Noname manuscript No.
(will be inserted by the editor)

Video Topic Discovering, Tracking and Summarization from Social Media Streams

Zhao Lu* · Yu-Ru Lin* · Xiaoxia Huang · Naixue Xiong · Zhijun Fang

Received: date / Accepted: date

Abstract Nowadays, microblogging has become popular, with hundreds of millions of short messages being posted and shared every minute on a variety of topics in social media such as Facebook, Twitter and Weibo. Many of such messages contain videos that captured particular events or moments in peo-

This research was supported in part by National Key Technology Support Program (No. 2015BAH01F02), in part by Science and Technology Commission of Shanghai Municipality (No.16511102702 and No.14DZ2260800).

Zhao Lu (Corresponding author)
Department of Computer Science and Technology, Shanghai Key Laboratory of Multidimensional Information Processing, East China Normal University, 500 Dongchuan Road, 200241, Shanghai, China
Tel.: 21-54345054
Fax: 21-62233650
E-mail: zlu@cs.ecnu.edu.cn

Yu-Ru Lin (Corresponding author)
School of Information Science, University of Pittsburgh, 135 North Bellefield Ave., PA 15260 Pittsburgh, USA
Tel.: 412-624-9470
Fax: 412-624-5231
E-mail: yurulin@pitt.edu

Xiaoxia Huang
Department of Computer Science and Technology, Shanghai Key Laboratory of Multidimensional Information Processing, East China Normal University, 500 Dongchuan Road, 200241, Shanghai, China

Naixue Xiong
School of Information Technology, Jiangxi University of Finance and Economics, 169 East Shuanggang Road, Changbei, Nanchang, Jiangxi, China

Zhijun Fang
College of Electronic and Electrical Engineering, Shanghai University of Engineering Science, 333 Longteng Road, 201620, Shanghai, China

ple's life. In this work, we seek to automatically identify the video topics posted in the social media streams on Weibo. While Topic Detection and Tracking (TDT) task has been extensively studied in multimedia retrieval, automatically discovering, tracking and summarizing video topics from social media streams is still challenging due to short and noisy content, diverse and fast changing topics, and large data volume. In this paper, we propose a K-partite graph based approach to address these challenges. We introduce a K-partite graph representation to simultaneously model the relationships among videos contained in the Weibo streams, their textural features and visual features. We propose a novel joint clustering algorithm to capture global structure of the K-partite graph in a "relation cluster network" (RCN) where latent, meta-nodes are added to the network to represent video clusters. Based on this network we propose methods for tracking and summarizing the videos in streams through fusing various types of features and multiple ranking schemes. The experiment results based on a real dataset show the effectiveness of our method with significant improvement over baseline.

Keywords Topic detection and tracking · Social media stream · K-partite graph · Multi-dimensional feature

1 Introduction

Nowadays, *microblogging* has become popular, with hundreds of millions of short messages being posted and shared every minute on a variety of topics on social media networks, such as Facebook, Twitter, and Weibo. Many of these messages contain videos that capture particular events or moments in users' lives [1]. We call these videos, which are contained in microblog messages in the form of video links, *blog-videos*. In general, blog-videos have textural messages and video links to their external content sharing systems, and as a result, YouTube-like video sharing sites, such as Youku, has become immensely popular. Social media sites usually allow users to view, forward, and comment on these blog-videos directly, which results in the ability to capture rich information on user activity on the videos, such as comments and forwarding times, among others.

Topic detection and tracking (TDT) aims to detect unknown topics or track known topics from multimedia data stream. The task of TDT has been studied for decades, and many effective TDT methods have been developed to deal with various media forms including articles, images and videos [3, 5, 15]. However, these methods may not be applicable to videos in social media streams, which are generally generated by non-professional users. TDT for large scale social media (web video and its related texts) has attracted considerable research attention in multimedia retrieval [2, 27, 17]. In contrast to general text streams or web videos, the task of TDT for blog-videos is still very challenging due to the following issues.

First, blog-videos often come from heterogeneous sources, among which latent relationships may exist. Information about blog-videos can be collected

from both microblogging sites and their sharing sites. From the first source, messages from blogs and their forwarding can not only have a set of words, but may also consist of information in multiple dimensions; for example, messages, forwarding and forwarding times, and comments. For the second source, we can extract either textural data (such as video titles and their descriptions) or visual features (for instance, key frames contained in video data). However, most of the existing TDT methods either rely only on textual information or visual content while ignoring the relationships among heterogeneous social media data.

Second, the messages that contain blog-videos are highly sparse and noisy. Due to message-length limits of 140 characters (Chinese characters in Weibo), retrieving effective textural features from microblogging sites and textural annotations of videos is difficult, as the texts are often short, ungrammatical, and unstructured, as well as noisy.

Third, blog-videos in social media streams are often of lower quality and exhibit topic drifting. Due to the popularity of microblogging platforms, users discuss the topics in which they are interested, report the events around them, or forward related videos. On one hand, the increasing social activities of people have produced a large volume of social data; but on the other hand, videos are often made and shared by non-professional users, and thus the quality of the videos are not consistent. Moreover, a video may contain various fragments that refer to different topics, often denoted as video topic drifting.

To tackle these issues, we propose a K-partite graph-based Microblog video Topic Discovering and Tracking approach (mvTDT). First, we extract a variety of features from different sources of blog-videos, including text features from messages of blogs (or forwards), text annotations of videos posted by users, and visual features from videos. Then we model the relationships among multiple features using a K-partite graph (KPG). Next, through adding latent nodes to the KPG, we construct an optimal *relation cluster network* (RCN) with minimal information entropy loss between the two graphs. Meta-nodes are created by grouping blog-videos and various types of features into video clusters (namely, topics) and feature clusters, respectively. Using the RCN, we track and determine whether a new blog-video belongs to an existing video topic or a new topic, using our novel multi-dimensional feature fusion algorithm. Finally, we summarize video topics and rank videos contained in topics using three ranking schemas. We evaluate our mvTDT approach on a real-world dataset. Experimental results show the effectiveness of our approach, with a significant improvement over the baseline. Our key contributions include:

1. We propose a new method to simultaneously model the relationships among videos contained in Weibo streams that includes both their textual and visual features.
2. We design a novel joint clustering algorithm to capture the global structure of a K-partite graph by constructing an RCN, where latent nodes are added to the network to represent video clusters and feature clusters.

3. We propose methods for tracking and summarizing the videos in Weibo streams through fusing multiple features and multiple ranking schemes.

The rest of this paper is organized as follows. In Section 2, we summarize related work. We present the structure of our approach for topic discovering and tracking for videos in Weibo streams in Section 3. Section 4 describes our strategies for dataset preparation, experimental design, experimental analysis, and video topics ranking. Section 5 concludes our work and discusses future work.

2 Related Work

The topic detection and tracking task for videos in Weibo streams extends the standard TDT tasks that have been intensively studied in dealing with web videos and texts, as well as videos at the same time. We discuss most related TDT work that has been applied to both general videos and blog-videos.

2.1 TDT methods for social media stream and their visualization

Existing methods of topic (or event) detection in tweet streams use labeled data or external corpus. Li et al. [3] proposed the Twevent system, which is a segment-based event detection system for tweets. It first detects burst tweet segmentations as event segments and then clusters the event segments into events by considering both their frequency distribution and content similarity. They use Wikipedia to identify realistic events and to derive the most newsworthy segments to describe the identified events. Li et al. [4] designed a topic tracker by learning from historical data, including labeled data and plenty of unlabeled data, using a semi-supervised multi-class multi-feature method.

Other work focused on the discovery of hot topics on social media networks. Ran et al. [5] proposed a method of hot topic discovery found in tweets based on the speed of their growth. Zhu and Yu [6] tried to discover potential hot topics before they boost and break out. They employed topic life cycle, hot velocity, and hot acceleration measures to calculate the changes in topic hotness. Tu et al. [7] designed a clustering approach to detect hot topics in Weibo. They first used Bayes classification to filter meaningless tweets, and then detected hot events from the valuable tweets by a mean calculation-based incomplete clustering algorithm.

More recently, some work has considered visualizing the social aspects of Twitter. Zhou and Chen [8] represent a social message as a probability distribution over a set of topics by inference, and the similarity between two messages was measured by the distance between their distributions. Then, events were identified by conducting similarity joins over social media streams. Favre et al. [26] designed MABED, which accounted for the social aspect of tweets by leveraging the creation frequency of mentions that users insert in tweets to engage discussion. Their study designed three visualizations for the detected

events: time-oriented visualization, impact-oriented visualization, and topic-oriented visualization.

In this paper, we focus on the TDT task of videos in Weibo streams. We collect various types of texts for videos from three aspects: messages that contain video links, forwards of target videos, and related descriptions of videos in their sharing websites. These descriptions include video titles, content descriptions, keywords, and tags uploaded by users.

2.2 TDTs on web videos

Tracking video topics using existing text and limited visual cues is becoming a new challenge, as the textual annotations of web videos tend to be sparse and noisy.

One pioneering work employed the bipartite graph method [9]. The method used two steps, coarse topic filtering and fine topic reranking, to discover topics. However, they did not explore multiple features. Shao et al. [10] were the first to introduce a star-structured K-partite graph for web video topic discovering. Their further work [11] proposed a star-structured K-partite graph (SKG) to first represent rich multi-modal features, then introduced a co-clustering process to discover web video topics. They represented web videos and their multi-model features (e.g., keyword, near-duplicate keyframe, near-duplicate aural frame, and others) as an SKG. They viewed an SKG as a specific case of K-partite graphs, where there is a central node set that connects the other node sets to form a starting structure from the relationships between web videos. Cao et al. [12] proposed extracting the global trajectory features to overcome the noisy problem of conventional discrete features, and discovered hot topics by selecting the optimal path in a global topic evolution graph.

Other works have used the social characteristics of web videos [13, 28, 25]. Zhang et al. [14] designed a community-driven web video topic discovery model to model the loose correspondence relationship between content and social networks by constructing a video social network, which is based on users' interactions with videos. They further proposed a community-driven web video topic discovery model.

However, all of the above methods have the following issues: (1) Those methods strongly rely on video tags, which are noisy or may be unavailable in some situations. (2) They pay the most attention to upload times, which can reduce their accuracy. And (3) those methods focus on detecting video topics, while they pay less attention to video topic tracking.

In summary, most of the above mentioned methods are not suitable to sufficiently utilize the rich cross-media information that can effectively improve the performance of topic detection and tracking. The approach in [11] is most similar to our method. However, the differences between two methods are two-fold: (1) we don't emphasize a central node set in the K-partite graph; and (2) we don't use the attribute set defined in an SKG. We compute the distance

between two graphs based on information entropy, and we further propose an joint clustering algorithm to detect video topics.

2.3 Fusing framework of TDTs

More recent work has proposed to model the relationships among a variety of media through fusing frameworks.

Some work has focused on fusing web videos and news articles. Zhang and Li et al. [14] proposed a flexible data fusion framework to detect topics that simultaneously exist in different media. They employed a multi-modality graph that fuses a text graph and a visual graph. However, the two graph were constructed separately. Zhang and Chen et al. [15] proposed a method of utilizing and fusing rich media information from both web videos and news reports. Their weighted keyword groups were used to detect topics from multiple types of media.

Other research has explored microblogging systems and web video. Mohanta et al. [16] propose an event detection method that generates an intermediate semantic entity–microblog clique (MC) to explore highly correlated information among the noisy and short microblogs. The heterogeneous social media data was formulated as a hypergraph, and the highly correlated microblogs are grouped to generate MCs. Based on these MCs, a bipartite graph is constructed and partitioned to detect social events. Wang et al. [1] explored the connections between information propagation in a microblogging system, as well as the number and distribution of actual views in a video-sharing site from two large-scale real world microblogging and video sharing systems.

A literature survey shows that exiting work pays less on video topic visualization, except for two works [18,26] that investigate few guidelines on how to build successful visual analysis tools that can handle specific event types and diverse textual data sources.

In summary, the relationship among various types of media has yet to be fully mined and few work has focused on topic summarization and ranking. In this paper, we focus on detecting topics for videos in a Weibo stream. We first propose to model videos, the textural features and visual features of videos, using a KPG. And we further construct a RCN to discover video topics. We then fuse all the extracted features and other statistical features of blog messages (or forwarded messages) related to videos. We further visualize video topics and rank the videos with same topics by three ranking schemes.

3 The Framework of Our Approach

Given a set of blog-videos ϖ, we propose the mvTDT approach to categorize the blog-videos into variety clusters (topics).

Figure 1 shows the architecture of the mvTDT approach. The four key steps of the proposed approach are: (1) Extracting textural features as well as visual

Table 1 Notations used in the paper

ϖ	Set of blog-videos (videos).
$V = \{v_1, v_2, \cdots v_{dv}\}$	v_i is a video.
$G_K = \{V, \{S^k\}_{k=1}^{K-1}, E\}$.	A K-partite graph (KPG).
$S^k = \{s_1^k, s_2^k, \cdots, s_{d_k}^k\}$	The k-th feature set, and d_k is the size of the feature set.
$E = \{e\left(v_i, s_j^k\right)\}$	Set of edge of v_i and feature s_j^k.
$M^k \in R^{d_v \times d_k}$	M_{ij}^k is the weight of $e\left(v_i, s_j^k\right)$.
$\widehat{G} = \{V, C^v, \{S^k\}_{k=1}^{K-1}, \{C^k\}_{k=1}^{K-1}, \widehat{E}\}$	A relation cluster network (RCN).
$C^v = \{C_1^v, C_2^v, \cdots, C_{d_{cv}}^v\}$	Set of video clusters, and d_{cv} is the number of video clusters.
$C_p^v = \{v_i : W_{ip}^v = 1\}$	Set of videos which can be clustered into video cluster c_p^v.
$C^k = \{C_1^k, C_2^k, \cdots, C_{d_{ck}}^k\}$	The k-th feature cluster set with d_{ck} elements.
$C_q^k = \{s_j^k : W_{jq}^k = 1\}$	Set of features clustered into the feature cluster c_k^q.
$\widehat{E} = \{e\left(v_i, c_p^v\right), e\left(c_p^v, c_q^k\right), e\left(c_q^k, s_j^k\right)\}$	Edge set of a RCN.
W^v, W^k and W^{vk}	Three kinds of weight matrices.
$D_G\left(G, \widehat{G}\right)$	Distance of two graphs.
$D_{vc}\left(v_l, c_p^v\right)$	Video-cluster distance between video v_l and video cluster c_v^p.
α_k	The k-th feature weight.
$\Delta I\left(cluster\right)$	Information loss.

features for videos from various sources. (2) Modeling the extracted features using a K-partite graph. (3) Jointly clustering videos and all extracted features into video clusters (topics) and feature clusters, respectively. Finally, (4) tracking video topics using the proposed multiply feature fusing algorithm. Videos with the same topics are further ranked by three ranking schemes. These steps will be detailed and discussed in the following sections. The symbols are summarized in Table 1.

3.1 Extracting Multiply Features of Videos

We describe our strategies for extracting the textual features and visual features for videos in social media streams.

3.1.1 Extracting textural feature

We have three basic steps for extracting textural features: collecting available texts, extracting keywords, and computing weights of keywords.

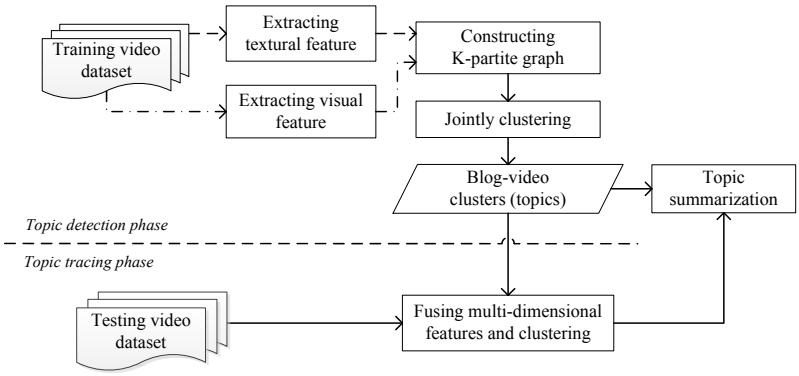

Fig. 1 The framework of our approach.

Collecting available texts. For videos, there are two kinds of textural descriptions: one is from microblogging sites, while the other is from video sharing sites. On one hand, users utilize blog messages and forwarding messages to express their comments on the target videos. On the other hand, there are some other descriptive texts obtained by the corresponding video sharing sites. For instance, Youku.com asks users to add some descriptive texts for their uploaded videos, such as video names, content descriptions, and tags. These texts can be viewed as one source to discover the topics of videos. Our strategy of collecting available texts includes: given a video, we first collect its related blog messages, forwarded messages, and comments. Then we locate its original sharing video site and get its title, keywords, content description, and tags uploaded by users.

Extracting keywords. We then preprocess the collected texts. The preprocessing steps include segmenting the collected texts and removing stopwords. After the step, we extract keywords for each video. However, there are noises in the keyword set for various reasons; for instance, users may upload some irrelevant tags to get the aim of improving clicking rate. If we view all keywords with the same weight, these noises further lead wrong topics or topic drifting. Therefore, we assign weights to keywords to remove noisy keywords.

Weighting keywords. We extend the standard Term Frequence/Inverse Document Frequency (TF-IDF) to weight keywords in social media streams. We compute the *weight* w of keyword t using:

$$w_t = TF_t \times \log\left(\frac{P}{DF_t}\right). \tag{1}$$

Here, P is the number of video clusters. TF_t is the number of keyword t contained in a video cluster, and DF_t refers to the number of video clusters that contain keyword t. In our work, we set TF_t to the maximum frequency of keyword t contained in cluster C_i^v, there is,

$$TF_t = \arg\max\{f\,(t, C_i^v)\},$$

and we have,

$$f(t, C_i^v) = \frac{\sum_{v \in C_i^v} A\,(v, t)}{|C_i^v|}.$$

If keyword t is belonged to video v, let $A\,(v, t) = 1$, else $A\,(v, t) = 0$. This weighting scheme means that the keywords contained in a particular video cluster are important, while the keywords with higher frequencies across many video clusters have less discriminative power. To remove noisy keywords, we introduce the *noise threshold* η. If $w_t < \eta$, keyword t will be removed. All remaining keywords are used in the following steps.

3.1.2 Extracting and clustering visual features

To extract visual features from videos, we perform video shot segmentation and extract key frames. Originating from [16,20,29], we employ a robust approach toward video shot segmentation. Our approach includes the following steps. First, we detect the shot boundary, and we divide a video into discrete video shots. These shots contain a series of consecutive video frames and similar visual features. We then extract key frames from these shots and use the extracted key frames to represent these shots. The key frames satisfy two rules: (1) the distance between two adjacent key frames are relatively large, and (2) the number of key frames are stable, with only small changes.

Next, we cluster key frames from video data. We first cluster video frames contained in a video shot, according to their grey values, by using K-means. We also iteratively update the centroids of clusters to get Z stable clusters. That is to say, we get Z key frames from each video shot. We then use 16-dimensional grey values of the clustered centroids to represent the visual features of key frames.

To define an appropriate value for the number of cluster Z, we adopt the Bayesian information criterion (BIC), which is defined in [19] as

$$BIC = \lambda L - \frac{1}{2}|\theta| \log T,$$

where L refers to the posterior likelihood after fitting to the data. $|\theta|$ denotes the complexity of the model, and T is the total number of samples, respectively.

To cluster visual features, we extend the general BIC computation as:

$$BIC = \log \Delta I\,(cluster) - \frac{1}{2} d_H \times \log d_s, \tag{2}$$

here, d_H refers to the complex of the model and is represented by the vector dimension. Since we represent video shot S_i using 16-dimensional grey values, then $d_H = 16$. d_s refers to the number of video shots. $\Delta I\,(cluster)$ refers to the information loss that occurs after the clustering procedure, and is computed as

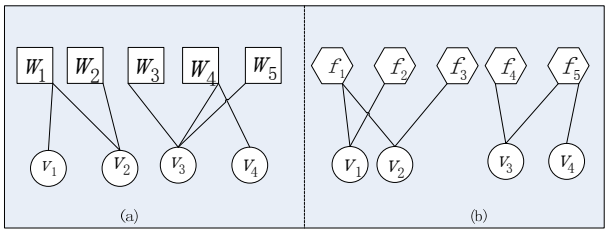

Fig. 2 Bipartite graph of a video and its two feature sets.

the sum of distance among all key frames of a cluster and the cluster centroid. We have

$$\Delta I\left(cluster\right) = \sum_{S_i \in C_{sc}, 1 \leq SC \leq |c_{sc}|} D\left(\overline{C}_{sc} - S_i\right).$$

Here C_{sc} represents the set of key frame clusters and $\overline{C}_{sc}$ refers to the centroid of the clusters. We summarize the steps of determining the value of Z as follows: we first let $Z \in \{1, d_s\}$. During each cluster procedure, we compute the corresponding BIC. We repeat the procedure until we get a minimized BIC. We view the corresponding value of Z as the number of key frame clusters.

3.2 Modeling Videos using K-partite Graph

3.2.1 Creating a K-partite graph

Given an undirected graph $G = (V, E)$, V is the set of nodes and E is the set of edges. If nodes in V can be divided into two disjointed node subsets A and B, and each edge (i, j) and two nodes i and j belong to the two node subsets $(i \in A, j \in B)$, then we call graph G as a bipartite graph [9].

For a given blog-video, we first construct two bipartite graphs for its textural features and visual features, respectively. An illustrative example is shown in Figure 2. We refer $\{v_1, v_2, v_3, v_4\}$ as the video set, $\{w_1, w_2, w_3, w_4, w_5\}$ as the keyword set, and $\{f_1, f_2, f_3, f_4, f_5\}$ as the visual feature set, respectively. Both the keyword set and the visual feature set are considered to be the feature sets of the video.

Figure 2 shows that, for a video, we simultaneously pay attention to the video and its textural features, as well as its visual features. However, a bipartite graph cannot accurately model the latent relationships among three kinds of data; thus, we use a K-partite graph instead of a bipartite graph to model blog-videos.

A K-partite graph is a kind of graph where the vertices can be partitioned into K disjoint sets and edges that exit between any two disjoint sets [21]. An

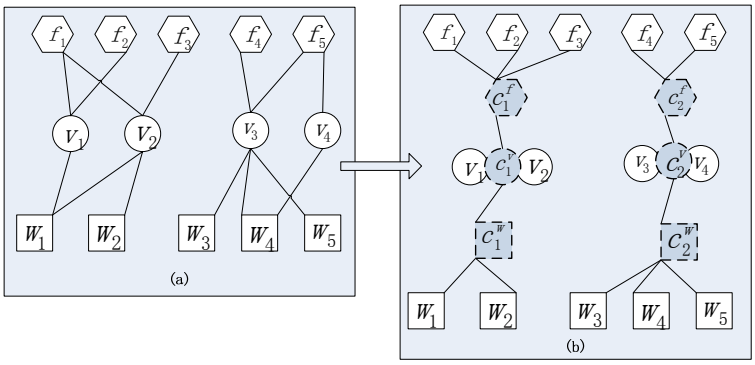

Fig. 3 (a) A K-partite graph, and (b) its revision with latent nodes (small dotted circles or rectangles).

illustrative example of a tri-partite graph is shown in Figure 3(a). We observe that there are three kinds of node sets: the video set $\{v_1, v_2, v_3, v_4\}$, the visual feature set $\{f_1, f_2, f_3, f_4, f_5\}$ and the textural feature set $\{w_1, w_2, w_3, w_4, w_5\}$. All elements in the visual feature set and the textural feature set are viewed as *feature nodes*. There are separate edges from video nodes to textural feature nodes and visual feature nodes, and we use the weights of edges as the weights among the relevant feature nodes and video nodes.

Formally, we denote a K-partite graph G for videos as

$$G = \{V, \{S^k\}_{k=1}^{K-1}, E\}.$$

Here, V refers to the video set. There is $V = \{v_1, v_2, \cdots, v_{d_v}\}$, and d_v is the number of videos. According to the definition of KPG, each video v_i contains K-1 kinds of features. $S^k = \{s_1^k, s_2^k, \cdots, s_{d_k}^k\}$ denotes the k-th feature set with element s_j^k, and d_k is the size of the k-th feature set. E represents the edge set of KPG. $e\left(v_i, s_j^k\right)$ refers to the edge between the i-th video v_i and the j-th feature s_j^k contained in the k-th feature set. We represent the weights of the edges using a co-concurrent matrix $M^k \in R^{d_v \times d_k}$. The element M_{ij}^k in the matrix refers to the weight of the edge $e\left(v_i, s_j^k\right)$.

Through creating a KPG for blog-videos, we build structural relationships between videos and their variety of features. Our basic idea is to obtain the topics of videos through cluster their features. However, as shown in Figure 3(a), it is hard to directly cluster features in a KPG. To cluster nodes in the KPG, it is important to first obtain its global structure.

3.2.2 Construct an optimal RCN

To capture the global structural information of a KPG, originate from [11], we add latent nodes to the KPG. This revision of the KPG is viewed as a *relation cluster network* (RCN). Formally, given a KPG G, it RCN $\widehat{G}$ is

$$\widehat{G} = \{V, C^v, \{S^k\}_{k=1}^{K-1}, \{C^k\}_{k=1}^{K-1}, \widehat{E}\},$$

where $C^v = \{C_1^v, C_2^v, \cdots, C_{d_{cv}}^v\}$ refers to video clusters and d_{cv} denotes the number of video clusters. We denote $\{C^k\}_{k=1}^{K-1}$ as the $K-1$ kinds of feature clusters. $C^k = \{C_1^k, C_2^k, \cdots, C_{d_{ck}}^k\}$ means that to the k-th feature cluster, d_{ck} is the number of feature clusters for the k-th feature. $\widehat{E} = \{e\left(v_i, c_p^v\right), e\left(c_p^v, c_q^k\right), e\left(c_q^k, s_j^k\right)\}$ refers to the edge set of the RCN.

In Figure 3, we add some latent nodes, $c_1^f, c_2^f, c_1^w, c_2^w, c_1^v, c_2^v$, to the KPG. We show the revision in the right of Figure 3. Our observation from Figure 3(b) shows that four videos are clustered into two clusters $\{v_1, v_2\} \in c_1^v$ and $\{v_3, v_4\} \in c_2^v$ respectively. The textural features and visual features are clustered into four clusters: $c_1^w = \{w_1, w_2\}$, $c_2^w = \{w_3, w_4, w_5\}$, $c_1^f = \{f_1, f_2, f_3\}$ and $c_2^f = \{f_4, f_5\}$, respectively.

We use three *weight matrices*, W^v, W^k and W^{vk}, to represent three kinds of edge weights, respectively. $W^v \in \{0,1\}^{d_v \times d_{cv}}$ refers to the weights between video nodes and video clusters. For the edge $e\left(v_i, c_p^v\right)$, if video node v_i is clustered into the video cluster c_p^v, $W_{ip}^v = 1$, otherwise $W_{ip}^v = 0$. Similarly, $W^k \in \{0,1\}^{d_k \times d_{ck}}$ refers to the weights between features and feature clusters. For edge $e\left(s_j^k, c_q^k\right)$, if feature s_j^k is clustered to feature cluster c_q^k, we set $W_{jq}^k = 1$; otherwise $W_{jq}^k = 0$. We use $W^{vk} \in R^{d_{cv} \times d_{ck}}$ to represent the weights between video clusters and feature clusters. For instance, W_{pq}^{vk} refers to the weight of edge $e\left(c_p^v, c_q^k\right)$, which links the p-th video cluster c_p^v to the q-th cluster c_q^k for the k-th feature.

It is important to keep the global structure of the RCN similar to that of the KPG while adding latent nodes to the KPG. The more similar the two graphs, the better the RCN. In other words, we are looking for an *optimal RCN* with the most similar global structure to the KPG. According to the definition of information entropy theory, the loss of information entropy between the KPG and the RCN is $I\left(V; S^k\right) - I\left(C^v; C^k\right)$.

The aim of the optimal joint clustering algorithm is to minimize information entropy, both before and after clustering,

$$\arg\min_{C^v, C^k} \{I\left(V; S^k\right) - I\left(C^v; C^k\right)\}.$$

For two graphs, the information entropy computation can be transferred to calculate *graph distance*

$D_g\left(G,\widehat{G}\right)$ between two graphs as follows:

$$I\left(V;S^k\right)-I\left(C^v;C^k\right)=D_G\left(G,\widehat{G}\right). \tag{3}$$

Based on this observation, we transfer the problem of minimizing the loss of information entropy into computing the distance between the KPG and the RCN. We get an optimal RCN through minimizing the graph distance $D_G(G,\widehat{G})$. From the RCN, we obtain further video clusters.

As we discussed previously, the KPG can be divided into K-1 bipartite graphs. Since each feature has a different influence on clustering videos, we use a weighted distance to reflect the influence. For a bipartite graph G_k, which is formed by the k-th features and videos, the graph distance D_G between the bipartite graph and the RCN $\widehat{G}$ is computed as

$$\begin{aligned}D_G\left(G_k,\widehat{G_k}\right)&=\sum_{v_i\in V,s_j^k\in S^k}D_G\left(<e\left(v_i,s_j^k\right)>,<e\left(v_i,c_p^v\right),e\left(c_p^v,c_q^k\right),e\left(c_q^k,s_j^k\right)>\right)\\&=\sum_{1\le k\le K-1}D_G\left(M^k,W^vW^{vk}W^k\right),\end{aligned}$$

which satisfy $W_{ip}^v=1$ and $w_{qj}^k=1$.

In this study, we use multiplied features. For K-1 types of features, we compute their graph distance $D_G(G,\widehat{G})$ of two graphs as

$$D_G\left(G,\widehat{G}\right)=\sum_{1\le k\le K-1}\alpha_k*D_G\left(M^k,W^vW^{vk}W^k\right). \tag{4}$$

Here α_k refers to the *feature weight* α of the k-th feature during the process of clustering videos. We have $\sum_{1\le k\le K-1}\alpha_k=1$. We learn the value of α_k during training process. The values are used in joint clustering to discover video topics.

3.3 Video Topics Detection

According to the theory of information entropy, a RCN will keep consistent to the KPG if there is a smaller change. The smaller the distance, the more possibility there is to obtain an optimal solution. Based on the previous analysis, our strategy is to minimize the graph distance $D_G(G,\widehat{G})$ within limited iteration times, and thus we can get an optimal RCN. Our key steps are: initialization, updating feature clusters, updating video clusters, and computing distances between two graphs, respectively.

Initialization. As we discussed above, in a KPG, each feature is linked to videos, and each video is represented by features. Our basic strategy is that we first initialize clusters using K-means, then we add the corresponding centroids of clusters to the KPG as latent nodes; thus we get an initial RCN.

For edge $e(v_i, c_p^v)$ in the KPG, if video v_i can be clustered into video cluster c_p^v, then we let the weight $W_{ip}^v = 1$; otherwise $W_{ip}^v = 0$. Similarly, for edge $e(s_j^k, c_q^k)$, if feature s_j^k can be clustered into feature cluster c_q^n, we let the weight of edge $W_{jq}^k = 1$, otherwise $W_{jq}^k = 0$. We initialize the weight matrix W between the video cluster and the feature cluster using

$$W_{pq}^{vk} = \frac{1}{|C_p^v| \times |C_q^k|} \sum_{v_i \in C_p^v, j_j^k \in C_q^k} M_{ij}^k. \tag{5}$$

Here, C_p^v is the videos contained in video clusters, that is, $C_p^v = \{v_i : W_{ip}^v = 1\}$ represents all videos that are clustered into the video cluster c_p^v. Similar, c_q^k is the features contained in feature clusters, there is $C_q^k = \{s_j^k : W_{jq}^k = 1\}$ refers to the features that are associated with feature cluster c_q^k. We limit $1 \le p \le d_{cv}$, $1 \le q \le d_{ck}$, $1 \le i \le d_v$ and $1 \le j \le d_k$.

Updating feature clusters. For each feature, we try to cluster it into a feature cluster; for example, we can cluster feature s_j^k into feature cluster c_q^k. Our basic idea is to compute the distance between the KPG and the current RCN. If the distance is smallest, we cluster the feature into the corresponding feature cluster; that is, if $W_{jq}^k = 1$, there is $\arg\min_q D_G(G, \widehat{G}_q)$. If a feature cluster has been changed, the weight matrix W^{vk} between video clusters and feature clusters should be updated using Eq.(5).

Updating video clusters. For each video, we try to cluster it into a video cluster; for instance, for video v_i and video cluster c_p^v, we iteratively compute the distance between the KPG and the RCN. If W_{ip}^v=1, there is $\arg\min_p D_G(G, \widehat{G}_p)$, and we can cluster the video into the video cluster. If a video cluster has changed, the weight matrix W^{vn} between video clusters and feature clusters should be changed, using Eq. (5).

Computing distances between two graphs. After updating feature clusters and video clusters, we compute the distance between the KPG and the RCN. If the information entropy loss, $I(V; S^k)$-$I(C^v; C^k)$=$D_G(G, \widehat{G})$, is smaller than a predefined threshold ϱ, then we stop the iteration. Otherwise, we repeat the step of updating the feature clusters.

We summarize our video topic detection algorithm using joint clustering in Algorithm 1. Using the proposed clustering algorithm, videos in the original KPG are represented by clusters in the RCN. That is to say, all of the relevant videos are clustered together, and all non-relevant videos are scared. Through getting video clusters, we can detect topics for videos.

Algorithm 1: A joint clustering algorithm for video topic detection from Weibo streams

Input: KPG G, feature weight α_k, threshold ϱ;
Output: RCN $\widehat{G}$, video clusters C_p^v, feature clusters C_q^k;

```
begin
    Initialize weight matrix W_pq^vk using Eq.5;
    Initialize feature clusters C_q^k and video clusters C_p^v;
    Repeat
        Updating feature clusters C_q^k;
        Updating video clusters C_p^v;
        Computing D_G(G, Ĝ) using Eq.4;
    Until D_G(G, Ĝ) ≤ ϱ ;
```

3.4 Video Topic Tracking

After we get video clusters (topics) using the proposed joint clustering algorithm, the problem of tracking topics for new videos can be described as: given video topics, we judge whether a new video v_l is related to an exiting topic or be a new topic.

For a video in social media stream, except its textural features and visual features, there are other features, such as its forward time and the number of forward. The later two features play important roles in tracking topics of videos.

Forward time. The forward time of a video is not an actual time, but a relative time. We view the forward time of a video as the difference between its actual time and the cluster time of a video cluster that contains the video. The actual time of the video v_i is denoted as t^{v_i}, and the cluster time of a video cluster is denoted as $\bar{t}$. All times are measured in minutes over a base time. Considering that our experimental data was collected from the first day of 2010, we view that time as the base time.

The cluster time $\overline{t_p^v}$ of video cluster c_p^v is computed as the average time of all contained videos. There is,

$$\overline{t_p^v} = \frac{\sum_{1 \le i \le d_p^v} t^{v_i}}{d_p^v},$$

here d_p^v refers to the number of videos contained in the video cluster c_p^v. The forward time $Time_{vc}$ of video v_l is computed as the time distance between video v_l and video cluster c_p^v:

$$Time_{vc}\left(v_l, c_p^v\right) = \frac{|t_l^v - \overline{t_p^v}|}{y}, \tag{6}$$

where y is the number of minutes for a year.

Forward number. The forward number is recorded as a chronological vector. We count the *forward number* $F_{num}(v_i)$ of video v_i every day, i.e., the number of blog messages of the video for one day. The distance of forward number is computed using a cosine vector measurement. The forward number of a video cluster $F_{num}(c_p^v)$ is computed as the sum of all forward numbers of all videos in the cluster. It is

$$F_{num}(c_p^v) = \sum_{1 \le i \le d_p^v} F_{num}(v_i),$$

where $F_{num}(v_i)$ is a vector that the forward number of video v_i recorded chronologically. We compute the forward number distance F_{num} between a new video and a video cluster through computing their cosine distance, as

$$F_{num}\left(v_l, c_p^v\right) = \frac{F_{num}(v_l) \cdot F_{num}(c_p^v)}{\sqrt{F_{num}(v_l)} * \sqrt{F_{num}(c_p^v)}}. \tag{7}$$

To employ above two new features, we extend the graph distance D_G between the two graphs as follows:

$$\begin{aligned} D_G\left(G, \widehat{G}\right) = & \sum_{1 \le k \le K-1} \alpha_n * D_g\left(M^k, W^v W^{vk} W^k\right) \\ & + \sum_{1 \le p \le d_{cv}} \sum_{1 \le \tau \le d_\tau} \beta_\tau * D_{vc}^\tau\left(v_l, c_p^v\right). \end{aligned} \tag{8}$$

and the video-cluster distance D_{vc} between video v_l and video cluster c_p^v is extended as:

$$D_{vc}\left(v_l, c_p^v\right) = \sum_{1 \le k \le K-1} \alpha_k * D^k\left(v_l, c_p^v\right) + \sum_{1 \le \tau \le d_\tau} \beta_\tau * D_{vf}^\tau\left(v_l, c_p^v\right), \tag{9}$$

here d_τ is the number of other feature categories, and $D_{vf}^\tau\left(v_l, c_p^v\right)$ is the vido-feature distance between video v_l to the τ-th feature category.

We design a video topic tracing algorithm that fusing various types of features in Algorithm 2. In Algorithm 2, if the distance $D_G\left(G, \widehat{G}\right)$ is less than a predefined threshold, then we view the computation is convergence.

4 Experiments and Analysis

In this section, we first describe our dataset and experimental setting; we analyze the experimental results; and we end with our visualization of video topics and the videos clustered in these topics.

Algorithm 2: Multiply features fusing based video topic tracking algorithm

Input: A RCN $\widehat{G}$, new video v_l
Output: The cluster containing video v_l, the updated $\widehat{G}$

begin
  Initialize video cluster c_p^v the video v_l belongs to;
  Initialize feature clusters the video v_l belong to;
  Get the weighting matrix $W^v W^{vk} W^k$;
  Repeat
    Update feature clusters of $\widehat{G}$;
    Update video clusters of $\widehat{G}$;
    Compute $Time_{vc}\left(v_l, c_p^v\right)$ using Eq.(6);
    Compute $F_{num}\left(v_l, c_p^v\right)$ using Eq.(7):
    Compute $D_{vc}\left(v_l, c_p^v\right)$ using Eq.(9);
    Compute $D_G\left(G, \widehat{G}\right)$ between G and $\widehat{G}$ using Eq.(8):
  Until convergence;

4.1 Dataset and Experimental Setting

To address topic detection and tracking from Weibo streams, we conduct experiments on the web video dataset MCG-WEBV [15] and the self-built Microblog video dataset MicroblogV.

MCG-WEBV dataset: the dataset consists of 80,031 web videos posted on YouTube from December 2008 to February 2009. The core dataset contains of 3,282 videos which were categorized into 73 ground-truth topics being manually annotated.

MicroblogV dataset: we collected the dataset ourselves from Weibo.com during November 2010 to January 2013. We first collected 10,387 microblogs that contained both blogs and videos, and we viewed those blogs as an original dataset. The topics in the original dataset related to economic, scientific, social, and cultural topics. Our investigation shows that some topics contain few videos or that many replicate videos in the original dataset; as a result, we removed those videos and their microblogs. We randomly select 2,000 videos from the remained videos, and manually labeled their topics. Several topics were removed, as there were few videos about them. After this processing, our self-built microblog video dataset contains 58 topics and 1,956 videos.

We divided the MicroblogV dataset into three parts: a *Training dataset* to learn the feature weight α, a *first testing dataset* to detect topics, and a *second testing dataset* to track video topics. We randomly selected 978 videos as the training dataset; the first testing dataset contains 869 videos posted before November 2012, and the remaining 109 videos are collected as the second testing dataset. We designed two experiments to evaluate the effectiveness of video topic discovering and topic tracking using our proposed approaches. We

limit runtime to 2s/video on our experimental machine (with 200 MHZ CPU and 4G memories).

4.1.1 Evaluation criteria of topic detection

In the first experiment, we use normalized mutual information (NMI) [22] to measure the effectiveness of our joint clustering algorithm. The better a clustering algorithm, the closer NMI is to 1; otherwise, NMI is close to 0. We use $C = \{c_1, c_1, \cdots, c_M\}$ to represent the clusters generated by our clustering algorithm, and $FC = \{fc_1, fc_2, \cdots, fc_N\}$ means the manually labeled ground truth. If the number of videos is d_v, then we have

$$NMI\left(C, FC\right) = 2\frac{I\left(C, FC\right)}{H\left(C\right) + H\left(FC\right)}. \tag{10}$$

Here, $I\left(C, FC\right)$ is the mutual information between the generated cluster C and the labeled cluster FC. It is computed by

$$\begin{aligned} I\left(C, FC\right) &= \sum_{fc \in FC} \sum_{c \in C} p\left(c, fc\right) \log \frac{p\left(c, fc\right)}{p\left(c\right) p\left(fc\right)} \\ &= \sum_{N} \sum_{M} \frac{|fc_n \cap c_m|}{d_v} \log \frac{d_v \cdot |fc_n \cap c_m|}{|fc_n| \cdot |c_m|}. \end{aligned}$$

$H\left(C\right)$ and $H\left(FC\right)$ are defined as:

$$\begin{aligned} H\left(C\right) &= \sum_{1 \le m \le M} p\left(c_m\right) I\left(c_m\right) = \sum_{1 \le m \le M} p\left(c_m\right) \log \frac{1}{p\left(c_m\right)} \\ &= -\sum_{1 \le m \le M} \frac{|c_m|}{d_v} \log \frac{|c_m|}{d_v}, \end{aligned}$$

and

$$\begin{aligned} H\left(FC\right) &= \sum_{1 \le n \le N} p\left(fc_n\right) I\left(fc_n\right) = \sum_{1 \le n \le N} p\left(fc_n\right) \log \frac{1}{p\left(fc_n\right)} \\ &= -\sum_{1 \le n \le N} \frac{|fc_n|}{d_v} \log \frac{|fc_n|}{d_v}. \end{aligned}$$

4.1.2 Evaluation criteria of topic tracking

The aim of the second experiment is to judge whether or not a video belongs to a certain topic. Inspired by [23], we classify a video contained in a cluster into three categories: related, partly related, and non-related. We invited 10 users to judge the videos in clusters. The users give their scores as: related videos get 1, partly related videos get 0.5, and non-related videos get 0. We

compute the whole *average score* (AS) [9] of a video v_i belonging to a topic c_p^v as

$$AS\left(v_i, c_p^v\right) = \frac{1}{10} \sum_{1 \le k \le 10} Score_b\left(v_i, c_p^v\right), \tag{11}$$

here, $score_b\left(v_i, c_p^v\right)$ is the score given by the b-th user, and we limit $score_b\left(v_i, c_p^v\right)$ in {0,0.5,1}.

The *Precision* of clustering video v_i into cluster c_p^v is computed as:

$$Precision\left(v_i, c_p^v\right) = \frac{\sum_{v_i \in c_p^v} AS\left(v_i, c_p^v\right)}{|C_p^v|}, \tag{12}$$

where C_p^v refers to the video set of the cluster c_p^v.

We employ *Average Precision* (AP) [9] to measure the precision after ranking videos in cluster c_p^v as

$$\begin{aligned} AP\left(c_p^v\right) = & \frac{1}{\sum_{1 \le i \le |C_p^v|} AS\left(v_i, c_p^v\right)} \\ & * \sum_{1 \le i \le |C_p^v|} \left(AS\left(v_i, c_p^v\right) * \frac{\sum_{j<i} AS\left(v_j, c_p^v\right)}{i} \right). \end{aligned} \tag{13}$$

4.2 Experimental Analysis

We first construct an tri-partite graph using textural features, visual features, and videos for two experimental datasets. We learn many times for the feature weight α during our experiments. Our experiments show that we can get the best experimental results when we set $\alpha_1 = 0.65$ for textural features, and $\alpha_2 = 0.35$ for visual features for the two experimental datasets.

4.2.1 Experiments about topic detection

To evaluate the effectiveness of topic detection of the proposed approach, mvTDT, is compared to the commonly used algorithms, the K-means [11] and R-WKG [15]. The work [11] compared their framework to K-means in their experiments, and R-WKG employs visual and textual linking.

Comparison on the MCCG-WEBV dataset. The comparison results to the MCG-WEBV are shown in Figure 4. The average precision of the top-10 topics are calculated for evaluation. From the figure, we can see that the performance of the mvTDT and R-WKG methods are all much better than that of the K-mean. The performance of R-WKG is poorer than that of mvTDT because of the employment of the modified TF-IDF socre in the document set.

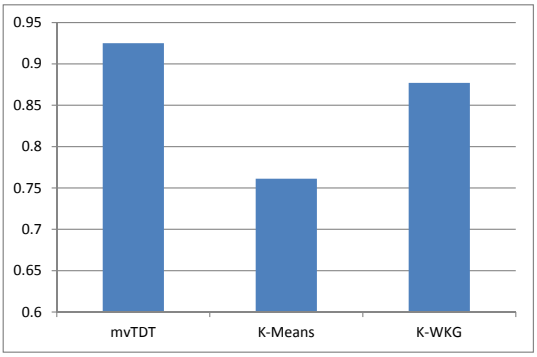

Fig. 4 Comparision on the MCCG-WEBV dataset by NMI%.

Comparison to the MicroblogV dataset. We compared our approach with K-means from four aspects: blogs, blogs+video description, visual features, and multiply features, respectively. The experimental results on the first testing dataset (with 869 videos) are shown in Table 2.

Table 2 Comparison on the MicroblogV dataset by NMI%.

	mvTDT	K-Means	R-WKG
Blogs	75.01%	70.56 %	71.34%
Blogs+video description	77.1%	72.04%	73.42%
Visual features	44.19%	35.21 %	45.03%
Multiply features	80.09%	75.37%	77.12%

Table 2 shows that textural features extracted from both microblogging sites and video-sharing sites perform better than those extracted from individual sources. The extracted visual features perform lowest, as compared with textural features. Our multiply features work best with a score greater than 80%. Compared to K-means and R-WKG, our method performs well in four kinds of features related experiments, except the experimental result of R-WKG on the visual features. Two types of textural features gain better results than that of visual features. The reason is topic drifting caused by segmenting shot and extracting key frame during extracting visual features. The multiply features perform best, which generate the result close to the clusters labeled by users.

4.2.2 Experiments about topic tracking

We conduct experiments about topic tracking on the self-built cross-media dataset MicroblogV. To evaluate the effectiveness of the proposed approach on topic tracking, we compare mvTDT to K-means.

We selected 109 videos as new videos to judge whether or not they are related to a certain topic. Partial experimental results of 10 topics are shown in Table 3. The average precision of the mvTDT approach on topic tracking is 81.2%, which is higher than the score 79.3% of the K-means approach. The precision is significantly better than the baseline ($p < 0.05$ assessed by McNemars test). Some topics with obvious characteristics have a better clustering effectiveness: for example, Topic 9 "Basketball match" and Topic 7 "Ancient Egypt culture". The two topics have unique keywords and distinct visual features. However, some videos with abstract content have a lower tracking precision. For instance, Topic 4 is about "Year-end award". Its tracking precision is more than 72%, which is relatively lower than that of the other topics.

Table 3 Experimental results of topic tracking for 10 topics

Topic ID	Precision	Video ID	Important keywords
1	0.786	8,996	梁静茹(Liang Jinru); MTV; 大手(Big star); 小手(Small star)
2	0.797	9,012	快乐(Happy); 大本营(Headquarter); 爆笑(Comedy); 谢娜(Xie Na)
3	0.873	9,228	钓鱼岛(Diaoyu Island); 中国(China); 飞机(Airplane); 日本(Japan)
4	0.729	9,269	年终奖(Year-end awards); 钱(Money); 绩效(Performance); 过年(Chinese New year)
5	0.754	9,317	广东(Gangdong); 毒打(Beat up); 女孩(Girl); 死亡(Death)
6	0.792	9,456	银行卡(Bank card); 保存(Saving); 收集(Collection); 转账(Transfer)
7	0.876	9,688	文明(Civilization); 埃及(Egypt); 四(Four); 古老(Ancient)
8	0.832	9,910	微软(Microsoft); 创意(Creative); 广告(Ad.); 宣传(Propaganda)
9	0.886	10,151	NBA; 篮球(Basketball); 火箭(Rocket); 赛事(Game)
10	0.795	10,369	2012; 末日(Judgment Day); 灾难(Disaster); 玛雅人(Maya)

Furthermore, we show the ranking results of videos that contain two topics in Figure 5. The two topics are Topic 5 "A Guangdong girl was beat up" and Topic 9 "NBA basketball match". After we rank all videos in the two topics, we observe the corresponding AS values of each videos in the two topics. We noticed that: the higher the AS value of a video, the more relevant the video. That means the video with higher AS value is viewed as more related to the topic. The videos are ranked lower if their AS values are lower; namely, they are not close to the topic. In Topic 9, there are 37 videos, while in Topic 5, there are 19 videos. When we compare the two topics, we can find that the videos in Topic 9 are more close to each other, and the similarity is higher than that of Topic 5.

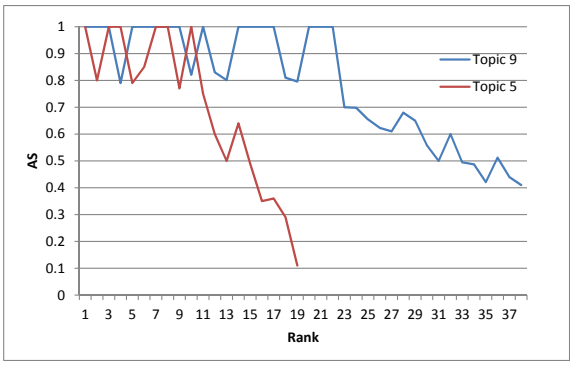

Fig. 5 AS values and the ranking results for two topics.

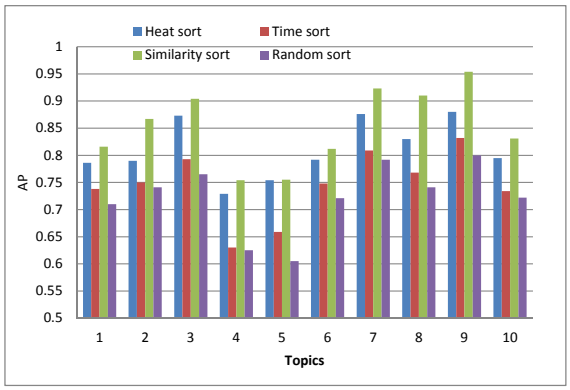

Fig. 6 AP values for microblog video clusters using 4 ranking schemes.

4.3 Ranking Videos with the Same Topic

We can cluster videos into a variety of topics by using the proposed approach. However, videos in each topic are still disordered. If all videos in clusters are ranked according to some rules, on one hand, it is easy to check the effectiveness of our approach of topic discovery and tracking; on the other hand, the ranked videos are able to satisfy user demands. In the following section, we discuss video topics summarization with three kinds of rankings: time-based, similarity-based, and hot-based.

Time-based ranking. This ranking depends on the uploaded chronological order of videos. We define the time-distance D_{time} for video v_i as,

$$D_{time}(v_i) = T_{current} - T(v_i). \tag{14}$$

We refer $T_{current}$ to the current time. The bigger the time distance is, the rear the video is located. That is to say, the newer a video is, the higher

possibility of the video being concerned by users; the older a video is, the less attention these people may pay on.

Similarity-based ranking. Since similarity-based ranking accurately depicts the effectiveness of the topic-tracking method, we compute the video-cluster distance D_{vc} between video v_i and cluster c_p^v, and we further define the similarity sim between videos and their corresponding topics as

$$sim\left(v_i, c_p^v\right) = \frac{1}{D_{vc}\left(v_i, c_p^v\right)}. \tag{15}$$

The smaller the distance, the higher the similarity. The higher the similarity between videos, the closer the video is to the video topic.

Heat-based ranking. Finally, we define the heat of a video as its browsed times. The heat of the topic is defined as the product of the number of videos that belong to the topic and their browsed times [24]. Given a topic c_p^v, its heat is computed as:

$$heat\left(c_p^v\right) = \frac{|C_p^v| * \sum_{v_i \in C_p^v} View\left(v_i\right)}{\max_{v_i \in C_p^v} T\left(v_i\right) - \min_{v_i \in C_p^v} T\left(v_i\right)}, \tag{16}$$

where $View\left(v_i\right)$ refers to the browsed number of the video, and $T\left(v_i\right)$ means the uploaded time of the video.

The videos in a topic cluster are also ranked by their heat as,

$$heat\left(v_i\right) = \frac{View\left(v_i\right)}{T_{current} - T\left(v_i\right)}, \tag{17}$$

Here $T_{current}$ refers to the current time. The closer to the current time, the newer the video is, and the higher its heat.

Figure 7 shows the AP values for ten topics using three ranking methods and a random method. According to the definition of AP, the higher the AP value, the better the ranking method. We notice that the three AP values after three kinds of rankings are higher than that of a random ranking. The AP value of the similarity-based ranking is highest, though the AP value of the heat-based ranking is less than that of the similarity-based ranking, while it is higher than that of the time-based ranking. The time-based ranking gets a lower AP score for the method of ranking videos according to their post times without considering their relevance to a given topic. Heat-based ranking gets a higher AP score for the method that follows the effect of interest on users' attention. Similarity-based ranking gets the highest AP value when considering the content similarity of videos.

Using these three ranking methods, we visualize the video clusters (topics) in Figure 7. There are seven topics, and each topic contains plenty of videos with similar topics. For a given topic, if we click it, we can show videos contained in the topic, and we show these videos ranked using one of three ranking schemes. In Figure 8, we show a sequence of videos that belong to the same

topic, according to their relevance to the topic. For each video, we show its title, keywords, posted time, and the similarity values. The similarity values show the effectiveness of our video topic tracking method. That is, the higher similarity value a video has, the earlier the video is listed in the sequence, which means the video is more close to the topic.

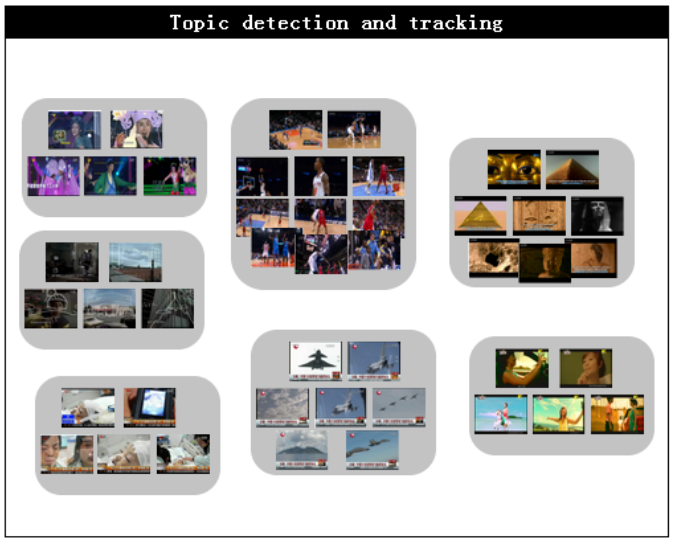

Fig. 7 Visualization of topic detection and tracking.

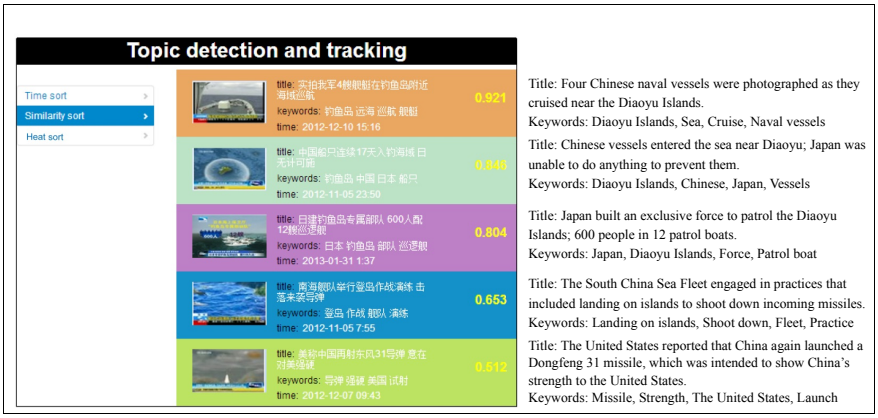

Fig. 8 Videos are ranked by their similarities with same topic and their translations.

5 Conclusion and Future work

In this study, we propose an approach to extract textural and visual features, as well as other features for videos in social media streams on Weibo. We first construct a KPG to represent the relationships between videos and two feature types of videos. Then we design a joint clustering method for the graph: by adding latent nodes to KPG to get RCN, according to the principle of minimal information entropy loss, we try to get an optimal RCN and view its video clusters as video topics. Based on the methods of topic detection, we propose an approach for tracking video topics using multiply features. Finally, we rank and summarize videos in clusters according to their posted time, relevance, and heat. Experimental results on a real dataset show the effectiveness of our approach.

Future work will include: first, using the multiply features fusing strategy discussed in this paper. A weighted factor is employed to adjust the influence of various types of features on topic detection and tracking. Our approach will experimentally learn an optimal value for the feature weight, and we will design an automatic learning algorithm for the feature weight in our future work. Second, we plan to use textural features and visual features of video contained in blog streams during video topic discovering; however, there are other features that may also be useful for topic detection, such as a video's time of upload. In the future, our approach will incorporate more types of features to enhance the overall effectiveness of the proposed method.

References

1. Z. Wang, L. Sun, C. Wu, and S. Yang, Enhancing internetscale video service deployment using microblog-based prediction, IEEE Transactions on Parallel and Distributed Systems, vol. 26, no. 3, pp. 775-785 (2015)
2. J. Cao, Y. Zhang, R. Ji, F. Xie, and Y. Su, Web video topics discovery and structuralization with social network, Neurocomputing (2015)
3. C. Li, A. Sun, and A. Datta, Twevent: segment-based event detection from tweets, in Proceedings of the 21st ACM international conference on Information and knowledge management, pp. 155-164 (2012)
4. G. Li, W. Zhang, J. Pang, Q. Huang, and S. Jiang, Online web-video topic detection and tracking with semisupervised learning, in Advances in Multimedia Information Processing (PCM 2013), pp. 750-759 (2013)
5. L. Ran, X. Suzhi, R. Yuanyuan, and Z. Zhenfang, A modified approach of hot topics found on micro-blog, in Frontier and Future Development of Information Technology in Medicine and Education, pp. 603-614 (2014)
6. T. Zhu and J. Yu, A prerecognition model for hot topic discovery based on microblogging data, The Scientific World Journal, vol. 2014 (2014)
7. H. Tu and J. Ding, An efficient clustering algorithm for microblogging hot topic detection, in 2012 International Conference on Computer Science & Service System (CSSS), pp. 738-741 (2012)
8. X. Zhou and L. Chen, Event detection over twitter social media streams, The VLDB Journal-The International Journal on Very Large Data Bases, vol. 23, no. 3, pp. 381-400 (2014)
9. L. Liu, L. Sun, Y. Rui, Y. Shi, and S. Yang, Web video topic discovery and tracking via bipartite graph reinforcement model, in Proceedings of the 17th international conference on World Wide Web, pp. 1009-1018 (2008)

10. J. Shao, W. Yin, S. Ma, and Y. Zhuang, Topic discovery of web video using star-structured k-partite graph, in Proceedings of the international conference on Multimedia, pp. 915-918 (2010)
11. J. Shao, S. Ma, W. Lu, and Y. Zhuang, A unified framework for web video topic discovery and visualization, Pattern Recognition Letters, vol. 33, no. 4, pp. 410-419 (2012)
12. J. Cao, C.-W. Ngo, Y.-D. Zhang, and J.-T. Li, Tracking web video topics: Discovery, visualization, and monitoring, Circuits and Systems for Video Technology, IEEE Transactions on, vol. 21, no. 12, pp. 1835-1846 (2011)
13. N. Xiong, X. Jia, L. T. Yang, A.V. Vasilakos, Y. Li, Y. Pan, A distributed efficient flow control scheme for multirate multicast networks, IEEE Transactions on Parallel and Distributed Systems, vol. 21, no. 9, pp. 1254-1266 (2010)
14. Y. Zhang, G. Li, L. Chu, S. Wang, W. Zhang, and Q. Huang, Cross-media topic detection: a multi-modality fusion framework, in 2013 IEEE International Conference on Multimedia and Expo (ICME), pp. 1-6 (2013)
15. W. Zhang, T. Chen, G. Li, J. Pang, Q. Huang, and W. Gao, Fusing cross-media for topic detection by dense keyword groups, Neurocomputing, vol. 169, pp. 169-179 (2015)
16. P. P. Mohanta, S. K. Saha, and B. Chanda, A model-based shot boundary detection technique using frame transition parameters, IEEE Transactions on Multimedia, vol. 14, no. 1, pp. 223-233 (2012)
17. Z. Pan, Y. Zhang, and S. Kwong, Efficient motion and disparity estimation optimization for low complexity multiview video coding, IEEE Transactions on Broadcasting, (DOI: 10.1109/TBC.2015.2419824) (2015)
18. F. Wanner, et al., State-of-the-art report of visual analysis for event detection in text data streams, in Computer Graphics Forum, vol. 33, no. 3 (2014)
19. D. Pelleg, A. W. Moore et al., X-means: Extending k-means with efficient estimation of the number of clusters. in ICML, pp. 727-734 (2000)
20. Y. Zheng, B. Jeon, D. Xu Danhua, J. Q.M. Wu and H. Zhang, Image segmentation by generalized hierarchical fuzzy C-means algorithm, Journal of Intelligent and Fuzzy Systems, vol. 28, no. 2, pp. 961-973 (2015)
21. B. Long, X. Wu, Z. M. Zhang, and P. S. Yu, Unsupervised learning on k-partite graphs, in Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining, pp. 317-326 (2006)
22. S. K. Shi and L. Li, A close-to-linear topic detection algorithm using relative entropy based relevance model and inverted indices retrieval, International Journal of Computational Intelligence Systems, vol. 5, no. 4, pp. 735-744 (2012)
23. Y. Zhai and M. Shah, Tracking news stories across different sources, in Proceedings of the 13th annual ACM international conference on Multimedia, pp. 2-10 (2005)
24. J. Cao, C.-W. Ngo, Y. Zhang, D. Zhang, and L. Ma, Trajectory-based visualization of web video topics, in Proceedings of the international conference on Multimedia 2010, pp. 1639-1642 (2010)
25. J. Li, X. Li, B. Yang, X. Sun, Segmentation-based Image Copy-move Forgery Detection Scheme, IEEE Transactions on Information Forensics and Security, vol. 10, no. 3, pp. 507-518 (2015)
26. A. Guille and C. Favre, Event detection, tracking, and visualization in Twitter: a mention-anomaly-based approach[J]. Social Network Analysis and Mining, vol. 5, no.4, pp. 1-18 (2015)
27. N. Xiong, A.V. Vasilakos, L.T. Yang, L. Song, Y. Pan, R. Kannan, Y. Li, Comparative analysis of quality of service and memory usage for adaptive failure detectors in healthcare systems, IEEE Journal on Selected Areas in Communications, vol. 27, no. 4, pp. 495-509 (2009)
28. Z. Xia, X. Wang, X. Sun, Q. Liu, and N. Xiong, Steganalysis of LSB matching using differences between nonadjacent pixels, Multimedia Tools and Applications, vol. 75, no. 4, pp. 1947-1962 (2016)
29. B. Chen, H. Shu, G. Coatrieux, G. Chen, X. Sun, J.-L. Coatrieux, Color image analysis by quaternion-type moments, Journal of Mathematical Imaging and Vision, vol. 51, no. 1, pp. 124-144 (2015)