A Distributed Streaming Framework for Connection Discovery Using Shared Videos

XIAOPENG LI, The Hong Kong University of Science and Technology
MING CHEUNG, The Hong Kong University of Science and Technology
JAMES SHE, The Hong Kong University of Science and Technology

With the advances in mobile devices and the popularity of social networks, users can share multimedia content anytime, anywhere. One of the most important types of emerging content is video, which is commonly shared on platforms such as Instagram and Facebook. User connections, which indicate whether two users are follower/followee or have the same interests, are essential to improve services and information relevant to users for many social media applications. But they are normally hidden due to users' privacy concerns, or are kept confidential by social media sites. Using user-shared content is an alternative way to discover user connections. This paper proposes to use user shared videos for connection discovery with Bag of Feature Tagging (BoFT) method and proposes a distributed streaming computation framework to facilitate the analytics. Exploiting the uniqueness of shared videos, the proposed framework is divided into Streaming processing, Online and Offline Computation. With experiments using a dataset from Twitter, it has been proved that the proposed method using user-shared videos for connection discovery is feasible. And the proposed computation framework significantly accelerates the analytics, reducing the processing time to only 32% for follower/followee recommendation. It has also been proved that comparable performance can be achieved with only partial data for each video and leads to more efficient computation.

CCS Concepts: • **Human-centered computing** → **Social recommendation**; **Social media**; • **Networks** → *Cloud computing*; • **Computing methodologies** → *Vector / streaming algorithms*;

Additional Key Words and Phrases: Social networks, connection discovery, Bag-of-Features Tagging, user shared videos, computation framework, streaming

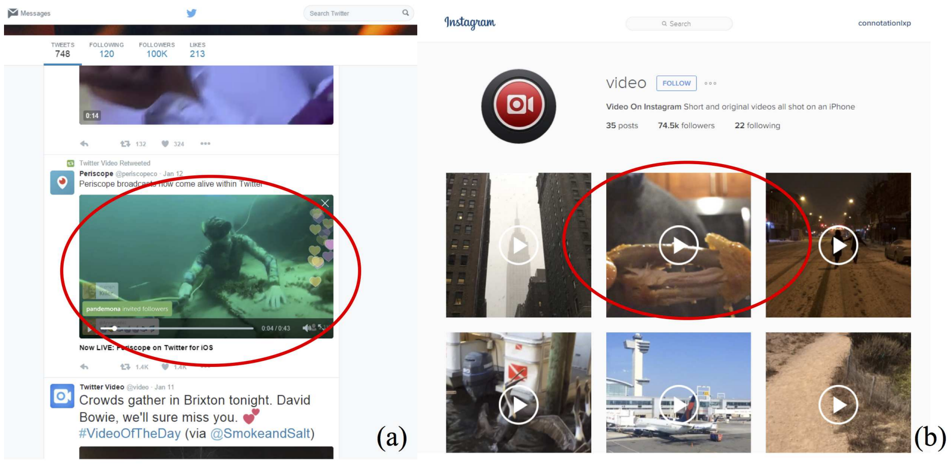
ACM Reference format:
Xiaopeng Li, Ming Cheung, and James She. 2017. A Distributed Streaming Framework for Connection Discovery Using Shared Videos. *ACM Trans. Multimedia Comput. Commun. Appl.* 9, 4, Article 39 (May 2017), 23 pages.
https://doi.org/0000001.0000001

1 INTRODUCTION

Social media has become prevalent among people and revolutionized peopole's interactions in our daily life. Social media platforms, like Facebook and Weibo, help people connect to each other, share content and exchange experiences/comments. With social network analysis, these social media platforms can discover user connections, such as online friendships, follower/followee relationships and community memberships. Many link-based methods for connection discovery has

This work is supported by HKUST-NIE Social Media Lab.
Author's addresses: X. Li, M. Cheung and J. She, Department of Electronic and Computer Engineering, the Hong Kong University of Science and Technology, Hong Kong.

1551-6857/2017/5-ART39 $15.00
https://doi.org/0000001.0000001


(a)
(b)
Instagram
video
FOLLOW
Video On Instagram Short and original videos all shot on an iPhone
35 posts 74.5k followers 22 following

Crowds gather in Brixton tonight. David Bowie, we'll sure miss you. #VideoOfTheDay (via @SmokeandSalt)

Fig. 1. User-shared videos in: (a) Twitter, (b) Instagram

been proposed in the literature using social graphs (SGs), which indicates users' connections and interactions with others [2, 7, 8, 17, 23]. By using those connections, personalized applications such as content recommendation and viral marketing can be developed. However, the limitation of link-based methods is that it suffers from sparsity and cold-start problems. When the SGs are very sparse or users are newly added and not connected to anyone, link-based methods are not applicable. Further, SGs can be hidden by users due to privacy concern, or kept confidential by social media platforms, which make it hard for outside companies to obtain the benefits of social networks. Recently, it has been proved that even without access to SGs, discovering connections is still possible by analyzing user-shared multimedia content [4, 13]. The content-based methods resolves the challenges of link-based methods, and the content has been extended beyond text to include images, audio and video.

In [3], annotating labels using Bag-of-features Tagging (BoFT) on user-shared images for connection discovery has been proved to be a more accessible alternative to social graphs. In [13], Gaussian Relational Topic Model has been proposed to solve the problem of connection discovery with shared images using machine learning. Beyond images, videos are now becoming a more and more popular type of user-shared multimedia contents. Fig. 1 (a) and Fig. 1 (b) show examples of videos shared on Twitter and Instagram. And for some video-based social media sites, such as Youtube and Vimeo, videos are the only sharing medium. Like user-shared images, utilizing user-shared videos could provide more insight into users' interests and possible connections. However, there is a significant difference between shared images and shared videos. Videos contain much richer information than images. Multi-modal information, i.e. visual, audio and textual, is embeded into videos and the visual information can be regarded as a sequence of images. Understanding and extracting the content of videos is much more challenging than doing so for images. Therefore, it is important to come up with an analytic method to mine users' interests and perform connection discovery from user-shared videos. Further, new challenges come into being since videos are generally much larger than images and more complex to process. Intensive computation is required, and the long processing time limits the practicability of the video-based connection discovery. Therefore, both the video analytics for connection discovery and the acceleration for the analytics are important in order for such systems to be practical.

To resolve above challanges, this paper presents a fast and scalable method for connection discovery using user-shared videos, and proposes a distributed streaming computation framework for the analytics. First, in order to extract visual information for connection discovery, user-shared videos are regarded as a sequence of keyframes and visual content is represented using BoFT approach. Users' characteristics are described using the videos they share and the connections between users are discovered afterwards. With the discovered connections, many applications such as collaborative recommendation, user-targeted advertising and user search can be possible. To the best of our knowledge, this is the first work that explicitly proposes to use user-shared videos for connection discovery. As processing user-shared videos is computationally intensive and poor on scalability, in order to serve the proposed connection discovery framework using user-shared videos, a computation framework is proposed. The proposed framework is divided into Streaming Process, Online Computation and Offline Computation. In Streaming Process, the relatively long uploading time of videos is exploited by dividing videos into multiple chunks, and transmitting and processing the chunks with different streaming servers to achieve fast response. Furthermore, the framework manages the computationally intensive parts in Offline Computation, while keeping the users' characteristics and connections incrementally updated in Online Computation to ensure real-time processing. We also find that videos have unique characteristics that may be exploited for faster processing, such as redundancy among sequences. With delicate design, even less computation is required. In summary, this paper makes the following contributions:

- Proposes a BoFT method for connection discovery with user-shared videos and proves its feasiblility through experiments.
- Proposes an efficient distributed streaming computation framework combining Streaming Process, Online Computation and Offline Computation to accelerate the analytics and achieve real-time processing of user-shared videos for connection discovery.
- Conducts several experiments with 35,000+ videos and about 500 users from Twitter, and proves that the proposed framework significantly acclerates the analytics compared with previous frameworks, taking only 30%∼50% as much time as previous works. Experiments also show that a comparable performance can be achieved with only partial data for each video and it further reduces the processing time.

The rest of this paper is organized as follows: Section II reviews the related research work on methods for modeling social media and efficient or even real-time systems in social networks. Section III introduces a BoFT approach for video and indicates the limitations of BoFT. Section IV describes the proposed computation framework for BoFT and how such a general framework is designed and implemented. Section V reports the results of comparative analysis with other frameworks on the Twitter datasets. Section VI discusses potential problems of current framework and gives alternatives that may improve the it. Section VII concludes the paper.

2 RELATED WORKS

As discussed above, it is proven that even without access to SGs, discovering connections is still possible by analyzing user-shared content. Using user interactions and other side information (e.g. location) it is possible to model the strength of connections between users [6, 11, 22]. Some researchers apply a general tie strength model for their own applications [5]. In [18], the authors learn the users' interests from shared content. In [25], the authors use the mean of the feature vectors, along with other features such as comments, to calculate the similarity between users. However, the features applied in these works could be unavailable to other social media platforms, which limits the use of these approaches. In [3], annotating labels using BoFT on user-shared images for connection discovery is proven to be a more accessible alternative to social graphs. Labels are assigned to those images, and the connections among users can be calculated based on the similarity of the occurrence of different labels. Follower/followee recommendation and gender identification are possible using these discovered connections. It has been proved that two users with a high similarity are more likely to be follower/followee than two users with a low similarity[4]. In [13], Gaussian Relational Topic Model has been proposed to solve the problem of connection discovery with shared images using machine learning. Yet when the user-shared content extends to videos, few previous works have attempted to utilize this type of multimedia content information for connection discovery.

The large amount of social media users generate a large amount of multimedia content in social media sites. For better user experience, instant response is generally desired in those systems. Substantial research efforts have focused on real-time computation frameworks in social networks and following are several of them. [19] proposed a distributed framework using Hadoop MapReduce for tag-LDA model training, and the tag recommendation is done in real-time by utilizing the model parameters obtained in the training process. However, since the parameters do not update periodically, the framework does not utilize newly generated data. Therefore the precision could be degraded when applied to social media, where data is generated continuously. [20] proposed a real-time framework for document classification. The framework consists of online computation and offline computation, where the offline computation does the classification using a clustering method and the online computation uses the cluster result to classify a document. [10] proposed a cloud assisted framework for bag-of-feature tagging with user-shared images, in which the computationally intensive part is accelerated by using the cloud. However, in general both documents and images are much smaller than videos, and processing videos generally demands a higher order of magnitude of computation. In real-time applications where instant response is required, it is challenging to design an efficient framework to process videos.

There are some unique characteristics of videos, compared with other forms of multimedia content that could be utilized to speed up their computation. The on-demand video streaming technique is a mature technology in which video fragments are fetched by a client from a server in small units called chunks [12]. [24] proposed a streaming data processing pipeline for content-based video similarity search. Their system parallelizes the feature extraction of a certain video at keyframe level and further aggregates the keyframe features to get the feature representation of videos. However, they did not consider that the uploading and keyframe extraction also costs a lot of time.

This paper investigates the feasibility of using user-shared videos for connection discovery and a framework for Bag-of-Features Tagging approach is designed by combining streaming processing, online and offline computation so as to make connection discovery faster compared to other frameworks on social media. To the best of our knowledge, this is the first work that explicitly proposes to use user-shared videos for connection discovery and the first work that proposes an efficient computation framework using user-shared videos applied to the Bag-of-Features Tagging approach for connection discovery in social networks.

3 CONNECTION DISCOVERY WITH SHARED VIDEOS

This section introduces how to generate video representations for user-shared videos and learn user profiles, as well as how to recommend follower/followee relationships based on the discovered connections. A general framework for connection discovery with user-shared videos is shown in Fig. 2. Three major steps for the framework are included: generating video representations (step 1), learning user profiles (step 2), and discovering connections (step 3). The video representations are generated by treating videos as a sequence of keyframes. Thus, step 1 includes both keyframe extraction from videos and keyframe labeling process using BoFT. With the video representations, user profiles thus can be learned by aggregating the videos shared by each user, as described in

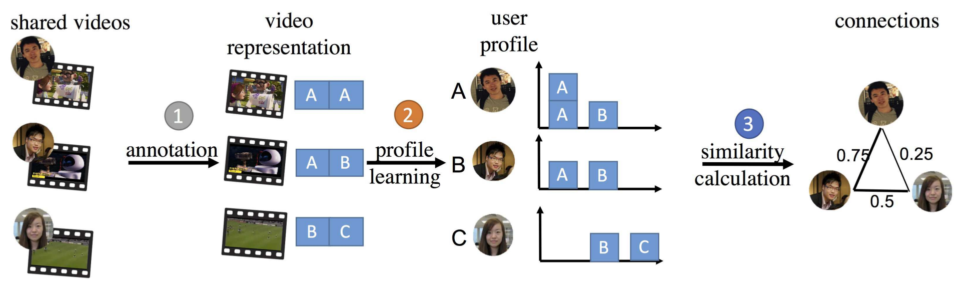


Fig. 2. An overview of the proposed method for connection discovery with user-shared videos

step 2. Finally, the similarity among users can be obtained using user profiles, and connections can be discovered, as in step 3. The technical details for each step are described as follows.

3.1 Keyframe Extraction

A video can be regarded as a sequence of images, and most of them are very similar. Thus they contain a lot of redundant information. Most videos shared by users have about 30 frames/second. Storing and processing all the frames in a video requires large storage capacity and intensive computation resources. Therefore, a common first step for most content-based video analysis techniques is to segment a video into elementary shots and extract keyframes for each shot. The videos can thus be represented by a sequence of keyframes, lowering the demand of both storage and computation.

There are numerous keyframe extraction methods [14, 21], such as the HSV-based approach and clustering-based approach. For simplicity, this paper utilizes the advantages of video coding. Normally, encoded videos have I-frames which are encoded wholly and do not require other video frames for decoding. And the other types of frames, i.e. P-frames and B-frames, are encoded using relative information with respect to the I-frames. Here the paper uses the I-frames as keyframes, thus reducing the complexity and increasing the computation speed.

3.2 Video Representation using BoFT

After the keyframe extraction process, a video can be effectively represented by a sequence of keyframes. In order to extract information from videos and encode videos as vectors for subsequent processing, this paper extracts visual features from keyframes and uses the BoFT approach to generate the codebook for the keyframes. The video representations thus can be obtained by integrating the keyframes of a video using the codebook. The video representation process is described in details as follows.

3.2.1 Feature Extraction. Feature extraction is a process to obtain the local features, as in step 2 of Fig. 3. One of the most common approaches for feature extraction is Scale-Invariant Feature Transform (SIFT) [15], proposed by D. Lowe in 2004. There are mainly four steps, Scale-space Extrema Detection, Keypoint Localization, Orientation Assignment and Keypoint Descriptor, involved in the SIFT algorithm. Features extracted from an image are represented by several vectors, and each vector represents one of the features of the image.

3.2.2 Feature Codebook Generation and Feature Coding. Codebook generation (step 3 of Fig. 3) is a process to obtain the visual words that can represent the features obtained in the feature extraction step. It is a clustering process that groups similar features. This paper adopts a common algorithm, *k*-means clustering [16], to group the feature vectors based on their visual similarity.

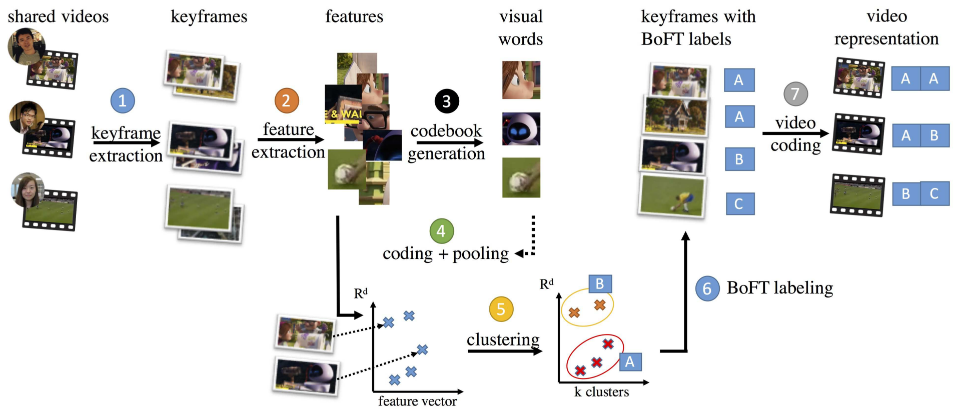


Fig. 3. Video representation using BoFT for user-shared videos (Step 1 in Fig. 2)

Feature coding is the encoding of features with the visual words. Each feature in every image is represented by a visual word in feature coding. The image is then represented by a feature vector in the feature pooling. This process is carried out in step 4 of Fig. 3. A histogram that counts the number of occurrences of each visual word in the image is used to represent an image.

3.2.3 Keyframe Clustering and Video Representation using BoFT. The goal of keyframe clustering is to group keyframes with similar feature vectors. Each cluster obtained in this operation corresponds to a group of similar objects. After obtaining the cluster, as in step 6 of Fig. 3, the images in any clusters are assigned with the same BoFT label to reflect that they are visually similar and belong to the same group. Video representation is achieved using the BoFT approach. That is, each video is represented by a histogram that counts the number of occurrences of each cluster label. The keyframe label distribution of video v is represented by L_v:

$$L_v = \{l_{v,1}, \cdots l_{v,k}, \cdots, l_{v,K}\}. \tag{1}$$

Each element $l_{v,k}$ in L_v is the number of occurrences of label k among all of the keyframes of video v.

3.3 User Profile and Connection Discovery

With encoded user-shared videos, the profile learning, as in step 2 in Fig. 2, is performed for each user. The user profile L_i can be generated by simply aggregating all the videos shared by the user and taking the average of all the video features:

$$L_i = \{l_{i,1}, \cdots l_{i,k}, \cdots, l_{i,K}\} = \sum_{v \in V_i} \frac{L_v}{||L_v||}, \tag{2}$$

where V_i is the set of videos user i has shared. With different content and durations, the number of keyframes can be varied among videos and the video vector L_v is normalized to 1. The user profile obtained is the keyframe label distribution, which reflects the visual features of the user-shared videos.

When the user profile is obtained, the next step is to calculate the similarity between users based on their user profiles. The similarity between two users is calculated by the cosine similarity in

BoFT:

$$S_{i,j} = S(L_i, L_j) = \frac{L_i \cdot L_j}{||L_i|| \cdot ||L_j||}, \tag{3}$$

where L_i and L_j are the distributions of user i and j, respectively. In [4] it is proved that two users who share similar images have a higher $S_{i,j}$. Since a video is a sequence of images, two users who share similar videos will have a sequence of similar images and thus have a higher $S_{i,j}$. Finally, a similarity graph based on label distributions is generated as in Fig. 2, and the most similar J users will be chosen for applications such as follower/followee recommendation.

3.4 Challenges and Limitations of BoFT

As real-world social media platforms require instant response to achieve a real-time experience, it is desired to reduce the computation time from the instant when a user finishes uploading a video to the instant when the recommendations are generated. However, the keyframe extraction and feature extraction take a long time and cannot be reduced. It is a challenge to accelerate the proposed method for video processing to make it practical. Further, as shown in step 5 of Fig. 3, tagging labels for keyframes consumes a lot of computational and storage resources because all the data needs to be stored in memory and many iterations are involved in the clustering process. A stand-alone machine definitely cannot efficiently process millions of videos from millions of users in real-world social networks. Motivated by these observations, the computation framework proposed will be introduced in the next section.

4 PROPOSED DISTRIBUTED COMPUTATIONAL FRAMEWORK FOR VIDEO-BASED TAGGING

This section describes how the original BoFT shown in Fig. 3 can be improved in a much more efficient fashion. Fig. 4 shows the proposed computation framework of connection discovery with user-shared videos on a cloud platform. The framework consists of three parts: (a) Streaming Processing Module; (b) Online Computation; (c) Offline Computation. As in Fig. 4, the streaming framework has the following steps:

- Step 1: When a user shares a video, Streaming Processing starts. User-side Streaming Module segments the video into smaller chunks and uploads the chunks to servers by interacting with the Tracker at the server side.
- Step 2: When one of the chunk servers receives a chunk of video, the server starts to extract keyframes and visual features for the chunk through chunk processing. After each server finishes the chunk processing, it submits the results to a Master server, which is responsible for aggregating information from all previous servers and for further processing.
- Step 3: When the Master server receives new data from chunk servers, the profile of the video is contructed by combining the visual information of the chunks. User profiles and similarity can be updated, and thus the connections can be discovered.

The discovered connections are not only for follower/followee recommendation, but also for other applications such as item recommendations, user-targeted advertising, etc. In the meantime, the Offline Computation periodically interacts with the servers for the chunk processing. The servers of the chunk processing send the keyframes extracted from every chunk to Offline Computation. The offline computation periodically conducts the computationally intensive image clustering and updates the clustering results to all servers for chunk process and the similarity and candidate lists to the Master server for the purpose of maintaining connection discovery performance.

To further elaborate why such a framework (Fig. 4) is designed in this paper, two types of requests from users are defined here: (1) user logs in and automatically receives recommendations from the

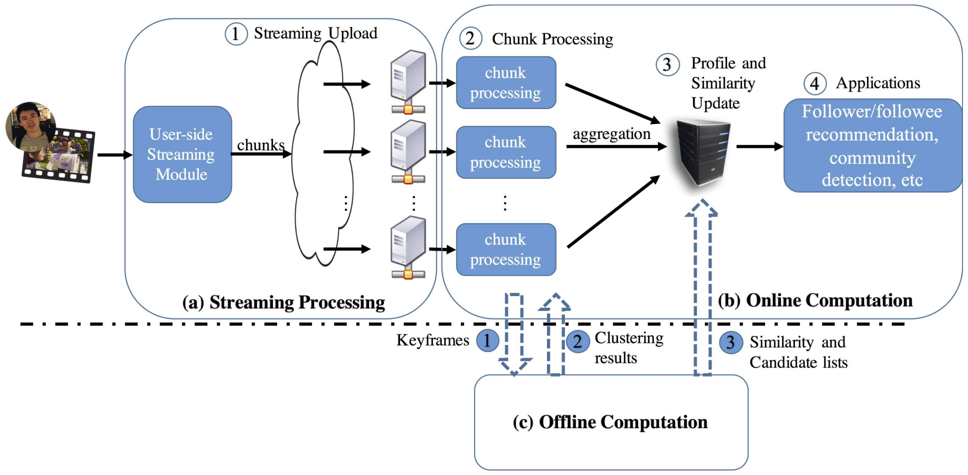


Fig. 4. The proposed computation framework of connection discovery with user-shared videos. (a) Streaming process to upload video chunks. (b) The online computation which contains five steps to efficiently discover connections. (c) The offline computation periodically runs the clustering and generates sorted lists for the online computation.

discovered connections; (2)user shares several videos and the connections are discovered within an instant time interval. In terms of the first type, the framework can simply extract the discovered connections from the list and send them back to the user as the response. For the latter case, the videos have to be uploaded and processed and the user profiles, as well as connections between users, have to be updated. Before the analytic framework of Fig. 1, the first bottleneck for instant response is the relatively long uploading time of the videos, depending on the size of the videos and the network bandwidth. However, instead of waiting for the completion of uploading of the whole-length video, it is desired to take advantage of the long uploading time and start processing while the video is still uploading. The strategy is revealed in the Streaming Processing in the proposed computation framework. The second bottleneck is the step 5 in Fig. 3, where the all the keyframe images are clustered and the BoFT labels are assigned to the keyframes. Clustering of all images is computationally intensive in that the amount of images is huge and generally it takes many iterations in order to converge. This bottleneck limits the speed of the instant response the system can achieve. The bottleneck is coped by dividing the anlytics into Online Computation and Offline Computation, as in the proposed system in Fig. 4.

The details of each part of the framework, Stream Processing, Online Computation and Offline Computation are explained as follows.

4.1 Video Streaming Processing

The durations of user-shared videos ranges from a minute to hours, and the sizes of the videos thus vary from several megabytes to several hundred megabytes. It takes a relatively long time for users to upload a video, for example, with around 1MB/s uploading rate by testing on YouTube, a 100MB video takes 100s just to be uploaded. This framework desires to exploit the uploading time for processing. Yet, as mentioned before, videos tend to have a lot of redundancy and not all the video data is valuable and desirable for connection discovery. Therefore, the above-mentioned unique characteristic gives us the opportunity to speed up the computation and make connection discovery by 1) using streaming processing to upload and process the video chunk by chunk such

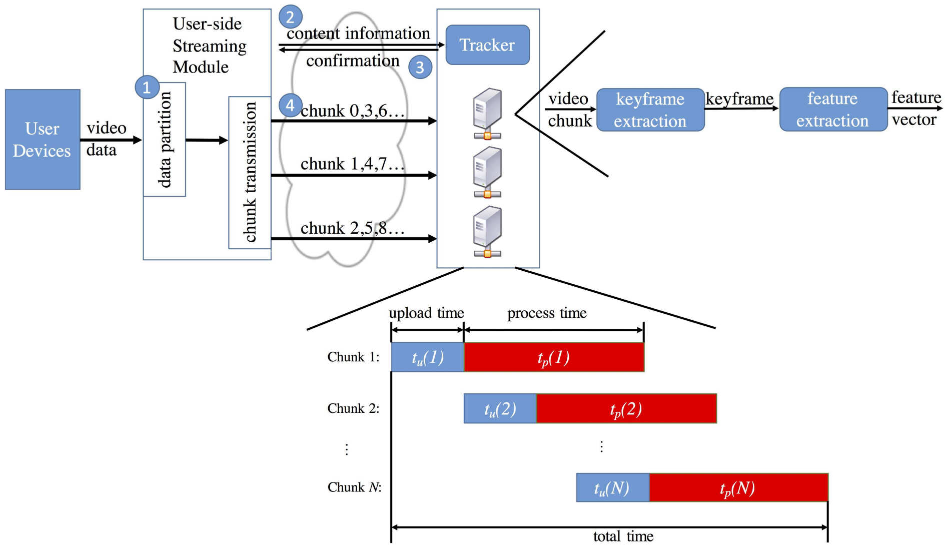


Fig. 5. Video Streaming Upload System (as Step 1 in Fig. 4)

that when one chunk is uploading previous chunks are being processed and 2) making connection discovery even before the completion of video uploading.

4.1.1 Video Streaming Upload. In the last couple of years, video streaming download using TCP/HTTP has become quite popular. And yet, video streaming upload is utilized in our framework. The Video Streaming Upload system is shown in Fig. 5. It mainly consists of Streaming Module in user side, Tracker and Slave Servers in server side. The User-side Streaming Module is responsible for video data segmentation and chunk transmission. A video file is generally composed of a sequence header followed by many Group of Pictures (GOPs), where the decoding of frames can be conducted within individual GOP, as shown in Fig. 6. The User-side Streaming Module segments the videos aligned to the GOP boundaries such that each video chunk can be processed without reliance on other video chunks. Then it interacts with the server to transmit the video chunks. In the server side, Tracker server is responsible for listening to the uploading requests and coordinating the scheduling of all the Slave Servers. In particular, user-side module makes uploading requests through Tracker Server and the Tracker Server redirects the user-side module to the idle Slave Servers that can process the uploading tasks and tracks the status of the tasks till they are finished. The Slave Servers are the ones that perform the transmission of chunks between the user side and server side, and further processing of the chunks.

An example is presented in Fig. 5 to demonstrate the streaming process. When a user shares a video through user device, as in step 1, the video is partitioned into N chunks by User-side Streaming Module and all the chunks are prepared to be uploaded sequentially. The User-side Streaming Module sends a request to the Tracker with the video information, as shown in step 2. The Tracker initializes the uploading task and sends a confirmation and an identifier back, as in step 3. Once the user-side module receives the confirmation, it starts to upload Chunk 1 by requesting an upload with the identifier to the tracker. The Tracker sends the URL of an idle Slave Server and the user-side module uploads Chunk 1 to the corresponding Slave Server, as in step 4. Once the first chunk finishes uploading, the user-side module continues to request next upload, while the

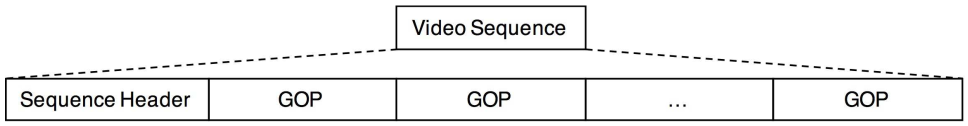


Fig. 6. video file structure

first chunk starts to be processed by the corresponding Slave Server. The timing of the process is also shown in Fig. 5. The uploading of the N chunks happens sequentially while the processing is conducted parallelly. Since the processing takes several times longer than the uploading, the benefit obtained is significant.

The algorithms of the user-side streaming module and the tracker server in the server-side module are shown in Algorithm 1 and Algorithm 2. In Algorithm 1, when the user starts to upload a video, the module receives the video data and the tracker server URL as the arguments. The chunk segmentation module segments the video into chunks according to the header information and requests initialization of the uploading by sending the header of the video to the tracker server. It expects to receive an identifier as a response to identify this uploading session. Then uploading of chunks starts in order by the requesting of the URL of the receiving server from the tracker server. It ends the uploading session by sending an end code to the tracker server. In Algorithm2, when the tracker server receives the initialization request, it starts to coordinate the transmission of chunks and dispatches the URL of the corresponding receiving server back to the user when the user requests to upload chunks. Once the receiving servers receive a complete chunk, the processing module starts the processing in parallel.

4.1.2 *Keyframe Extraction and Feature Extraction for Chunks.*

After each chunk finishes uploading, the server can send the chunk for processing. And to process multiple chunks in parallel, the cloud is employed. In general, the master node will send each chunk to slave nodes and each slave node processes a chunk till it finishes and returns the computation result back to the master for aggregation in the following step. The tasks for the slave nodes are keyframe extraction and feature extraction for each keyframe. In order to do keyframe extraction in the slave nodes, the header information of the video has to be passed to the slave nodes as well. And since the video is partitioned according to the boundaries of GOPs, chunk video decoding is not a problem. The keyframe extraction method is described in the previous section. As for the feature extraction, SIFT features will be extracted for each keyframe of the chunk video and SIFT vector for each keyframe will be obtained according to the SIFT feature codebook.

ALGORITHM 1: User-side Streaming Module

```
Input: The video $V$ shared by user $i$, tracker_server_url
{header,$\{v_1, v_2, \cdots, v_N\}$} ← video_segmentation($V$) ;
Identifier ← initialize_upload(tracker_server_url,header) ;
for $v_i \in \{v_1, v_2, \cdots, v_N\}$ do
    url ← request_upload(tracker_server_url,Identifier) ;
    start_upload(url, $v_i$) ;
end
status ← end_upload(tracker_server_url,Identifier) ;
```

4.1.3 *Video Data Aggregation.*

There are two kinds of video data aggregation that need to be done. One is to aggregate all the video chunks into a complete video, which is the user-shared

ALGORITHM 2: Tracker in Server-side Streaming Module

```
while header ← wait_for_initialize() do
    Identifier ← generate_identifier(header) ;
    respond_initialize(Identifier) ;
    Status ← INITIALIZED ;
    while true do
        (Identifier, Code) ← wait_for_request_or_end() ;
        if Code == END_CODE then
            status ← enquiry_upload_status(Identifier) ;
            break ;
        end
        url ← generate_upload_url(Identifier) ;
        respond_request(url) ;
    end
    respond_end(status);
end
```

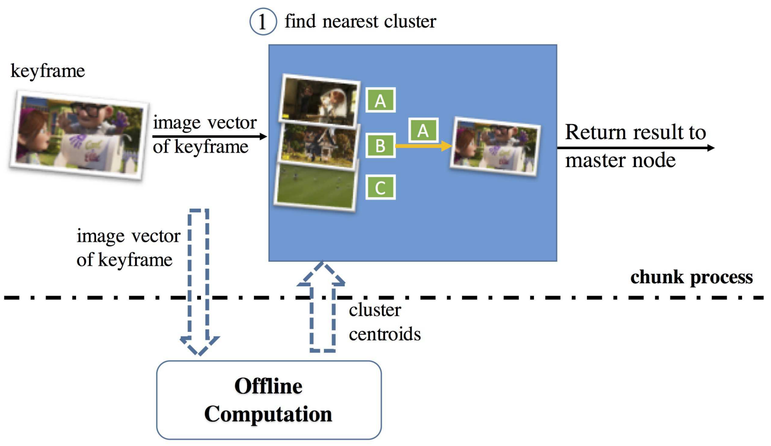


Fig. 7. Chunk process (as Step 2 in Fig. 4)

content on a social media site. This is conducted by concatenating the chunks consecutively and combining the video header information into a video file. The other is to aggregate the processing results of the video. The video feature is the aggregation of all the feature vectors of the keyframes by combining the results from chunk servers in a consecutive way.

4.2 Online Computation

The online computation deals with the process from the instant when keyframes of videos are available to the instant when the recommendation is made. In the framework of this paper, the online computation contains one online server to process users' requests, while it is also flexible in having many online servers to support many user requests. The online computation comprises of four main parts: assigning the BoFT labels for the incoming keyframes, updating the profile of each user, recalculating the similarity and sorting the list of discovered connections based on the similarity.

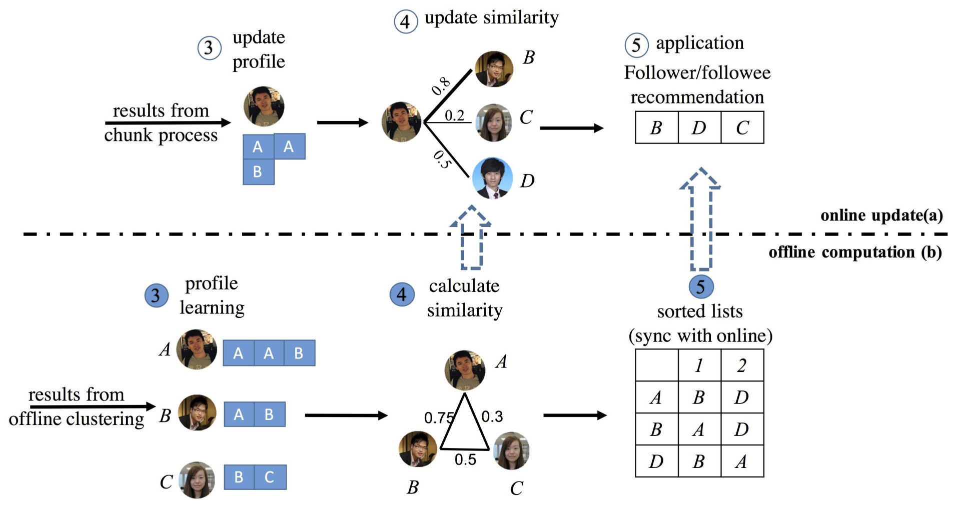


Fig. 8. Update user profile and similarity (as Step 3 in Fig. 4)

4.2.1 Video-based Non-user Generated Annotation. Generating labels for keyframes and video representations for shared videos is conducted chunk by chunk in chunk processing as shown in step 2 in Fig. 4. The details of the chunk processing are shown in Fig. 7. With the feature vectors of all keyframes of a video available, each keyframe can be assigned with a label using BoFT. In general, this should be conducted by reclustering all the keyframes available in the database to maintain a good precision. However, since the number of all keyframes is very large, it is computationally intensive. To alleviate the computation intensity, an efficient algorithm is developed for the online computation, while still maintaining comparable precision.

The offline computation will train a model using previously obtained keyframes and the chunk process reads the clustering result from the offline computation and updates the cluster centroids periodically. With the cluster centroids available, the chunk process compares the input keyframe with all the cluster centroids and assigns the keyframe f with the cluster whose centroid is the nearest to the keyframe with the Euclidean distance metric:

$$label_f = \arg\min_i d(C_i, f), \tag{4}$$

where C_i is the cluster centroid of cluster i.

With the label of each keyframe assigned, the video representation is simply obtained using the histogram of all the clusters the video has, as described in the previous section. And the video vector will be normalized later in order to deal with the problem of different video durations.

4.2.2 Profile Regeneration and Similarity Update. As described in Section 3.3, the profile of user i is defined by a vector L_i and obtained by taking the average of all video vectors that belong to user i. When user i posts a new video, the keyframe images will be assigned a BoFT label and the profile of user i needs to be updated. This step is represented in step 3 in Fig. 4(a).

Users who have corresponding videos will acquire the corresponding BoFT labels, as shown in Fig. 9. Thus the corresponding user vector L_i will be updated by adding video labels with normalization in the following way:

$$L_i = L_i' + \frac{L_v}{||L_v||}, \tag{5}$$

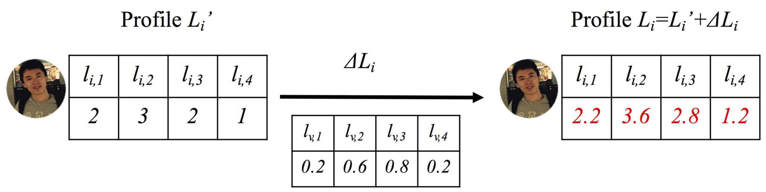

Profile L_i'
$l_{i,1}$ $l_{i,2}$ $l_{i,3}$ $l_{i,4}$
2 3 2 1
ΔL_i
$l_{v,1}$ $l_{v,2}$ $l_{v,3}$ $l_{v,4}$
0.2 0.6 0.8 0.2
Profile L_i=L_i'+ΔL_i
$l_{i,1}$ $l_{i,2}$ $l_{i,3}$ $l_{i,4}$
2.2 3.6 2.8 1.2

Fig. 9. Profile update with an incoming normalized video label

where L_v is the video vector of the newly uploaded video. Since the video is processed chunk by chunk, the user profile can still be updated without having to wait for all the chunks to finish processing. For example, user i, who has user profile L'_i has posted a video and the video is segmented into N chunks, in which M chunks have already been processed. The video vector can be represented simply as:

$$L_v = L_{v,1} + L_{v,2} + \cdots + L_{v,k} + \cdots + L_{v,M}, \tag{6}$$

were $L_{v,k}$ is the results returned by the server processing chunk k. When the user profile is obtained, similarities between L_i and all other users need to be updated as well.

If the similarity is purely recalculated without considering previous information then the complexity will still be maintained as $O(N^{(u)}K)$, where $N^{(u)}$ is the number of users and K is the dimension of the user profile, the distribution of labels. Here an efficient incremental algorithm has been developed. For example, user i, who has user profile L'_i has posted a video v and the assignment of the fth keyframe image is label k. So the number of label k of video v has is $l_{v,k} = l'_{v,k} + 1$. That is

$$L_v = L'_v + \Delta L_v = (l'_{v,1}, l'_{v,2}, \cdots, l'_{v,k} + 1, \cdots, l'_{v,K}). \tag{7}$$

The similarity between user i and user j can be updated in constant time with an incoming keyframe label from Eq. 3 as

$$S_{i,j} = \frac{L'_i \cdot L_j + \frac{L'_v \cdot L_j + l_{j,k}}{||L'_v|| + 2l'_{v,k} + 1}}{\sqrt{||L'_i||^2 + 2 \cdot \frac{L'_i \cdot L'_v + l_{i,k}}{||L'_v||^2 + 2l_{v,k} + 1} + 1} \cdot ||L_j||}, \tag{8}$$

where $L'_i \cdot L_j$, $L'_v \cdot L_j$, $L'_i \cdot L'_v$ and the norm $||L'_i||$, $||L'_v||$, $||L_j||$ are already known. The deduction of the incremental update is shown in detail in the Appendix.

In Eq. 8, the computation can be done in constant time by using the similarity and norm computed previously and updating the dot products and norms as described above. The operation involved is only scalar addition, which is different from recalculating the similarity, where calculating the norm and vector dot product are involved. Thus the complexity for calculating the updated similarity with all other users reduces to only $O(N^{(u)}F)$, where F is the number of keyframes a video has and normally has a level of magnitude smaller than the dimension K of the user profile.

4.2.3 Efficient Follower/Followee Recommendation. To discover connections for a user who has just posted videos, the framework will generate the most similar J users from all $N^{(u)}$ users based on their similarities. Heap search, whose complexity is only $O(N^{(u)} logJ)$ is used to efficiently generate the top-J connections. It is also useful to create a more efficiency by having some elegant pruning for connection discovery. Because the offline computation has already generated sorted lists, users that are impossible to be in the top-$\mathcal{J}$, even with the increment on the similarity, can be located and excluded in the calculation. The framework first calculates the similarity between

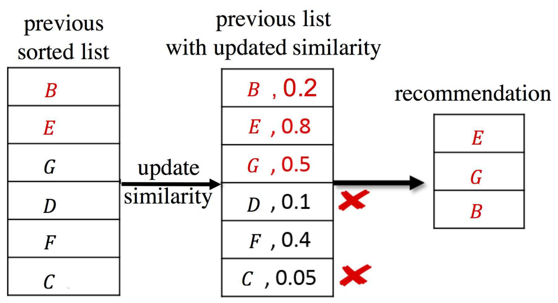


Fig. 10. Pruning in connection discovery for user L_1 with $M = 3$ discovered connections. (L_3 and L_4 are not necessary in sorting)

the user who triggered the request and the J users who are listed as the top-$\mathcal{J}$ connections only. The pruning process happens while calculating the similarity. For example, in the Fig. 10 example, $M = 3$ and the list is for user 1 with profile L_1. While calculating the similarity between L_1 and other users, the framework starts calculating from the previous sorted list and maintains a minimum similarity value, 0.2 in this example, during calculation of the first M similarities. Continuing to calculate others, L_4 in this example, the framework will compare the similarity with the minimum value. If that similarity is even smaller than the minimum value, then there is no need to sort the corresponding user L_4 because its ranking must be over M. To formulate the pruning process, $\mathbf{C}_i$ is used to denote the current top-M discovered connections for user L_i, while $\mathbf{C}_i^{'}$ represents the previous ones. If the following inequality is satisfied for user L_p, it is not necessary to do the heap search for L_p:

$$S(L_i, L_p) < \min_q S(L_i, L_q) \quad \forall q \in \mathbf{C}_i^{'}, p \notin \mathbf{C}_i^{'} \tag{9}$$

Through this pruning process, the efficiency of heap search to generate $\mathbf{C}_i$ will obviously improve because the $N^{(u)}$ users needed in heap search decreases, and this will be very helpful for real-world applications. In summary, the procedure of an efficient online connection discovery framework is described in Algorithm 3:

ALGORITHM 3: Online Connection Discovery

```
Input: The video $V$ shared by user $i$
Output: Recommendations $\vec{C_i}$ for user $i$
for keyframe image vector $\vec{x} \in V$ do
    $\Delta L_v \leftarrow \vec{0}$ ;
    label($\vec{x}$) $\leftarrow$ nearest_cluster($\vec{x}$) ;
    $\Delta L_v \leftarrow \Delta L_v$+label($\vec{x}$) ;
    $L_v \leftarrow L_v + \Delta L_v$ ;
    for $L_j \in \mathrm{N}^{(u)}$ do
        $S(L_i, L_j) \leftarrow$ updateSim($S(L_i^{'}, L_j)$) ;
    end
end
$\vec{C_i}$ = topJConnections($L_j$) ;
```

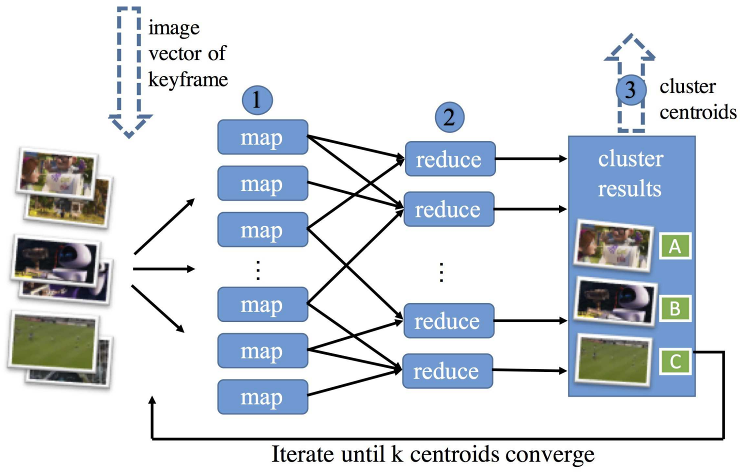


Fig. 11. Offline computation for keyframe clustering

4.3 Offline Computation

As shown in Fig. 4(b), computationally intensive tasks are performed periodically in an offline cloud platform, where Hadoop[1] is used in this paper. As offline computation can have a relaxed time requirement in the whole framework, it will perform the clustering on the cloud platform so that the online computation can utilize the clustering result immediately. Besides the computationally intensive parts, offline computation will also update the similarity and the sorted list time by time. This section will present the details of the offline implementation on Hadoop.

4.3.1 Cloud-assisted Keyframe Clustering. This section presents a distributed framework of the clustering algorithm, k-means, on Hadoop cluster, as shown in Fig. 11. The Hadoop cluster contains one master node to coordinate the jobs and several slave nodes to execute the jobs submitted from the client. Each slave node can have multiple workers to run part of the job. The input is a large file that contains lots of image vectors generated from feature extraction, which is represented in step 5 of Fig. 3. Each numeric vector represents an image. The first process is a *map* transformation in Hadoop, as in step 1 in Fig. 11. This *map* transformation in each worker is to calculate the Euclidean distance between each vector and all the centroids, and assign the vector to the cluster whose centroid has the minimum distance with the vector. Once the assignments for all vectors are done, the *reduce* step, as in step 2 of Fig. 11, can be performed to calculate the new centroid for each cluster by combining the points within the same cluster. Thus the clustering will iterate until the centroids converge. Once the clustering result is obtained, the offline computation will send the result to the online computation and continue distribution calculation, as shown in step 3 of Fig. 11.

4.3.2 Generating Candidate Lists. As described above, the offline processing will be periodically updated. Besides clustering in offline, similarity calculation and the sorted list (step 4 and 5 of Fig. 8(b)) for each user will also be updated once the clustering is done, such that the whole offline processing is updated periodically. Once the clustering is done, the BoFT labels will be assigned to the users who have corresponding images. With these BoFT labels, the framework can easily construct the user distributions in a vector format by counting the occurrences of each label.

[1]Hadoop: https://hadoop.apache.org/.

From Section 3.3, the similarity between two users L_i and L_j is updated periodically based on the cosine similarity in Eq. 3. The sorted list for each user L_i can then be generated in descending order according to the similarity between L_i and all other users.

4.4 Communication between Online and Offline Computation

This section describe how the online and offline computations are connected and how they facilitate an efficient connection discovery. As shown in Fig. 4, communications between online and offline happen in two places, when the image features are received and when the clustering results are generated. A synchronization process from offline to online will also occur when the sorted lists in the offline computation are updated. When the image feature vectors are obtained, the framework distributes the features to offline and the offline computation will save them to buffer for processing. Once the buffer is full or the preset time is up, the offline computation will combine all the old images and the new images together to undertake the process from step 1 to step 3 of Fig. 4(b).

While the offline computation is generating the sorted lists for all users, the clustering results will be sent to the online computation (step 3 of Fig. 4(b)). Once the online computation has received the clustering results, it stores the results into memory in the online server and replaces the previous clustering results. Once the offline computation is done, the sorted lists will also synchronize for all users in the online computation so that the framework is also available for the first kind of request described in Section 4.

5 PERFORMANCE EVALUATION

This section first presents the setup for the experiments and then introduces the social network dataset used in the experiments. The experiment uses follower/followee recommendation to evaluate the connections obtained with the proposed method. Some methods are implemented as a baseline to compare their performance with the proposed framework using the four metrics defined in this section. Finally, the evaluation results are presented to shows the efficiency of the proposed framework.

5.1 Experimental Setup

The dataset used in the experiments is collected from Twitter, a general social network which allows users to share multiple forms of content, such as text, images, and videos. To evaluate the discovered connections from shared videos by follower/followee recommendation, a set of 490 users are randomly selected using the Twitter API. There are 1,542 follower/followee relationships among the set of users, and 35,330 shared videos by them are collected. These videos were initially shared on various platforms such as YouTube and Vine, and the links to those videos were shared as tweets on Twitter by the 490 users. Each video is shared by only one user. The 35,330 videos have different content, sizes and resolutions, and are good representatives of videos shared on social media. For all the experiments, the ground truth follower/followee relationships are assumed hidden, and the connections between users are discovered using only user-shared videos. The recommendation of follower/followee relationships is based on the discovered connections and compared with the ground truth for evaluation.

The proposed analytics and streaming framework are implemented. The experiments on the proposed computation framework are performed in Amazon AWS clusters. For the Streaming Processing, 9 VMs are deployed, one as master node and another 8 VMs as slave nodes. The master node works as Tracker and the slave nodes works as Slave Servers as in Fig. 5. For the Online Computation, one single machine is deployed to aggregate the results from Streaming Process and update the user profile and connections incrementally. For the Offline Computation, another Amazon AWS cluster is deployed with 1 master node and 8 slave nodes to periodically conduct image clustering and update the user profile and connections from scratch. Each node has four virtual CPUs running at 2.4GHz, with an 16 GB RAM, a 250GB hard disk and a Gigabit network interface card. Requests for uploading videos are from local machine. In order to evaluate the framework, the experiments compares a baseline non-streaming BoFT and a keyframe-level streaming framework proposed in [24] with the proposed streaming BoFT:

- **Baseline**: The baseline BoFT connection discovery framework shown in Fig. 3 is implemented on a stand-alone machine to be compared with the proposed framework. The baseline BoFT uses a non-streaming method to upload and process videos. The uploading process is in series, and the computation starts when the videos are completely uploaded. The baseline runs the complete process in a single machine and does not distribute any parts of the computations in BoFT.
- **Marlin**: Marlin, proposed in [24], is a streaming data processing pipeline for content-based video similarity search. Unlike our proposed streaming framework, Marlin employs keyframe-level parallelism. The system acellerates the computation by distributing the feature extraction of a certain video at keyframe level and further aggregates the keyframe features to get the feature representation of the videos. However, the major bottleneck in video uploading and keyframe extraction is not dealt with in the system.
- **Proposed**: The proposed BoFT framework uses a streaming method to upload and process videos. The uploading process segments each video into approximately equal chunks and uploads the chunks in series to different servers for distributed processing. When one chunk finishes uploading, the processing of this chunk starts while the next chunk starts to upload. The proposed BoFT framework further improves the scalability by using online computation to update user profiles and similarity and generates recommendations instantly, while using offline computation to distribute the clustering process via Hadoop MapReduce periodically.

5.2 Evaluation Metrics and Results

The experiments in this paper use three metrics to evaluate the performance of the proposed framework. The first one is the processing time, t_p, which is measured from the time a video starts to upload to the time that recommendation is made, as shown in Fig. 4. The second one is the precision for evaluating how accurate the recommendations are. Precision is defined as the number of true predicted follower/followee relationships (T_p) divided by the total number of predicted follower/followee relationships ($T_p + F_p$):

$$Precision = \frac{T_p}{T_p + F_p}, \tag{10}$$

where F_p is the number of falsely recommeded follower/followee relationships. Also the metric of measuring the processing time per video $t_{pi} = \frac{t_p}{n}$ is given. There is another variable, which is the percentage of total keyframes extracted from videos. The experiment measures the recommendation precision by using partial video data to demonstrate how much video data is necessary to achieve comparable precision. As this paper claims a the framework that provides an efficient connection discovery, the framework is expected to have a fast and stable processing time for each video input.

5.2.1 Effectiveness of Video-based Connection Discovery. In order to show the effectiveness of the proposed method for connection discovery using user-shared videos, an experiment for evaluating the recommendation precision is performed. In Fig. 12, the precision of the video-based recommendation is compared with random recommendation. As expected, for the proposed connection discovery with user-shared videos, as the number of recommendation made increases, the precision of recommendation goes down. And the video-based recommendation has 50% higher

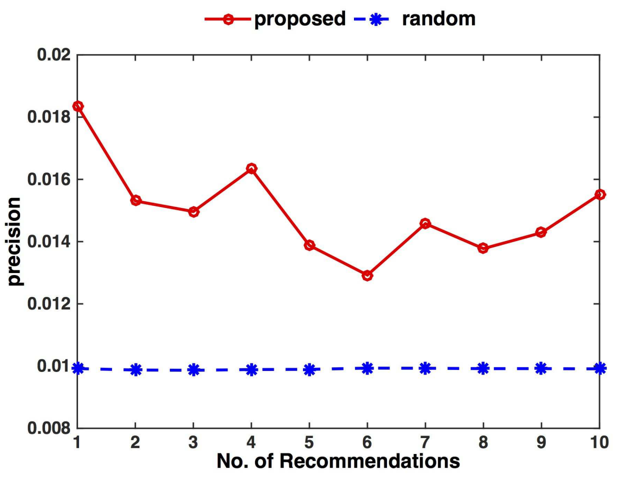


Fig. 12. Effectiveness of video-based connection discovery

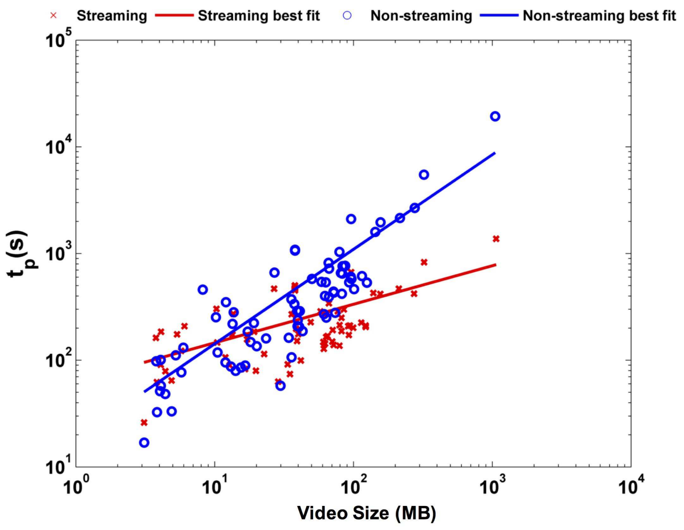


Fig. 13. Processing Time on Twitter with Non-stream and Stream Processing

precision than random recommendation on average when the number of recommendations is below 10, which validates the feasibility of the proposed method.

5.2.2 Processing Time for Streaming. In order to investigate the benefit of the proposed Streaming Processing, the processing time for streaming is measured separately by experiment. The processing time in the experiments includes the video uploading time and the chunk process time. The experiment focuses on the processing time comparison of non-streaming method and streaming method. For baseline non-streaming BoFT, the computation starts when the whole video finishes uploading, till step 5 in Fig. 2 where visual information is extracted. And for the proposed framework, the uploading and computation is performed in streaming style, in which the videos are segmented

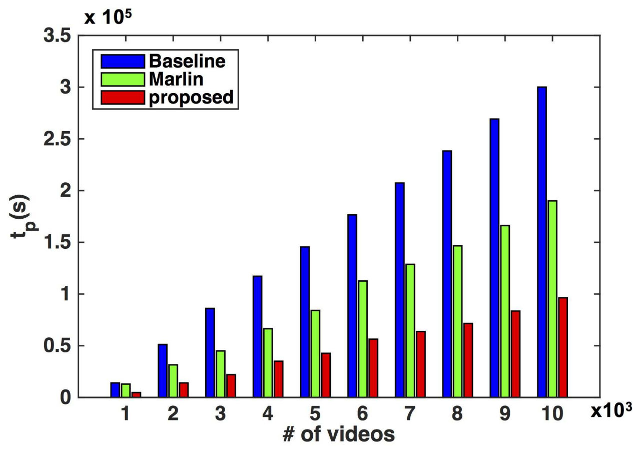


Fig. 14. Overall processing time of Baseline, Marlin and the proposed streaming framework

into chunks of about equal size. The chunk size is set to about 10MB in this experiment. The chunks are uploaded in series and in order, meaning when one chunk finishes uploading it will be processed and in the meanwhile the next chunk starts to upload. In terms of investigating the processing time of the baseline non-streaming framework, and proposed streaming framework, the experiment first randomly samples 100 videos and examines the processing time of each video. Fig. 13 shows the processing time of the baseline non-streaming BoFT and proposed streaming BoFT introduced above. It is observed that when the video size is very small, the process time of the proposed streaming framework is actually slightly longer than the baseline non-streaming framework due to the overhead of the system. However, when the video size is larger than 10MB, the advantages of the proposed streaming framework start to be revealed. And as the video size increases, it is apparent that the proposed streaming BoFT framework outperforms the baseline non-streaming BoFT framework. The proposed framework takes only 35% as much time as the non-streaming framework on average in terms of the processing time.

5.2.3 Processing Time for Overall Framework. In this experiment, the proposed overall streaming framework, including Streaming Processing and Online/Offline Computation, are compared with the Baseline, single machine framework, and Marlin, keyframe-level streaming. The processing time of the three methods is shown in Fig. 14. From the results, it can be seen that Marlin indeed outperforms Baseline framework, spending only 63% as much time as Baseline. However, our proposed streaming framework significantly outperform these two methods. The proposed streaming framework spends only 50% as much time as Marlin, and only 32% as much time as Baseline. The results validates the superior acceleration of our proposed streaming framework and and the contribution of this work.

5.2.4 Processing Time Per Video. As described in the beginning, t_{pi} is the processing time divided by the number of videos. The experiment evaluates the average processing time of videos that fall into a particular video size range. The average processing time measures the time from when an video starts to upload to when the connections are found. The following description in this section will simply use the average processing time to denote the processing time per video. Fig. 15 shows the average processing time ratio with the processing time of the proposed framework divided by the baseline framework. The average video size of all videos in the dataset in the experiment is 65MB. As shown in Fig. 15, on this social network the average processing time per video of the

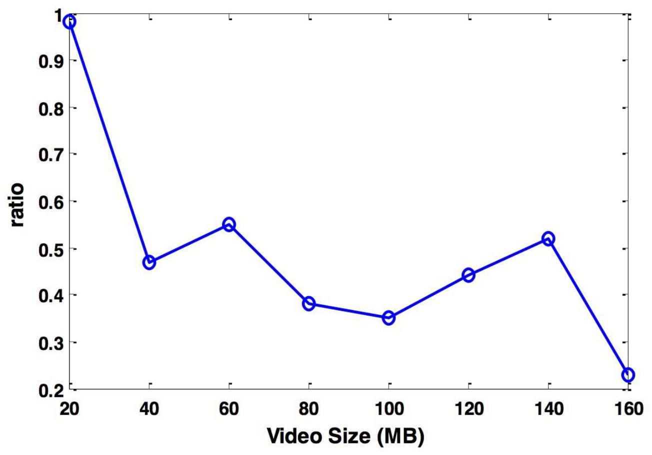


Fig. 15. Ratio of processing time per video between proposed framework and baseline

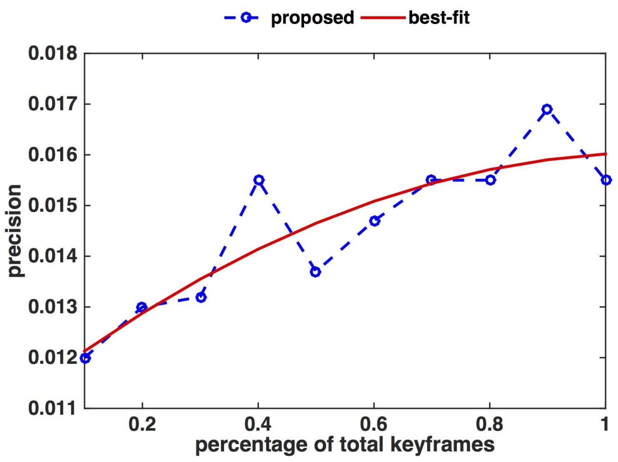


Fig. 16. Precision using partial keyframes of video

baseline non-streaming BoFT framework is evidently longer than that of the proposed streaming BoFT framework when the video size is beyond 20MB size range. In fact, the proposed framework takes only 32% as much time as the baseline non-streaming framework to upload and process a video on total average. For time-sensitive applications, it is very important to rapidly learn factors for new items (e.g. event updates, tweets, images) so that the applications can give a satisfactory user experience and attract more users [1, 9]. With the stable and low processing time per video, the framework can easily scale and handle large amounts of data in an efficient fashion.

5.2.5 Precision with Partial Video Data. This experiment evaluates the precision of recommended relationships compared with the ground truth on the Twitter dataset with partial video data information. As discussed previously, the videos generally contain much redundancy and by exploiting partial video data connection discovery can be made without even waiting for the videos to finish uploading. Since the keyframes are generally extracted in order in our framework, the percentage of keyframes used in this experiment are those starting from the beginning. That is with 10% of total keyframes, the first 10% of keyframes are used, and so on. In this manner, we evaluate at what percentage of the videos having been uploaded can the recommendation be made. It is shown in Fig. 16 that with more complete video data the precision of recommendation is better and with only 70% of video data the precision is almost as good as using complete video data. Therefore, instant response can be made exactly when, or even before, the user finishes uploading the video.

6 DISCUSSIONS

This section discusses potential problems and highlights future research directions. The general ideas of the analytics is based on the intuition that the videos users share are normally what they like, e.g. it is the case in Twitter and Instagram. Thus, the users' preferences can be inferred from their shared videos, and follower/followee relationships can be recommended based on similar preferences. The proposed methods can be used in applications that have such property. The paper mainly focuses on the visual information of a video, whereas other multimedia information, such as audio and textual information, could also be utilized for connection discovery. This other multimedia information could be incorporated to help understand the content of a video more deeply and find the similarity between users more effectively. For example, one of the most important types of information is audio. It is observed during the experiment that a proportion of videos have only static visual images with music, such as popular songs. Such videos cannot be handled by the proposed approach as the visual information of the video is not representative, whereas th audio information has high distinctiveness.

The streaming process proposed in the paper is an efficient technique for processing multimedia data in applications that require instant response. These videos used in the experiments have different durations, qualities and content, and are therfore good representatives of what is being shared on social media today. The proposed framework can be easily applied to different social media platforms with video sharing features for connection discovery, and many time-sensitive applications that depend on connections become possible. Furthermore, the video representation using BoFT can be saved as video signatures, and when users upload videos duplicated in the database, the video signatures can be directly used without going through video representation process.

To deploy the streaming framework to practical systems, some insights are provided here. First, in the proposed framework, multiple servers are used for video streaming uploading and chunk video processing. The chunk video processing in the experiment is mainly based on CPUs. However, with GPU parallelism, the video processing for each chunk can be further improved. Second, task scheduling is an important topic in big data system. The scheduling policy of the tracker server in the proposed framework is currently first-come-first-serve. It may not be an optimal solution since scheduling a large video request before several small videos might decrease the overall user satiesfaction. This can be replaced with more sophisticated policies depending on different applications. Also, it would be worthwhile to investigate how to minimize the overhead among the servers, such as optimizing the chunk size. Lastly, the streaming processing and online computation serve as a front-end of a real-time system, where the data are collected and analyzed. On the back-end, both the video data and video representations need to be stored in efficient database. Also, large-scale data move between the database and map-reduce servers in offline computation has to be maintained periodically.

7 CONCLUSION

This paper proposes to employ user-shared videos for connection discovery by using the BoFT method to generate video representations. The effectiveness has been proved through experiments. An efficient computation framework is proposed for connection discovery using user-shared videos. The framework is designed separately by Steaming Processing, Online and Offline Computation. Streaming Processing is proposed to utilize the video uploading time and speed up the processing. The computationally intensive parts are handled offline on cloud platform, and the Online Computation incrementally performs updating for connection discovery in an efficient fashion. Experiments show that the BoFT method with user-shared videos can achieve significantly higher precision than random recommendation on average in Twitter dataset. The proposed framework takes on average only 30%~50% as much time as existing frameworks in terms of the processing time. Also the recommendation precision with partial video data is evaluated. It is concluded that even without completion of full video uploading, the recommendation can be made. The advantages of the proposed framework with shorter processing time and less video data make it suitable for time-stringent applications in the real world.

APPENDIX

In this appendix, the incremental implementation of similarity updating between user i and user j is explained as follows. With an input of video v belonging to user i, the similarity between user i and user j is updated as

$$S_{i,j} = \frac{L'_i \cdot L_j + \frac{L_v \cdot L_j}{||L_v||}}{||L_i|| \cdot ||L_j||}, \tag{11}$$

where $L'_i \cdot L_j$ is already known. The calculations of $L_v \cdot L_j$, $||L_v||$ and $||L_i||$ are implemented incrementally. That is,

$$L_v \cdot L_j = L'_v \cdot L_j + l_{j,k}, \tag{12}$$

where $L'_v \cdot L_j$ is the dot product of the previous result. The two norms are updated as

$$\begin{aligned} ||L_v||^2 &= ||L'_v||^2 + 2l'_{v,k} + 1, \\ ||L_i||^2 &= ||L'_i||^2 + 2 \cdot \frac{L'_i \cdot L_v}{||L_v||} + 1. \end{aligned} \tag{13}$$

where $||L'_v||$ and $||L'_i||^2$ are already known and $L'_i \cdot L_v$ is updated in the same way as $L_v \cdot L_j$. With the above updates, the similarity update with an incoming keyframe label of a video can be done in constant time, as in Eq. 8.

ACKNOWLEDGMENTS

This work is supported by HKUST-NIE Social Media Lab., HKUST.

REFERENCES

[1] Deepak Agarwal, Bee-Chung Chen, and Pradheep Elango. 2010. Fast online learning through offline initialization for time-sensitive recommendation. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 703–712.

[2] Vinti Agarwal and KK Bharadwaj. 2013. A collaborative filtering framework for friends recommendation in social networks based on interaction intensity and adaptive user similarity. *Social Network Analysis and Mining* 3, 3 (2013), 359–379.

[3] Ming Cheung and James She. 2014. Bag-of-Features Tagging approach for a better recommendation with social big data. In *IMMM 2014, The Fourth International Conference on Advances in Information Mining and Management*. 83–88.

[4] Ming Cheung, James She, and Zhanming Jie. 2015. Connection discovery using big data of user-shared images in social media. *Multimedia, IEEE Transactions on* 17, 9 (2015), 1417–1428.

[5] Eric Gilbert. 2012. Predicting tie strength in a new medium. In *Proceedings of the ACM 2012 conference on Computer Supported Cooperative Work*. ACM, 1047–1056.

[6] Eric Gilbert and Karrie Karahalios. 2009. Predicting tie strength with social media. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 211–220.

[7] Liang Gou, Fang You, Jun Guo, Luqi Wu, and Xiaolong Luke Zhang. 2011. Sfviz: interest-based friends exploration and recommendation in social networks. In *Proceedings of the 2011 Visual Information Communication-International Symposium*. ACM, 15.

[8] William H Hsu, Andrew L King, Martin SR Paradesi, Tejaswi Pydimarri, and Tim Weninger. 2006. Collaborative and Structural Recommendation of Friends using Weblog-based Social Network Analysis.. In *AAAI Spring Symposium: Computational Approaches to Analyzing Weblogs*. 55–60.

[9] Junzhong Ji, Zhiqiang Sha, Chunnian Liu, and Ning Zhong. 2003. Online recommendation based on customer shopping model in e-commerce. In *Web Intelligence, 2003. WI 2003. Proceedings. IEEE/WIC International Conference on*. IEEE, 68–74.

[10] Zhanming Jie, Ming Cheung, and James She. 2015. A cloud-assisted framework for bag-offeatures tagging in social networks. In *IEEE 4th Symposium on Network Cloud Computing and Applications*.

[11] Jason J Jones, Jaime E Settle, Robert M Bond, Christopher J Fariss, Cameron Marlow, and James H Fowler. 2013. Inferring tie strength from online directed behavior. *PloS one* 8, 1 (2013), e52168.

[12] R. Kuschnig, I. Kofler, and H. Hellwagner. 2011. Evaluation of HTTP-based request-response streams for internet video streaming. In *Proceedings of the second annual ACM conference on Multimedia systems*. ACM, 245–256.

[13] Xiaopeng Li, Ming Cheung, and James She. 2016. Connection Discovery using Shared Images by Gaussian Relational Topic Model. In *International Conference on Big Data*. IEEE, 931–936.

[14] Gentao Liu, Xiangming Wen, Wei Zheng, and Peizhou He. 2009. Shot boundary detection and keyframe extraction based on scale invariant feature transform. In *Computer and Information Science, 2009. ICIS 2009. Eighth IEEE/ACIS International Conference on*. IEEE, 1126–1130.

[15] David G Lowe. 2004. Distinctive image features from scale-invariant keypoints. *International journal of computer vision* 60, 2 (2004), 91–110.

[16] Jirı Matoušek. 2000. On approximate geometric k-clustering. *Discrete & Computational Geometry* 24, 1 (2000), 61–84.

[17] Xueming Qian, He Feng, Guoshuai Zhao, and Tao Mei. 2014. Personalized Recommendation Combining User Interest and Social Circle. *IEEE Transactions on Knowledge and Data Engineering* 26, 7 (2014), 1763–1777.

[18] Jitao Sang and Changsheng Xu. 2012. Right buddy makes the difference: An early exploration of social relation analysis in multimedia applications. In *Proceedings of the 20th ACM international conference on Multimedia*. ACM, 19–28.

[19] Xiance Si and Maosong Sun. 2009. Tag-LDA for scalable real-time tag recommendation. *Journal of Computational Information Systems* 6, 1 (2009), 23–31.

[20] Yang Song, Ziming Zhuang, Huajing Li, Qiankun Zhao, Jia Li, Wang-Chien Lee, and C Lee Giles. 2008. Real-time automatic tag recommendation. In *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*. ACM, 515–522.

[21] C Sujatha and Uma Mudenagudi. 2011. A study on keyframe extraction methods for video summary. In *Computational Intelligence and Communication Networks (CICN), 2011 International Conference on*. IEEE, 73–77.

[22] Rongjing Xiang, Jennifer Neville, and Monica Rogati. 2010. Modeling relationship strength in online social networks. In *Proceedings of the 19th international conference on World wide web*. ACM, 981–990.

[23] Xing Xie. 2010. Potential friend recommendation in online social network. In *Green Computing and Communications (GreenCom), 2010 IEEE/ACM Int'l Conference on & Int'l Conference on Cyber, Physical and Social Computing (CPSCom)*. IEEE, 831–835.

[24] N. Zhu, W. He, Y. Hua, and Y. Chen. 2015. Marlin: Taming the big streaming data in large scale video similarity search. In *International Conference on Big Data*. IEEE, 1755–1764.

[25] Jinfeng Zhuang, Tao Mei, Steven CH Hoi, Xian-Sheng Hua, and Shipeng Li. 2011. Modeling social strength in social media community via kernel-based learning. In *Proceedings of the 19th ACM international conference on Multimedia*. ACM, 113–122.

Received February 2017; revised May 2017; accepted June 2017