t is the total amount of SW_t, S_t can be calculated by (3):

$$S_t = v * p \tag{3}$$

When $p = 1$ (unit time interval), $v = 1$ (one text per unit time), then $S_t = 1$, indicating that only one piece of text enters the time window in the unit time interval, which is the classic single-pass process. If p is a constant, SW_t is a steady sliding time window. If p is mutable, SW_t is a dynamic window with higher flexibility and complexity.

B. BURST TOPIC

1) BURST TOPIC

A burst topic is defined as a topic with peak popularity above 20% of its total popularity [2].

In this paper, Twitter dataset "tweet7" [50] collected by Yang and Leskovec is tested. As shown in Figure 2, the horizontal axis represents the total popularity obtained by the hashtag, and the vertical axis represents the total number of hashtags whose total popularity exceeds the corresponding horizontal axis value. The total popularity of these hashtags is subject to a power-rate distribution, with most hashtags get a small total popularity, and only a few hashtags gain a lot of popularity.

As shown in Figure 3, the horizontal axis represents the peak amount, and the vertical axis represents the number of hashtags whose peak amount exceeds the corresponding horizontal axis value. Most burst hashtags peaked in the first 5 hours, and only a few burst hashtags peaked above 1000.

TABLE 1. Notations used in this paper.

Notation	Description
TS	Text sequence
$x_k = (u_k, W_k, t)$	The kth arriving text x_k, micro-text author u_k, keywords set W_k *in* x_k and posting time t
p	Time interval
v	Speed of text sequence
SW_t	Sliding window text set
S_t	Total amount of SW_t
w_i^k	The ith keyword of x_k
m_i^k	Frequency of w_i^k
vec_i^k	Word vector of w_i^k
C	Similarity matrix of x_k and x_{k-1}
D	Keyword offset matrix
L	The latest texts set of all topic sequences
L'	The copy of L before t - G×p time, G is a positive integer
z_k	Topic and its texts
Z	Topics set
TP_k	All related texts set with z_k
l_k	Total number of TP_k
lifet_k	Lifetime of z_k
$\boldsymbol{\alpha}_k$	Initial weight vector of x_k
$\boldsymbol{\beta}_k$	Updated weight vector of x_k

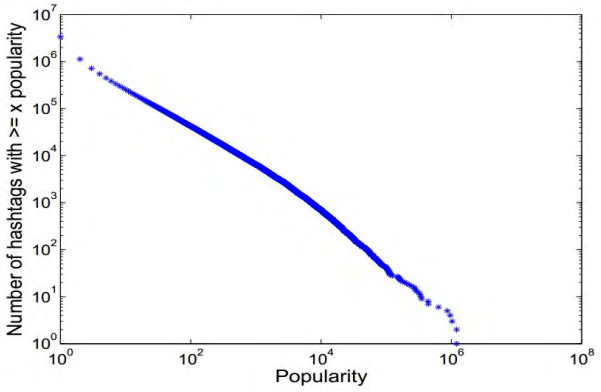

FIGURE 2. Popularity distribution of hashtags in tweet7 dataset.

As shown in Figure 4, this paper logarithmically rescales the horizontal axis in this figure due to the large variances present among active periods of different hashtags (notice that they range from one to several thousand hours). For each observed value on the blue line, the empirical cumulative distribution tells the fraction of hashtags for which the durations of active periods are shorter than this value. For more than 15% of hashtags, their durations of active periods are shorter than 24 hours. For about 60% of hashtags, their durations are shorter than 100 hours. For about 20% of hashtags, their durations are longer than a week.

The number of texts on most topics is small, and these topics are not popular. It can be said that they are noises. In addition, the number of burst topics that are popular is

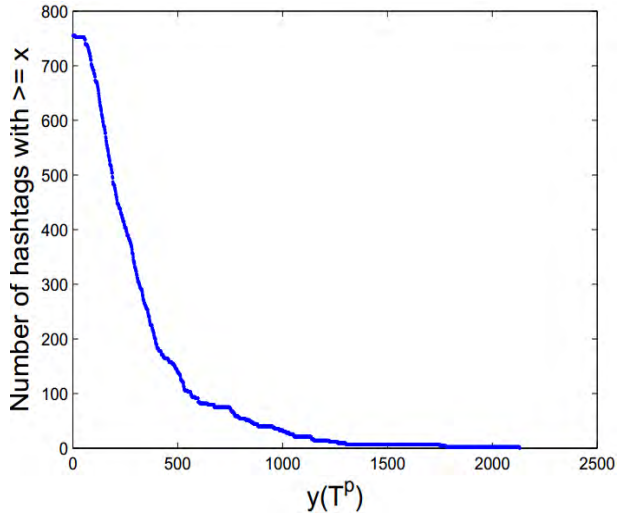

FIGURE 3. Distribution of burst hashtags.

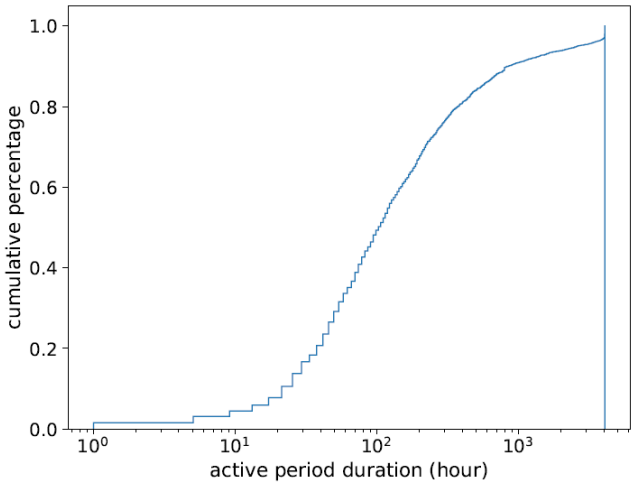

FIGURE 4. The distribution of active periods.

small, but these burst topics have a great influence on the public and society. Therefore, most topics that are not popular can be ignored. Detection and tracking the few burst topics are more important.

IV. MODEL

In order to detect and track burst hotspots on social networks, this paper first extracts keywords according to the characteristics of short text. Then, according to the keywords, the corpus is clustered to discover and track the topics. Finally BHDDT (burst hotspots dynamic detection and tracking) is implemented on Hadoop platform to improve efficiency.

A. KEYWORD EXTRACTION

The number of keywords in a short text is very small, so each keyword has a great contribution to the description of the text. If extracted keywords are not accurate enough, there will be a large error, which will have a great effect on clustering results. Therefore, extracted keywords should reflect the relevant information contained in the text as much as possible. Related keywords must be used to identify the event features such as "What is it?", "Who are relevant?", "When does it happen?", and "Where is it?". Consequently, relevant characters, places and event description are extracted as the keywords of a text.

A typical microblog text generally contains nickname of microblog author, @someone, and specific content (related event information and involved people). Word segmentation methods do not deal with nicknames in texts very well, so this paper extracts character information of the text with "@" followed by "space" or "@" followed by ":" as dividing line (identifier). Geographical names can be extracted directly according to the part of speech provided by word segmentation method. Since event keyword is after verb, this paper takes verb-like word in each sentence as the beginning of event keyword, and then uses noun-like word within certain distance as event keyword. The length in this algorithm is the distance between the verb-like word and the noun-like word. To ensure they are a group and avoid other situations, the upper limit of length is set to 3. The details of keyword extraction algorithm are shown in Algorithm 1.

Algorithm 1 Keyword Extraction Algorithm

Input: WS //WS is the list of x_k after word segmentation
Output: $W_k = \{w_1^k, w_2^k, \ldots, w_M^k\}$

```
for w in WS
  if the part of speech of w is nr/nr2/nrj/nrf/nz
    add w to W_k; // w is the real name
  else if w behind @
    add w to W_k; // w is the nickname
  else if the part of speech of w is ns/nsf
    add w to W_k; // w is the place name
  else if w is noun after verb and length <= 3
    //length is the distance to the verb
    add w to W_k; // w is the event keyword
  end if
end for
return W_k;
```

B. ASSOCIATION MODEL

Texts belonging to the same topic have very strong correlation on content. In order to overcome the sparsity problem existing in short texts processing on social networks, this paper constructs similarity matrix through two texts of the temporally nearest neighbors and determines the correlation between each other by proposed association model.

Assuming current processing text is $x_k = (u_k, W_k, t)$, $W_k = \{w_1^k : m_1^k, w_2^k : m_2^k, \ldots, w_M^k : m_M^k\}$, where w_i^k is ith keyword of x_k and m_i^k is the frequency of w_i^k, L is the set of latest text in all topic sequences and sorted by the time in descending order, $x' = (u', W', t)$, $W' = \{w_1' : m_1', w_2' : m_2', \ldots, w_N' : m_N'\}$, and $x' \in L$. Here, each keyword is represented with word vector (100 dimensions) by

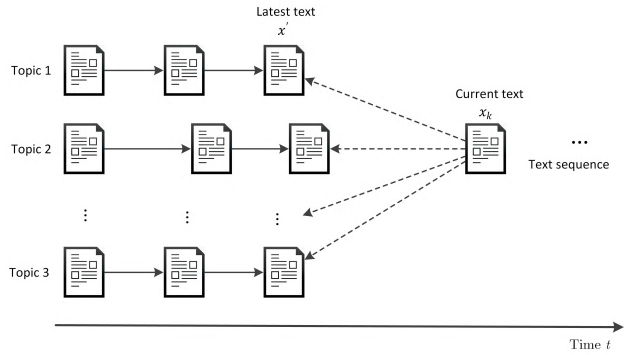

FIGURE 5. Clustering process of texts.

Word2Vec [51]. vec_i^k is word vector of w_i^k, and vec_j' is word vector of w_j'. The similarity is calculated by cosine distance. $C = (c_{ij})_{\mathrm{M}\times\mathrm{N}}$ is the similarity matrix of W_k and W'. c_{ij} can be calculated by (4).

$$c_{ij} = vec_i^k \bullet vec_j' \tag{4}$$

According to the similarity matrix C, the offset of each keyword within x_k can be known. In order to quantify the offset of each keyword, if some elements of C exceed threshold λ, these elements are set to 1, and other elements are set to 0. Keyword offset matrix $D = (d_{ij})_{\mathrm{M}\times\mathrm{N}}$ can be calculated by (5).

$$d_{ij} = \begin{cases} 1, & if\ c_{ij} = \max(C[i][*]) \geq \lambda \\ 0, & if\ c_{ij} = \max(C[i][*]) < \lambda\ or\ c_{ij} < \max(C[i][*]) \end{cases} \tag{5}$$

where $C[i][*]$ denotes the ith row of C, and threshold $\lambda \in (0, 1]$ is used to measure deviation degree of each keyword.

If all elements in one line are 0, it means that the keyword has small similarity with all keywords of previous text, and the keyword can be judged as a noise. By multiplying associated topic text weight vector, primary keywords are retained, and noise is removed. Then the weights of primary keywords are reassigned, which enhances the primary keywords and reduces the effect of noise.

In order to judge the relevance of current text and existing topics, the relevance degree of x_k and x' can be calculated by (6).

$$sim_k = n_k/M \tag{6}$$

where n_k is the number of non-zero lines in D.

The clustering process is shown in Figure 5. If there is more than half of keywords in one text are similar with the keywords of another text and these similarities exceed the threshold, the two texts are associated; otherwise the two texts are not associated. If $\mathrm{sim}_k \geq 0.5$, x_k is associated with $x'.x'$ in L is replaced with x_k and added to the topic that contains x'. Here, two main factors are considered. The first factor is time. We select the temporal nearest topic whose similarity is greater than the threshold. The second is to reduce computation and improve performance. Otherwise, x_k and x' will not be associated. According to the principle of temporal nearest neighbors, x' traverses L in time sequence until all texts in L are traversed. If x_k is not correlated with any text in L, x_k is used as the initial text of new candidate topic. The details of algorithm clustering are shown in Algorithm 2.

Algorithm 2 Clustering algorithm

Input: $TS = \{x_1, x_2, \ldots, x_k, \ldots\}, L$
Output: Z // topics set Z

```
keywords of TS are converted into word vectors by Word2Vec;
for each x_k in TS
  sort(L); //Sort L in descending order of time
  flag = 0; //Mark the result after traversing L
  for x' in L
    Calculate matrix C by (4);
    Calculate matrix D is by (5);
    Calculate n_k is;
    sim_k = n_k/M;
    if sim_k >= 0.5
      flag = 1;
      Add x_k to the topic that contains x' in Z;
      x' is replaced with x_k in L;
      break;
    end if
  end for
  if flag = 0
    add x_k to L;
    x_k is used as the initial text of new topic in Z;
  end if
end for
return Z;
```

C. DYNAMIC UPDATE

In the massive social networks short texts, there are many types and numbers of topics. The text size of each topic is unevenly distributed, and there are a large number of isolated texts and noise words. In order to eliminate isolated points in the topics set Z, prevent the scale of L from increasing indefinitely, and ensure algorithm's stability, efficiency and accuracy, Z and L are regularly updated. The topic sequence $TP_k \subseteq Z$, TP_k is all related texts set of z_k, lifet_k is lifetime of z_k, and l_k is total number of texts in TP_k. If $\mathrm{lifet}_k > \tau$ hour and $l_k < \eta$, delete TP_k from Z and x_k from L, where x_k is the latest text in TP_k. L' is the copy of L before the t-G$\times p$ time, t is the time of current processing text and G is a positive integer. To control the time interval, we set $p = 1$ minute and $\mathrm{G} = 5$, that is, every 5 minutes detect whether topics have new text. If $L \cap L'$ is not empty, it means that some texts in L are not updated continuously, that is, some topics have no new text coming. Then update

$L = L - L \cap L'$, move these topics out of TP, and save them in the external storage. The details of updating algorithm are shown in Algorithm 3.

Algorithm 3 Updating Algorithm

Input: Z, TP_k, L, L', z_k
Output: Updated Z, L

```
for TP_k in TP
    if lifet_k > 1 hour and l_k < 10
    //candidate topic has too little text in a long time
        delete TP_k in Z; // TP_k is noise or isolated point
        delete x_k in L; // x_k ∈ TP_k and x_k ∈ L
    end if
    if L ∩ L' != Φ
//some topics have no new text coming continuously
        L = L − L ∩ L'; //Update L
        for x_k in L ∩ L'
            save TP_k to the external file z_k.txt;
            delete TP_k from Z; //Update Z
        end for
    end if
end for
return Z and L;
```

D. NOISE REDUCTION

Any text that is unrelated to context and final output can be considered as noise. Text is unstructured data and has a variety of noise. Noise will increase time and memory cost, and more seriously, precision will decrease. Therefore, it is very important to eliminate noise and minimize the adverse effects of noise.

In this paper, stopwords and irrelevant characters are filtered out by stopword list (1900 stopwords) in preprocessing, and noise is further filtered by the association model proposed in section IV-B.

Current processing text is $x_k = (u_k, W_k, t)$, $W_k = \{{w_1}^k : {m_1}^k, {w_2}^k : {m_2}^k, \ldots, {w_M}^k : {m_M}^k\}$, where w_i^k is ith keyword of x_k and m_i^k is the frequency of w_i^k. The initial weight vector of x_k is $\boldsymbol{\alpha}_k = [{\boldsymbol{\alpha}_1}^k, {\boldsymbol{\alpha}_2}^k, \ldots, {\boldsymbol{\alpha}_M}^k]$, and the initial weight $\boldsymbol{\alpha}_i^k$ of w_i^k can be calculated by (7).

$$\alpha_i^k = \frac{m_i^k}{\sum_{i=1}^{M} m_i^k}, \quad and\ \alpha_i^k \in [0, 1],\ \sum_{i=1}^{M} \alpha_i^k = 1 \tag{7}$$

The updated weight vector of x_k is $\boldsymbol{\beta}_k = [{\boldsymbol{\beta}_1}^k, {\boldsymbol{\beta}_2}^k, \ldots, \boldsymbol{\beta}_M^k]$, and the updated weight vector of x' is $\boldsymbol{\beta}' = [\boldsymbol{\beta}_1', \boldsymbol{\beta}_2', \ldots, \boldsymbol{\beta}_N']$, D^T is the transposed matrix of D and $D^T[*][i]$ represents the ith column of D^T. The updated weight $\boldsymbol{\beta}_i^k$ of w_i^k in x_k can be calculated by (8).

$$\beta_i^k = \frac{\alpha_i^k \times [\beta_1', \beta_2', \ \ldots\ , \beta_N'] \times D^T[*][i]}{\sum_{j=1}^{M} \alpha_j^k \times [\beta_1', \beta_2', \ \ldots\ , \beta_N'] \times D^T[*][i]} \tag{8}$$

In order to highlight important keywords and suppress secondary keywords, the keyword weight of x_k is adjusted by the keyword weight of x'. In addition, if all the elements of $D[i][*]$ are equal to zero, where $D[i][*]$ is the ith row of D, w_i^k of x_k is noise. Then delete w_i^k from x_k and renormalize all keyword weights of x_k.

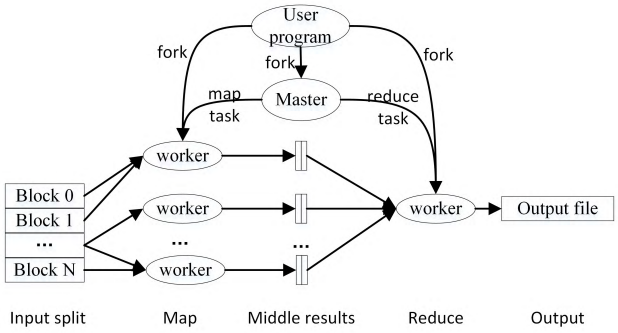

FIGURE 6. Distributed processing mode.

E. DISTRIBUTED PROCESSING

In this paper, Hadoop distributed platform and Map/Reduce mode are used to preprocess massive short texts. The text data is assigned to each distributed node for preprocessing. As shown in Figure 6, the text sequence is divided into N blocks; NameNode and JobTracker are deployed in Master node; DataNode and TaskTracker are deployed in worker nodes. The key value of Map output is set to be the same, and all Map output is controlled in a Reduce function. Get the keywords from each text first, and then cluster the texts. The algorithm is implemented as follows:

1) JobTrack is responsible for the assignment of tasks and equalization scheduling. The text sequence is divided into N pieces of SW_t data according to the time interval p, and the task data is distributed to each node (TaskTracker);
2) The Map function obtains the key value pairs of <line number, line content>, extracts keywords of line content, obtains keywords of the texts, and outputs the key value pair of <keyword, text ID_ keyword>;
3) The combine process merges the key-value pairs that have the same key. Since all keys of the map output are keyword, all the map output will be merged into the key-value pairs like <keyword, <text1 keyword, text2 keyword, ...>>, and obtains N key-value pair sequences of <key, list<value>>;
4) In the Reduce process, a reduce function gets all the map output and merges them into a text sequence. The text $x_k = (u_k, W_k, t)$, $W_k = \{{w_1}^k : {m_1}^k, {w_2}^k : {m_2}^k, \ldots, {w_M}^k : {m_M}^k\}$, $TS = \{x_1, x_2, \ldots, x_k, \ldots\}$. Then the hotspots are detected and tracked in the text sequence.

V. EXPERIMENTS

Experiments in this paper comprehensively evaluate the performance of BHDDT both in stand-alone mode and Map/Reduce mode from running time, speed-up ratio,

precision, recall, F1-score and NMI. Experimental environment is a Hadoop platform with 11 nodes. Each server has 16*Intel(R) Xeon(R) CPU E7320 @ 2.13GHz, 16GB memory, 1GB cache and 2GB swap area. The operating system is 64-bit Linux operating system (Red Hat 4.1.2-42).

A. METRICS FOR EVALUATING ALGORITHM QUALITY

Precision P, recall R, F1-score and NMI are used to evaluate the performance, as shown in (9), (10), (11) and (12), respectively. The precision rate and recall rate are taken as the average of precision rate and recall rate under each topic category.

$$P = \text{TP}/\ (\text{TP+FP}) \tag{9}$$

$$R = \text{TP}/\ (\text{TP+FN}) \tag{10}$$

$$F1 = 2 * P * R/(P + R) \tag{11}$$

where TP is the number of the true positives, FP is the number of the false positives, and FN is the number of the false negatives.

1) NMI

Suppose $A = \{A_1, A_2, \ldots, A_k\}$ is a set of k clusters contained in a dataset and $B = \{B_1, B_2, \ldots, B_k\}$ is a set of k clusters obtained by a specific algorithm. NMI is defined as

$$NMI(A, B) = \frac{-2 \sum_{i=1}^{k} \sum_{j=1}^{k} n_{ij} \log \frac{n \cdot n_{ij}}{n_i^A \cdot n_j^B}}{\sum_{i=1}^{k} n_i^A \log \frac{n_i^A}{n} + \sum_{j=1}^{k} n_j^B \log \frac{n_j^B}{n}} \tag{12}$$

where n denotes the total number of texts, n_{ij} is the number of data points in the ground truth cluster A_i that are assigned to the computed cluster B_j, n_i^A is the number of data points in the ground truth cluster A_i, and n_j^B is the number of data points in the computed cluster B_j.

B. PRELIMINARIES TEST

Short texts in social networks have the characteristics of fast update and large quantity. If they are processed in stand-alone mode, it cannot meet the demand when the amount of data is large, because of CPU processing power and memory size limitations. The clustering program in stand-alone mode is tested and the results are shown in Figure 7. 60%-70% of the running time is consumed in preprocessing such as word segmentation, denoising, feature extraction, etc. The time involved in the specific clustering operation is only a small part.

We test 434 topics, and the results are shown in Figure 8. It can be seen that most of the topics can reach 10 within 1 hour. Compared with the initial stage of topics, we pay more attention to the outbreak stage. Although it will miss a few scattered texts from initial to outbreak, all texts can be collected and analyzed after topics burst. As a result, we set the parameters of BHDDT as follow, $\lambda = 0.5$, $\tau = 1$, $\eta = 10$.

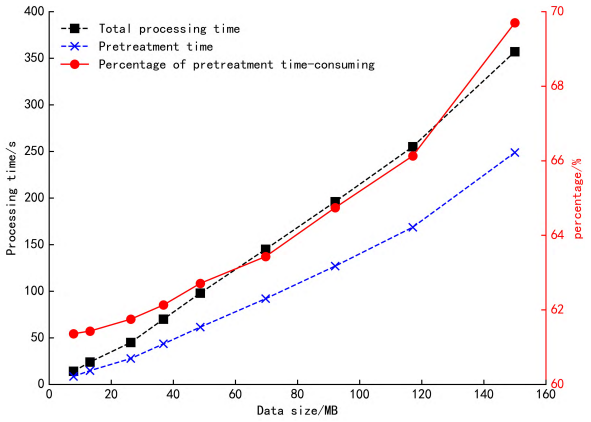

FIGURE 7. Pretreatment time test in stand-alone mode.

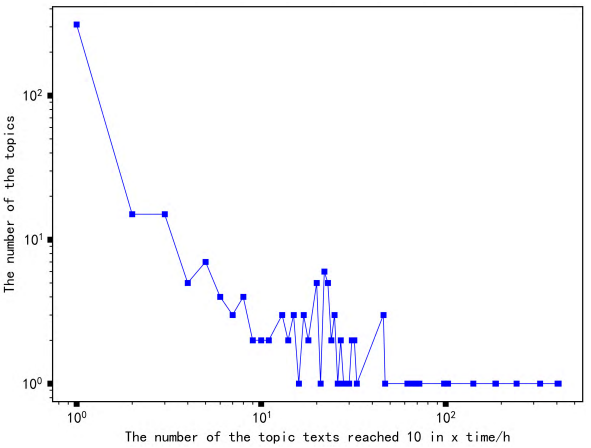

FIGURE 8. The distribution of the topics reaching the threshold.

C. PERFORMANCE EVALUATIONS

We compared our proposed method BHDDT with existing state-of-the-art algorithms including spherical K-means (spk -means) [34], K-means [16], SP-WC [35], SP-NN [35], LDA [47], GSDMM [49] and WordCom [44].

1) DATASETS

Experimental datasets for evaluation include Sina microblogs dataset (including 10 topic categories and 6000 texts that have been labeled), DBLP titles dataset (including 5 labels Data Mining, Affective Computing, Database, NLP and Parallel Computing) and Twitter_30 (30 classes and each class contains at least 50 short texts).

2) PARAMETER SETTINGS

First of all, the parameter settings are set for Sina dataset. For LDA, we used the same parameter settings as in [48], where parameters were tuned via grid search for short text corpora, $\alpha = 0.05$ and $\beta = 0.01$. For GSDMM, we set $\alpha = 0.1$ and $\beta = 0.1$ as was done in [49]. For WordCom, we set $\alpha = 0.5$ and $\beta = 0.05$ as in [44]. For SP-NN and SP-WC, the similarity threshold is fixed as 0.53, t_{sim} is fixed as 0.53 and the window size is fixed at 7 as was done in [35]. For K-means and spk-means, we set K = 10, and initial centers

TABLE 2. Result evaluation on sina microblogs dataset.

Noise texts	0			1000			2000		
Algorithm	*P*	*R*	*F1*	*P*	*R*	*F1*	*P*	*R*	*F1*
BHDDT	**0.8639**	0.8378	0.8506	**0.8201**	**0.7903**	**0.8049**	**0.7943**	**0.7782**	**0.7862**
WordCom	0.8337	0.8217	0.8277	0.6671	0.6723	0.6697	0.6311	0.5471	0.5861
SP-WC	0.7634	0.7556	0.7595	0.4417	0.3289	0.3770	0.2814	0.2743	0.2778
SP-NN	0.8013	0.7976	0.7994	0.4782	0.3647	0.4138	0.3017	0.2978	0.2997
K-means	0.6894	0.7071	0.6981	0.5614	0.4931	0.5250	0.3732	0.4394	0.4036
spk-means	0.7356	0.7289	0.7322	0.5571	0.5623	0.5597	0.4413	0.4677	0.4541
LDA	0.8287	0.8268	0.8277	0.7341	0.6712	0.7012	0.6358	0.6451	0.6404
GSDMM	0.8476	**0.8734**	**0.8603**	0.7733	0.7884	0.7808	0.7132	0.7213	0.7172

are randomly selected. For BHDDT in stand-alone mode, we set the parameters of BHDDT as described in section V-B and select 10 initial texts as the initial centers from 10 topic categories respectively.

Secondly, the parameter settings are set for Twitter_30. The parameters of LDA, SP-WC, SP-NN, GSDMM and WordCom remain the same. In addition, for K-means and spk-means, we set K = 30 and initial centers are randomly selected. For BHDDT in stand-alone mode, we set the parameters of BHDDT as described in section V-B and select 30 initial texts as the initial centers separately from 30 topics.

Thirdly, the parameter settings are set for DBLP titles dataset. The parameters remain the same for LDA, SP-WC, SP-NN, GSDMM and WordCom. In addition, for K-means and spk-means, we set K = 5 and initial centers are randomly selected. For BHDDT in stand-alone mode, we set the parameters of BHDDT as described in section V-B and select 5 initial titles as the initial centers separately from the 5 labels' titles.

The results of experiment on Sina microblogs dataset are shown in Table 2 We marked the best performing algorithm in bold. First, when there is no noise, BHDDT has the highest precision, and recall is lower than GSDMM. It is because BHDDT filters out some texts that have low popularity and edge points. GSDMM has the highest recall and F1, but could not converge at the exact number of clusters. Secondly, as shown in Table 1 and Figure 9, when add 1000, 2000 and 3000 noise texts to the Sina microblogs dataset, BHDDT has higher stability, because BHDDT only pay attention to the 10 topics, ignoring the noise. On the contrary, the performance of SP-WC, SP-NN, K-means and spk-means decline quickly, because they are very sensitive to the noise texts and hardly work.

The results of experiment on Twitter_30 dataset are shown in Table 3 It can be seen that BHDDT has the best performance. There are a lot of internet slangs, polysemy and approximation terms in Twitter_30, which are the shortcomings of term co-occurrence methods. And each short text within Twitter_30 contains few words and the vast majority occurs only once. Therefore, WordCom hardly works and the performance of GSDMM is not so good. In addition, SP-WC, SP-NN, K-means, spk-means and LDA have poor

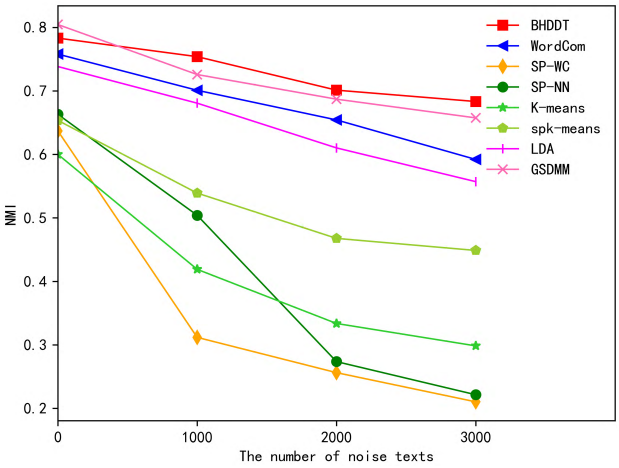

FIGURE 9. The performance affected by noise texts.

TABLE 3. Result evaluation on Twitter_30 dataset.

Algorithm	Precision	Recall	F1-score	NMI
BHDDT	**0.8139**	**0.8073**	**0.8106**	**0.7633**
WordCom	0.4832	0.4987	0.4908	0.4512
SP-WC	0.6775	0.6834	0.6804	0.6587
SP-NN	0.7156	0.7241	0.7198	0.6692
K-means	0.6359	0.6288	0.6323	0.6045
spk-means	0.7312	0.7051	0.7179	0.6639
LDA	0.6984	0.7143	0.7063	0.6702
GSDMM	0.7691	0.7784	0.7737	0.7356

performance on Twitter_30 dataset. BHDDT is specially designed to handle large-scale short text stream on social networks, where BHDDT has the best performance.

The results of experiment on DBLP titles dataset are shown in Table 4. DBLP is an academic paper dataset that use normative terms, so the performance of all methods is quite good. The performance of BHDDT, WordCom and GSDMM is good. Especially, BHDDT has the best performance.

D. PROCESSING TIME TEST

The processing time of BHDDT in stand-alone mode and Map/Reduce mode (6 distributed nodes) is tested. The time-consuming of stand-alone BHDDT, Map/Reduce

TABLE 4. Result evaluation on dblp titles dataset.

Algorithm	Precision	Recall	F1-score	NMI
BHDDT	**0.8839**	**0.8773**	**0.8806**	**0.8133**
WordCom	0.8737	0.8677	0.8707	0.8081
SP-WC	0.7718	0.7811	0.7764	0.6433
SP-NN	0.7621	0.7713	0.7667	0.6975
K-means	0.7133	0.7062	0.7097	0.6692
spk-means	0.7356	0.7589	0.7471	0.6815
LDA	0.8087	0.8168	0.8127	0.7333
GSDMM	0.8676	0.8534	0.8604	0.7958

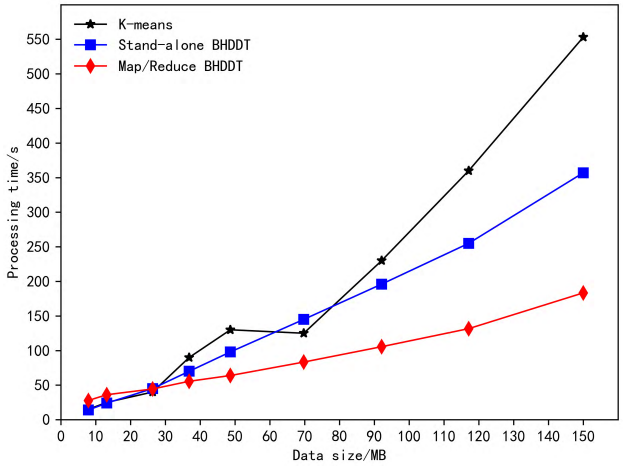

FIGURE 10. Processing time of different methods.

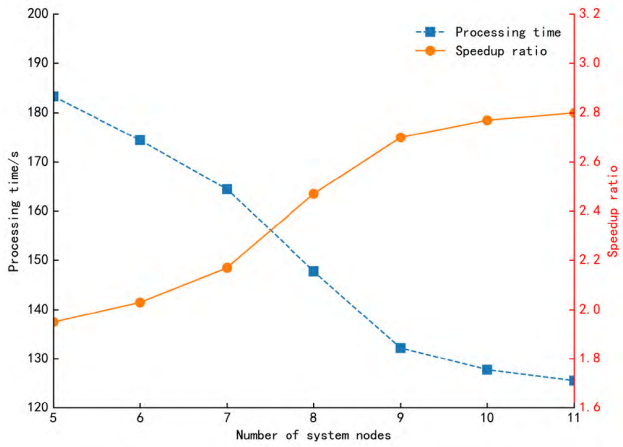

FIGURE 11. Scalability performance test on Hadoop.

BHDDT and K-means to process different sizes of data are compared as shown in Figure 10.

In the case of small amount of data, BHDDT takes less time in stand-alone mode than in Map/Reduce mode. This is because Hadoop platform itself is designed to handle big data, but it cannot take advantage of it when the amount of data is small. On the contrary, the consumption of network transmission and task scheduling increases time-consuming. However, as the size of data increases, the time-consuming of BHDDT increases linearly, while the time-consuming of K-means grows in an unstable state. This is mainly because the performance of K-means has a great relationship with the selection of initial points.

In the case of large-scale data, the time-consuming of K-means increases sharply. Compared with K-means, the time-consuming of BHDDT is low, especially Map/Reduce BHDDT. This shows that K-means cannot solve the problem of big data clustering well, and the time-consuming of BHDDT increases linearly as the amount of data increases, which indicates that BHDDT has better stability and scalability. In the case of large-scale data, Hadoop platform takes advantage of its potential and has a high throughput rate for big data.

E. SCALABILITY TEST

The scalability of Map/Reduce BHDDT is tested. The speedup ratio is the ratio of the time it takes for the same task to run in a single-processor system and a parallel processor system, measuring the performance and effectiveness of parallel system or program parallelization. Time-consuming and speedup experiments of Map/Reduce BHDDT for processing 150MB text sets (approximately 400,000 microblogs) using different number of nodes, shown in Figure 11.

It can be seen that time-consuming decreases with the increasing of the number of nodes, and speedup ratio increases with the number of nodes, indicating that Map/Reduce BHDDT has good scalability.

VI. CONCLUSIONS

This paper proposes a distributed method of dynamic detection and tracking burst hotspots on social networks. The problem of sparsity is solved by judging the relevance of current text between previous texts by the keyword similarity matrix of adjacent text. The topic detection and tracking technology is integrated into the same system. By analyzing the algorithm's bottleneck, BHDDT algorithm is parallelized based on Map/Reduce, which greatly improves the efficiency of the program operation and solves the problem that the clustering algorithm degrades with the increase of data volume. According to our comparison experiments, BHDDT has better performance in terms of running time, precision, recall rate, NMI and scalability than other algorithms. In this paper, BHDDT only implements partial parallelization. Although the clustering part is still handled by a single machine, the performance of BHDDT is greatly improved. The processing power of each node of Hadoop is not well utilized, and there is room for further improvement. Therefore, BHDDT will be further improved and optimized. In future work we will carry out evolutionary prediction analysis on short text sequences of social networks.

REFERENCES

[1] Y. Hu, C. Hu, S. Fu, M. Fang, and W. Xu, "Predicting key events in the popularity evolution of online information," *PLoS one*, vol. 12, no. 1, 2017, Art. no. e0168749.

[2] R. Crane and D. Sornette, "Robust dynamic classes revealed by measuring the response function of a social system," *Proc. Nat. Acad. Sci. USA*, vol. 105, no. 41, pp. 15649–15653, 2008.
[3] S. Kong *et al.*, "Predicting bursts and popularity of hashtags in real-time," in *Proc. 37th Int. ACM SIGIR Conf. Res. Develop. Inf. Retr.*, 2014, pp. 927–930.
[4] S. Kong, Q. Mei, L. Feng, and Z. Zhao, "Real-time predicting bursting hashtags on Twitter," in *Proc. Int. Conf. Web-Age Inf. Manage.* Cham, Switzerland: Springer, 2014, pp. 268–271.
[5] H. Kwak, C. Lee, H. Park, and S. Moon, "What is Twitter, a social network or a news media?" in *Proc. 19th Int. Conf. World Wide Web*, 2010, pp. 591–600.
[6] X. Luo *et al.*, "Short-term wind speed forecasting via stacked extreme learning machine with generalized correntropy," *IEEE Trans. Ind. Informat.*, vol. 14, no. 11, pp. 4963–4971, Nov. 2018.
[7] M. Chen, Y. Li, X. Luo, W. Wang, L. Wang, and W. Zhao, "A novel human activity recognition scheme for smart health using multilayer extreme learning machine," *IEEE Internet Things J.*, to be published. doi: 10.1109/JIOT.2018.2856241.
[8] S. Aslam. *Twitter by the Numbers: Stats, Demographics and Fun Facts*. Accessed: Nov. 31, 2017. [Online]. Available: https://www.omnicoreagency.com/twitter-statistics
[9] H. L. Nguyen, Y.-K. Woon, and W.-K. Ng, "A survey on data stream clustering and classification," *Knowl. Inf. Syst.*, vol. 45, no. 3, pp. 535–569, 2015.
[10] J. Huang, M. Peng, H. Wang, J. Cao, W. Gao, and X. Zhang, "A probabilistic method for emerging topic tracking in microblog stream," *World Wide Web*, vol. 20, no. 2, pp. 325–350, 2017.
[11] W. Xie, F. Zhu, J. Jiang, E.-P. Lim, and K. Wang, "TopicSketch: Real-time bursty topic detection from Twitter," in *Proc. 13th Int. Conf. Data Mining*, Dec. 2013, pp. 837–846.
[12] X. H. Li, T. He, H. Ran, and X. Lu, "A novel graph partitioning criterion based short text clustering method," in *Proc. Int. Conf. Intell. Comput.* Cham, Switzerland: Springer, 2016, pp. 338–348.
[13] F. Beil, M. Ester, and X. Xu, "Frequent term-based text clustering," in *Proc. 8th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2002, pp. 436–442.
[14] S. A. Salloum, M. Al-Emran, A. A. Monem, and K. Shaalan, "A survey of text mining in social media: Facebook and Twitter perspectives," *Adv. Sci. Technol. Eng. Syst. J*, vol. 2, no. 1, pp. 127–133, 2017.
[15] A. Ali, J. Qadir, R. U. Rasool, A. Sathiaseelan, and A. Zwitter, "Big data for development: Applications and techniques," *Comput. Soc.*, vol. 1, no. 2, pp. 1–24, 2016.
[16] M. Capó, A. Pérez, and J. A. Lozano, "An efficient approximation to the K-means clustering for massive data," *Knowl. Based Syst.*, vol. 117, pp. 56–69, Feb. 2017.
[17] P. Arora and S. Varshney, "Analysis of K-means and K-medoids algorithm for big data," in *Proc. 1st Int. Conf. Inf. Secur. Privacy (ICISP)*, vol. 78, Dec. 2015, pp. 507–512.
[18] R. T. Ng and J. Han, "CLARANS: A method for clustering objects for spatial data mining," *IEEE Trans. Knowl. Data Eng.*, vol. 14, no. 5, pp. 1003–1016, Sep. 2002.
[19] L. M. Abualigah and A. T. Khader, "Unsupervised text feature selection technique based on hybrid particle swarm optimization algorithm with genetic operators for the text clustering," *J. Supercomputing*, vol. 73, no. 11, pp. 4773–4795, 2017.
[20] X. Hu, N. Sun, C. Zhang, and T. S. Chua, "Exploiting internal and external semantics for the clustering of short texts using world knowledge," in *Proc. 18th ACM Conf. Inf. Knowl. Manage.*, 2009, pp. 919–928.
[21] M. Steinbach, M. S. G. Karypis, and V. Kumar, "A comparison of document clustering techniques," in *Proc. KDD Workshop Text Mining*, vol. 400, no. 1, 2000, pp. 525–526.
[22] R. E. Thomas and S. S. Khan, "Co-Clustering with side information for text mining," in *Proc. Int. Conf. Data Mining*, Mar. 2016, pp. 105–108.
[23] S. S. Bhanuse, S. D. Kamble, and S. M. Kakde, "Text mining using metadata for generation of side information," *Procedia Comput. Sci.*, vol. 78, pp. 807–814, 2016.
[24] M. Hahsler and M. Bolaños, "Clustering data streams based on shared density between micro-clusters," *IEEE Trans. Knowl. Data Eng.*, vol. 28, no. 6, pp. 1449–1461, Jun. 2016.
[25] G. Karypis, E.-H. Han, and V. Kumar, "Chameleon: Hierarchical clustering using dynamic modeling," *Computer*, vol. 32, no. 8, pp. 68–75, Aug. 1999.
[26] E. Schubert, J. Sander, M. Ester, H. P. Kriegel, and X. Xu, "DBSCAN revisited, revisited: Why and how you should (still) use DBSCAN," *ACM Trans. Database Syst.*, vol. 42, no. 3, p. 19, 2017.
[27] T. Gao, A. Li, and F. Meng, "Research on data stream clustering based on FCM algorithm," in *Proc. 5th Int. Conf. Inf. Technol. Quant. Manage. (ITQM)*, vol. 122, Dec. 2017, pp. 595–602.
[28] H. Rehioui, A. Idrissi, M. Abourezq, and F. Zegrari, "DENCLUE-IM: A new approach for big data clustering," in *Proc. 7th Int. Conf. Ambient Syst., Netw. Technol. (ANT)*, vol. 83, pp. 560–567, May 2016.
[29] K. S. Jones, "A statistical interpretation of term specificity and its application in retrieval," *J. Documentation*, vol. 28, no. 1, pp. 11–21, 1972.
[30] Y. Cong, Y.-B. Chan, and M. A. Ragan, "A novel alignment-free method for detection of lateral genetic transfer based on TF-IDF," *Sci. Rep.*, vol. 6, Jul. 2016, Art. no. 30308.
[31] L. Guo, C. J. Vargo, Z. Pan, W. Ding, and P. Ishwar, "Big social data analytics in journalism and mass communication: Comparing dictionary-based text analysis and unsupervised topic modeling," *J. Mass Commun. Quart.*, vol. 93, no. 2, pp. 332–359, 2016.
[32] M. Allahyari *et al.* (2017). "A brief survey of text mining: Classification, clustering and extraction techniques." [Online]. Available: https://arxiv.org/abs/1707.02919
[33] J. Xu *et al.*, "Self-taught convolutional neural networks for short text clustering," *Neural Netw.*, vol. 88, pp. 22–31, Apr. 2017.
[34] I. S. Dhillon and D. S. Modha, "Concept decompositions for large sparse text data using clustering," *Mach. Learn.*, vol. 42, nos. 1–2, pp. 143–175, 2001.
[35] D. Shen, Q. Yang, J. T. Sun, and Z. Chen, "Thread detection in dynamic text message streams," in *Proc. 29th Annu. Int. ACM SIGIR Conf. Res. Develop. Inf. Retr.*, 2006, pp. 35–42.
[36] X. Luo *et al.*, "Towards enhancing stacked extreme learning machine with sparse autoencoder by correntropy," *J. Franklin Inst.*, vol. 355, no. 4, pp. 1945–1966, Mar. 2018, doi: 10.1016/j.jfranklin.2017.08.014.
[37] S. Banerjee, K. Ramanathan, and A. Gupta, "Clustering short texts using Wikipedia," in *Proc. Int. ACM SIGIR Conf. Res. Develop. Inf. Retr.*, 2007, pp. 787–788.
[38] L. Wang, Y. Jia, and W. Han, "Instant message clustering based on extended vector space model," in *Proc. Int. Symp. Intell. Comput. Appl.*, vol. 4683, in Lecture Notes in Computer Science, 2007, pp. 435–443.
[39] J. Tang, X. Wang, H. Gao, X. Hu, and H. Liu, "Enriching short text representation in microblog for clustering," *Frontiers Comput. Sci.*, vol. 6, no. 1, pp. 88–101, 2012.
[40] T. Mikolov, K. Chen, G. Corrado, and J. Dean. (2013). "Efficient estimation of word representations in vector space." [Online]. Available: https://arxiv.org/abs/1301.3781
[41] Q. Le and T. Mikolov, "Distributed representations of sentences and documents," in *Proc. Int. Conf. Mach. Learn.*, 2014, pp. 1188–1196.
[42] T. Kenter and R. M. De, "Short text similarity with word embeddings," in *Proc. 24th ACM Int. Conf. Inf. Knowl. Manage.*, 2015, pp. 1411–1420.
[43] C. Akimushkin, D. R. Amancio, and J. O. N. Oliveira, Jr., "Text authorship identified using the dynamics of word co-occurrence networks," *PLoS ONE*, vol. 12, no. 1, 2017, Art. no. e0170527.
[44] C. Jia, M. B. Carson, X. Wang, and J. Yu, "Concept decompositions for short text clustering by identifying word communities," *Pattern Recognit.*, vol. 76, pp. 691–703, Apr. 2018.
[45] S. Deerwester, S. T. Dumais, G. W. Fumas, T. K. Landauer, and R. Harshman, "Indexing by latent semantic analysis," *J. Amer. Soc. Inf. Sci.*, vol. 41, no. 6, pp. 391–407, 1990.
[46] T. Hofmann, "Probabilistic latent semantic indexing," *ACM SIGIR Forum.*, vol. 51, no. 2, pp. 211–218, 2017.
[47] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent Dirichlet allocation," *J. Mach. Learn. Res.*, vol. 3, pp. 993–1022, Jan. 2003.
[48] X. Yan, J. Guo, Y. Lan, and X. Cheng, "A biterm topic model for short texts," in *Proc. Int. World Wide Web Conf. Steering Committee*, 2013, pp. 1445–1456.
[49] J. Yin and J. Wang, "A Dirichlet multinomial mixture model-based approach for short text clustering," in *Proc. ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2014, pp. 233–242.
[50] J. Yang and J. Leskovec, "Patterns of temporal variation in online media," in *Proc. 4th ACM Int. Conf. Web Search Data Mining*, 2011, pp. 177–186.
[51] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Proc. Adv. Neural Inf. Process. Syst.*, 2013, pp. 3111–3119.

JIANYI HUANG received the B.S. degree in School of Computer and Communication Engineering, University of Science and Technology Beijing and is working toward the Ph.D. degree in the same department. He is also working in Beijing Key Laboratory of Knowledge Engineering for Materials Science. His main research interests are online social network analysis and artificial intelligence.

JIANJIANG LI is currently an Associate Professor at School of Computer and Communication Engineering, University of Science and Technology Beijing, China. He is also working in Beijing Key Laboratory of Knowledge Engineering for Materials Science. He received his Ph.D. degree in computer science and technology from Tsinghua University, in 2005. He was a Visiting Scholar at Temple University, from 2014 to 2015. His current research interests include parallel computing, cloud computing, parallel compilation, and big data.

YINGYING CHEN is a master student at School of Computer and Communication Engineering, University of Science and Technology Beijing, China. She is also working in Beijing Key Laboratory of Knowledge Engineering for Materials Science. She received her B.S. degree in Shandong University of Science and Technology. Her main research interest is deep learning.

JIANKUN SUN is currently pursuing the Ph.D. degree with the University of Science and Technology Beijing, Beijing, China. His current research interests include machine learning and computational intelligence.

PENG SHI is an Associate Professor at National Center for Materials Service Safety, University of Science and Technology Beijing, China. He is also working in Beijing Key Laboratory of Knowledge Engineering for Materials Science. He got his Ph.D. from Institute of Computing Technology, Chinese Academy of Science, in 2007. His interests are mainly in social network analysis, knowledge engineering, and big data application in materials science.