An Efficient Computation Framework for Connection Discovery using Shared Images

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

With the advent and popularity of the social network, social graphs become essential to improve services and information relevance to users for many social media applications to predict follower/followee relationship, community membership, etc. However, the social graphs could be hidden by users due to privacy concerns, or kept by social media. Recently, connections discovered from user shared images using machine generated labels are proved to be more accessible alternatives to social graphs. But real-time discovery is difficult due to high complexity, and many applications are not possible. This paper proposes an efficient computation framework for connection discovery using user shared images, which is suitable for any image processing and computer vision techniques for connection discovery on the fly. The framework includes the architecture of online computation to facilitate real time processing, offline computation for a complete processing, and online/offline communication. The proposed framework is implemented to demonstrate its effectiveness by speeding up connection discovery through user-shared images. By studying 300K+ user shared images from two popular social networks, it is proven that the proposed computation framework reduces 90% of runtime with a comparable accurate with existing frameworks.

CCS Concepts: • **Human-centered computing** → **Collaborative and social computing systems and tools**; • **Information systems** → *Multimedia information systems*; • **Social and professional topics** → User characteristics;

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

ACM Reference format:
Ming Cheung, Xiaopeng Li, and James She. 2017. An Efficient Computation Framework for Connection Discovery using Shared Images. *ACM Trans. Multimedia Comput. Commun. Appl.* 1, 1, Article 1 (January 2017), 21 pages.
https://doi.org/https://doi.org/10.1145/3115951

1 INTRODUCTION

Social media becomes prevalent among people in our daily live nowadays. People can discover their like-minded friends on Facebook, Twitter, Weibo, etc. These social media sites help users find their potential relationships, such as online friendships, follower/followee relationship and community memberships, based on their social graphs (SGs), a form of connection, and interactions with others

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

1551-6857/2017/1-ART1 $15.00
https://doi.org/https://doi.org/10.1145/3115951

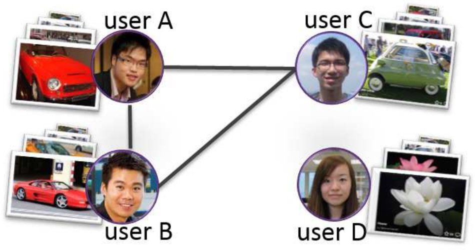


Fig. 1. An example of user shared images and user connections.

[2, 16, 17, 29, 42]. With all these social information of users, tons of interesting and personalized applications have been developed to make life convenient in real world [10, 31, 35, 39, 43]. Obviously, all these applications can have a better understanding of users with more user participation under this big data era with sufficient computing resources. However, SGs could be hidden by users due to their privacy concern, or kept by social media, which make it hard for companies to taste the benefits of social networks.

Without the access to SGs, it is still possible to discover another form of connections between users. Intuitively, similar users may have similar user behaviors and interactions on the sites. Thus using these interactions and other side information (e.g., location) can model the strength of connections between users [14, 22, 41]. Some media applies a general tie strength model for their own applications [13]. In [32], the authors learn the user interests from the shared content. In [46], the authors use the mean of the feature vectors, along with other features such as comments to calculate the similarity of users. However, the features applied in those works could be unavailable to other social media, which limits the use of those approaches. Recently, annotating machine generated labels using Bag-of-features Tagging (BoFT) on user shared images for connection discovery is proven to be a more accessible alternative to social graph [5]. Machine generated labels are assigned to those images, and the connections among users can be calculated based on the similarity of the occurrence of different labels. Fig. 1 shows some examples of user shared images and user connections, where user *A*, *B* and *C* share a lot of car images while user *D* shares a lot of flower images and few car images. Inferring from their shared images, it is more likely that user *A*, *B* and *C* have relatively stronger connections. With connections between users available, relationships between users can be predicted and many applications such as follower/followee recommendation and gender identification are possible using the discovered connections [6].

The generation of machine generated labels is not limited by bag-of-features, but also many other techniques such as GIST and color-based techniques that convert images to a feature vector[7]. The images are then assigned with a machine generated label by clustering all the user shared images and assigning each image a cluster label. There are two major challenges in order to build a practical system with this method for connection discovery. The first is scalability problem. The amounts of users in social network sites are millions or more. Shared images by those users are enormous, billions or more. In order to assign machine generated labels to shared images, clustering has to be conducted over all images. The process could be extremely slow even with cloud-assisted computing. Without efficient computation and delicate design, it is impractical to build an online connection discovery system with the BoFT technique. The second problem is that shared images are continuously generated. For example, in March 2013, Flickr had a total of 87 million registered members and more than 3.5 million new images uploaded daily. With more and more newly generated images, the concepts of image clusters might also drift. One way to incorporate the newly generated images is to rebuild the model with all images. Rebuilding (re-clustering), however, is typically an expensive task and should not be done too frequently. How to incorporate the fast generated images for connection discovery while making the computation feasible for real time requirement is a big challenge.

In order to resolve above challenges, this paper presents a fast and scalable connection discovery framework using machine generated labels, and demonstrates the efficiency of the framework for follower/followee recommendation using BoFT. With the purpose of instant response, the framework manages the parts with intensive computation in offline and discovers connections by combining the results of the offline process and streaming image input. In online connection discovery, the framework can efficiently assign suitable labels for the images posted by user i. Then the similarity between other users and user i will be recomputed based on the labels they have as an instantly updated result. With the connections between users available, many applications such as gender prediction, collaborative recommendation, user-targeted advertising and user search can generate instant results to users. According to [6], it has been proved that 2 users with a high similarity are more likely to be follower/followee than 2 users with a low similarity[6]. Thus through evaluating the performance in the application of follower/followee recommendation, the validity of discovered connections can be demonstrated. The most similar J users are selected for user i as the recommended follower/followee relationships. In summary, this paper makes the following contributions:

- Proposed a novel computation framework for connection discovery using user shared images with an online/offline architecture that achieves instant response, in which the framework can be applied to any computer vision/image processing techniques;
- Implemented the proposed framework and demonstrates its effectiveness by scaling and speeding up connection discovery through user-shared images with cloud;
- Devised an efficient online algorithm to incrementally update the parameters of the analytics with streaming input of user shared images for connection discovery; and
- Conducted several experiments with 300K+ user shared images from two popular social networks to evaluate the performance of the proposed framework, and proved that the proposed computation framework significantly improves the computation speed, reducing 90% of runtime with comparable precision.

The rest of this paper is organized as follows: Section II reviews the related work in research on some efficient or even real-time systems in social networks. Section III introduces how connections can be discovered using user shared images, and indicates the limitations of it. Section IV describes proposed computation framework for connection discovery and how such a general framework is designed and implemented. Section V reports the evaluation results of the comparative analysis with other frameworks on two social network datasets, using BoFT. Section VI discusses a parameter setting in experiments, a comparison of a related work and future works. This section further raises some problems of current framework and other alternatives that can probably improve the framework. Section VII concludes the paper.

2 RELATED WORKS

Recommendations, such as items [24], rating [45] and multimedia content [26, 33] have long been researched with the use of social graphs. However, due to user privacy concern, or policy of social media, the social graphs are only available for exclusive parties. Even the social graph is publicly accessible, social connectivity of its users is very low that is not usable for many applications [26]. Recently, the use of visual appearance has been proven to be promising as an alternative for recommendation [5, 40, 44, 46]. However, the high computation load of analyzing images reduces the usefulness of the analytics.

Substantial research efforts have focused on real-time processing framework in social networks. Since complex computations are targeted on a large amount of data in real time, the right types of infrastructures are needed to be explored. Thus a real-time system with the desired expressiveness and scalability is needed. [12] developed a social information discovery system for social network analysis in a very quick response time by accelerating the extraction, transformation and loading (ETL) process. [38] proposed a real-time website security protection mechanism based on the concept of proxy to prevent an attack from some suspicious web pages. [37] proposed a highly-automated framework for real-time tag recommendation. The framework is divided into two parts: offline computation and online recommendation. A mixture model for document classification is built based on the distribution of documents and words during the offline stage. In the online recommendation stage, a document is classified according to predefined clusters acquired in the offline stage. Online/offline framework is also applicable to other applications, such as location recommendation [20]. On the other hand, [34] proposed a scalable and real-time method for tag recommendation. This method created a tag-LDA model and implemented a distributed training tool for the model using Hadoop MapReduce framework such that the recommendation can be done in real-time by only using model parameters. However, their method cannot handle continuously generated data from the web in that the model does not update periodically.

In terms of online friendship and follower/follower relationship recommendation, there are also plenty of research groups have contributed a lot to achieve those recommendations in a very fast way. [42] designed a general connection recommendation framework to recommend potential online friendship and follower/follower relationship in real online social network. The recommendation is achieved in real time by monitoring users' new adding connection to know when to recompute the recommendations. [30] considered the real-time physical location proximity and historical check-in behavior similarity in the location-based mobile social network to facilitate users to make suitable friends from the surrounding people in a real time. Both of the work help users to find their potential friends in real-time. Those works are significant but there is no technique to speed up their algorithms for a large scale data. Understanding the community structure and dynamics of social networks is vital for the design of related applications and devising business strategies. [25] empirically analyzed a scalable, efficient and accurate community detection algorithm and observed how it can be potentially applied online in large-scale and real-time social networks. Google designed a scalable online collaborative filtering algorithm to build online personalized recommendation engine on a large web property like Google News [11]. Their real time recommendation system consists of three components: an offline component that is responsible for periodically clustering users; a set of online servers; and data tables for mapping user ids and news ids. In the system architectures of Netflix [3], there are also three main computational component: online computation, offline computation, and nearline computation. Online computation can respond to requests in real-time while offline computation trains a model in a batch manner with relaxed timing requirement. Unlike the data tables in Google news recommendation system [11], nearline computation is an intermediate compromise between online and offline mode in which it can do online-like computations, but do not require them to be served in real-time.

This paper designed a framework for Bag-of-Features Tagging approach by combining online and offline computations so as to make a faster connection discovery compared to the original framework using the most accessible data on social media, i.e. user shared images. To the best of our knowledge, this is the first work that explicitly proposes an efficient computation framework applied on Bag-of-Features Tagging approach for connection discovery in social networks.

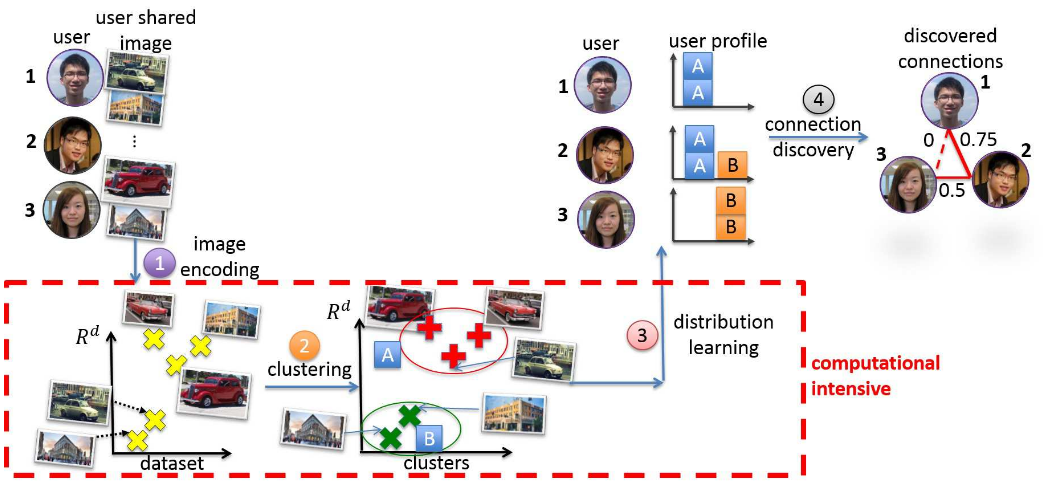


Fig. 2. An overview of connection discovery [5]

3 CONNECTION DISCOVERY USING MACHINE GENERATED LABELS

This section introduces how to generate machine generated labels to user shared images, as well as how to recommend follower/followee based on those discovered connections. Although different computer vision approaches are proven to be able to generate machine generated labels, this paper applies BoFT to demonstrate the proposed framework. The section first introduces how machine generated labels can be assigned to images, followed by connection discovery with machine generated labels and the limitations of it.

3.1 Assigning Machine Generated Labels

The first step of machine generated labels is image encoding, where images are encoded as feature vectors. One of the possible ways is using a Scale-Invariant Feature Transform (SIFT)-based approach [28]. The generation of machine generated labels is not limited by SIFT, other techniques such as GIST and color-based techniques can also encode images into feature vector[7]. In computer vision/image processing tasks such as object recognition, classification is based on the feature vector. But in connection discovery, the goal is to assign a machine generated label to an image, which is not necessary to be the objects on the image. The labels are assigned clustering, which groups images with similar feature vectors. Each cluster obtained in this operation corresponds to similar objects to which a machine-generated label is assigned. After obtaining the cluster in step 4 of Fig. 2, the images in any clusters are assigned with the same machine generated label to reflect that they are visually similar and belong to the same group. It is an unsupervised operation and no assumption is made or information on the image is known.

3.2 Connection Discovery with Machine Generated Labels

The label distribution, which reflects the visual features of user shared images, is the key in the content recommendation. In connection discovery, it is assumed that no user input is needed and the label distribution is the occurrence of each label as in step 5 of Fig. 2. For example, the label distribution of user i is represented by a vector L_i.

$$L_i = (l_{i,1}, \ldots l_{i,k}, \ldots l_{i,K}) \tag{1}$$

Each element $l_{i,k}$ in L_i is the number occurrence of label k in the shared images of user i. When the distribution is obtained, the next step is to calculate the similarity between users based on their distributions. The similarity between two users is calculated by the cosine similarity:

$$S_{i,j} = S(L_i, L_j) = \frac{L_i \cdot L_j}{||L_i|| \cdot ||L_j||} \tag{2}$$

where L_i and L_j are the distributions of user i and j, respectively. Two users who share similar images have a higher $S_{i,j}$., and are more likely to be follower/followee[6]. Finally, a similarity graph based on label distributions is generated as in Fig. 2 and the most similar J users will be chosen for each user i as the recommendation.

3.3 Limitations

As shown in step 4 of Fig. 2, connection discovery requires many computational and storage resources because all the data need to be stored in memory and many iterations are involved in the clustering process. When the dataset is very large, such as billions of images, the whole process will last several hours, or longer, even without the step of preprocessing the images. Besides the number of images, similarity calculation is also computationally intensive when there are millions of users in social networks. A stand-alone machine definitely cannot efficiently process billions of images and millions of users in real-world social networks. The computational parts should have a relaxed time requirement so that the framework can be accelerated. Therefore, a more efficient computation framework is proposed in this paper to help to solve the above problems for the approach.

4 PROPOSED COMPUTATION FRAMEWORK FOR CONNECTION DISCOVERY

This section describes how assigning machine generated labels shown in Fig. 2 can be improved into a much more efficient fashion. Fig. 3 shows the proposed framework on cloud. The proposed framework is not only available for connection discovery, but also for other applications like item recommendation, user-target advertising, etc (Fig. 3(a)). There are two main computations in the framework to facilitate efficient processing: an online computation that is required to respond to users instantly and an offline computation that is responsible for periodically update the information that does not require strict timing requirement. How online and offline communication is very essential to connection discovery and this section will elaborate the detail inside.

4.1 Online Computation

In the implementation, the online computation contains one online servers to process users' requests while it is also flexible to have many online server to support many user requests. The online computation comprises of four main parts: assigning machine generated labels for the incoming images, updating the label distributions of each user, recalculating the similarity and sorting the list of discovered connections based on the similarity. To further elaborate why such a framework (Fig. 3) is designed in this paper, two types of requests from users are defined here:

(1) user logins and automatically receives recommendations from the discovered connections.
(2) user shares several images and the connections are discovered within an instant time interval.

In terms of the first type, only a sorted list of discovered connections for the user who raised the request needs to maintain in offline to present the discovered connections. The sorted lists for all users would be updated periodically in offline with the incoming images in online as shown in step 5 in Fig. 3. In other words, the offline computation will synchronize the sorted list for each user with the one in online whenever there is an updated list appears. The framework can simply extract the discovered connections from the list and send back to users as the response. For the latter case, the images shared by users first go through the feature extraction process (step 1 and

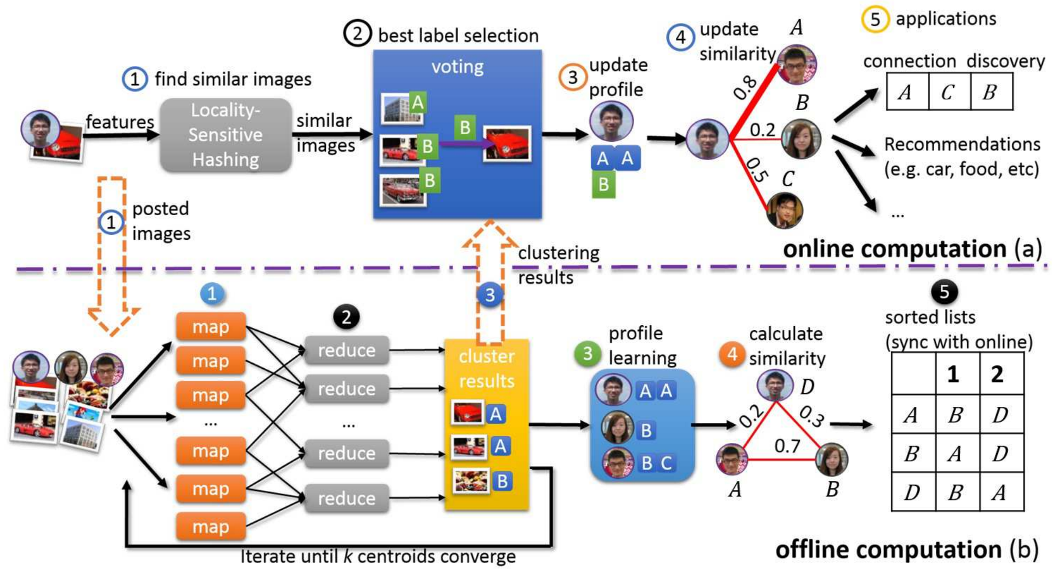


Fig. 3. The proposed computation framework of connection discovery with user shared images. (a): The online computation which contains 5 steps to efficiently discover connections with instant input. (b): The offline computation periodically runs the clustering, generates the sorted lists for online

2 of Fig. 2) such that visual words can be obtained to represent the images. A visual word of an image is a numeric vector, which means each image is denoted by a numeric vector before sending to the proposed framework. When the feature vector of an image is sent to the framework, the online computation will go through the four main parts to discover the connections for the users who posted the image.

4.1.1 Machine Generated Label Assignment. The first part is to assign a machine generated label to the incoming image. In order to incorporate the newly generated images into consideration for connection discovery, the user profiles and similarities have to be updated by re-estimating the parameters. Re-building the model whenever new data comes in, however, is computationally inefficient and is not an option for the system to be practical. Thus, an incrementally update method is proposed here. Since each image in offline already has a corresponding label by clustering, the new assignment can be made efficiently based on those labels. Thus this is a general classification problem for the new image, many classification algorithms [9, 23, 27] can be fitted into this part to classify the image. This paper divides this part into two sub-elements as step 1 and 2 in Fig. 3(a). The first part quickly locates some images which are similar to the incoming one and the second one will assign a machine generated label to the new image based on these the labels of these similar images.

(1) Search Similar Images: When an image vector is input to the framework, this element searches visually similar images to the input. The simplest and most naive solution to this problem is a linear search, which is to calculate the distance from the query image vector to every other image vectors with a label, keeping track of the near neighbors within a distance threshold. However, linear search has a running time of $O(KN)$ where N is the number of images and K is the dimension of image vector. This paper applies the classical method, Locality Sensitive Hashing (LSH). LSH is more efficient to locate similar images with high dimension as shown in step 1 of Fig. 3(a). The LSH

technique was introduced by Indyk and Motwani [18] to efficiently solve the near-neighbor search problem and many applications have used it for neighbor searching [4, 8, 15]. Given a query point, LSH can help find the points in a large dataset that are close to the query point. LSH guarantees a high probability that the neighbor is nearest for any query point. The key idea is to hash the data points using several hash functions so as to ensure that, for each function, the probability of collision is much higher for objects which are close to each other than for those which are far apart. Then, one can determine near neighbors by hashing the query point and retrieving elements stored in buckets containing that point.

LSH exhibits the locality-aware property which supports similarity aggregation. The images with similar features can be hashed into the same buckets with a higher probability. The hash function is defined by a random projection, $h^{\vec{x},b}(\vec{v}) = \left\lfloor \frac{\vec{x}\cdot\vec{v}+b}{w} \right\rfloor$. $\vec{x}$ is a vector with components that are selected at random from a Gaussian distribution. The number of hash functions N_h and the number of buckets N_b are the parameters while constructing the LSH. Within each set of N_h dot products, success is achieved if the query and the nearest neighbor are in the same bin in all N_h dot products. To reduce the impact of an "unlucky" quantization in any one projection, there are N_b independent projections and the neighbors are pooled from all of these projections. This is motivated by the fact that a true near neighbor will be unlikely to be unlucky in all the projections. By increasing N_b we can find the true nearest neighbor with arbitrarily high probability. Therefore with LSH, the framework can identify several images which are similar to the input.

(2) Best Label Selection: Selecting the best label for the incoming image among all the neighbors, the simplest one is to assign the label of the closest image in terms of distance. However, this will inaccurate if the point distribution is a skewed distribution. This can be solved by giving weights to images or labels so that the label of the incoming image will be a function of the neighbors and the correlate weights. This framework applies the *majority voting* method as shown in step 2 of Fig. 3(a). The percentage among all neighbors that are acquired in LSH is defined as β. For instance, $\beta = 50\%$ means that the nearest 50% neighbors are selected from all neighbors acquired in LSH. A detailed discussion can be found in Section 6. A voting process is applied to assign the most occurred label in the neighbor est of images as the label of the incoming image. Suppose the neighbor set generated from LSH and selected using β is $\mathbf{N}$, which means there are $\|\mathbf{N}\|_1$ neighbors in the voting process. The occurrences of each label i in $\mathbf{N}$ is l_i, so the following equation is used to select the best label among the neighbor set:

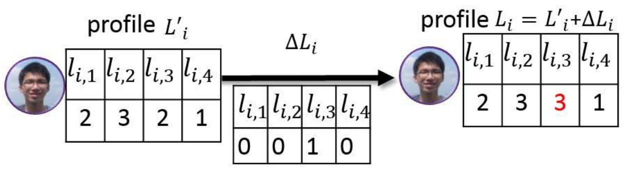
$$label = \arg\max_i l_i \quad \text{where } i \in \mathbf{N}. \tag{3}$$

Through this process, the machine generated label is obtained and then the framework will go through to update the similarity. As described above, not only *majority voting* can be applied to select a machine generated label, many other more accurate approaches can be used. Other approaches can also work applicable to the proposed framework.

4.1.2 Profile Regeneration and Similarity Update. As described in Section 3.2, the distribution of user i is defined by a vector L_i. When user i posts a new image, the image will be assigned a machine generated label and the distribution of user i needs to be updated. This step is referred to the step 3 in Fig. 3(a).

Users who have the corresponding images will acquire the corresponding machine generated labels as shown in Fig. 4. Thus the corresponding user vector L_i will be updated by adding a number of labels and normalization in the following way:

$$L_i = L_i' + \Delta L_i, \tag{4}$$



profile L'_i
ΔL_i
profile $L_i = L'_i + \Delta L_i$
$l_{i,1}$ $l_{i,2}$ $l_{i,3}$ $l_{i,4}$
2 3 2 1
0 0 1 0
2 3 3 1

Fig. 4. Profile update with an incoming label

where ΔL_i is the incremental label vector, indicating the newly added labels. When the user vector is obtained, similarities between L_i and all other user vectors need to be updated as well. If the similarity is purely recalculating without considering previous information then the complexity will still maintain $O(N^{(u)}K)$, where $N^{(u)}$ is the number of users and K is the dimension of the user profile, the distribution of labels. For example, user i, who has a user profile, L'_i, has posted an image and the assignment is label k. So the number of label k of user i has, $l_{i,k}$, is $l_{i,k} = l'_{i,k} + 1$. By storing the $L2$-norm of each distribution and previous similarity, the updated cosine similarity, $S(L_i, L_j)$, can be re-calculated as follows:

$$\begin{aligned}
S(L_i, L_j) &= \frac{L_i \cdot L_j}{\|L_i\|\|L_j\|} \\
&= \frac{l_{i,1}l_{j,1} + \cdots + (l'_{i,k}+1)l_{j,k} + \cdots + l_{i,d}l_{j,K}}{\sqrt{l_{i,1}^2 + \cdots + (l_{i,k}+1)^2 + \cdots + l_{i,K}^2}\|L_j\|} \\
&= \frac{l_{i,1}l_{j,1} + \cdots + l'_{i,k}l_{j,k} + \cdots + l_{i,K}l_{j,K} + l_{j,k}}{\sqrt{l_{i,1}^2 + \cdots + l_{i,k}^2 + \cdots + l_{i,K}^2 + 2l_{i,k} + 1}\|L_j\|} \\
&= \frac{L'_i \cdot L_j + l_{j,k}}{\sqrt{\|l'_i\|^2 + 2l_{i,k} + 1}\|L_j\|} \\
&= \frac{\left\|L'_i\right\| \left\|L_j\right\| S(L'_i, L_j) + l_{j,k}}{\left\|L^*_i\right\| \left\|L_j\right\|} \\
\text{where } & \|L^*_i\|^2 = \|L'_i\|^2 + 2l_{i,k} + 1
\end{aligned} \tag{5}$$

In Eq. 5, the computation can be done in a constant time by directly using the norm of label distribution and previous cosine similarity. $l_{i,K}$ and $l_{j,K}$ are the number of occurrence of label k for user i and j, respectively. Then the complexity of calculating the updated similarity with all other users reduces to only $O(N^{(u)})$ and no need care about how many dimensions the distribution has.

4.1.3 Efficient Connection Discovery. To discover connections for the user who has just posted the images, the framework will then 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 top-J connections. This, however, can be significantly improved in a more efficient way by having some elegant pruning for connection discovery. Because the offline has already generated the sorted lists, users that are impossible to be in top-$\mathcal{J}$ after updating the similarities can be located and excluded in the calculation. The framework first calculates the similarity between the user who triggered the request and the J users who are listed as top-$\mathcal{J}$ connections only. The pruning process happens while calculating the similarity. For example, in Fig. 5 example, $M = 3$ and the list is for

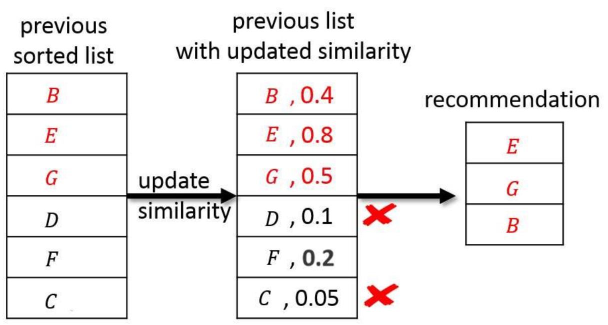


Fig. 5. Pruning in connection discovery for user A with $M = 3$ discovered connections. (L_C and L_D are not necessary for sorting)

user A with profile L_A. While calculating the similarity between L_A and other users, the framework starts calculating from the previously sorted list and maintains a minimum similarity value, 0.2 in this example, during calculating the first M similarities. Continuing to calculate others, L_D in this example, the framework will compare the updated similarity with the minimum value obtained. If that similarity is even smaller than the minimum value, then it's no need to sort the corresponding user L_D because its ranking must be over M. As all similarities are recomputed and the top M users are regenerated in offline computation, the same list of users would not be recommended forever. 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 one. If the following inequality is satisfied for user L_p, it is not necessary to do the heap search for L_p:

$$S(L_i, L_p) < \min_q S(L_i, L_q) \quad \forall q \in \mathbf{C}_i^{'}, p \notin \mathbf{C}_i^{'} \tag{6}$$

The time complexity of the pruning strategy is $O(N^{(u)})$. Through this pruning process, the efficiency of heap search to generate $\mathbf{C}_i$ will obviously be improved because the number of users needed in heap search decreases and the time complexity for heap search is about $O(MlogJ)$. It will be very helpful for real-world applications. In summary, the procedure of an efficient connection discovery framework for online can be described in Algorithm 1.

ALGORITHM 1: Online Connection Discovery

```
**Input:** The image feature vector $\vec{x}$ shared by user $i$
**Output:** Recommendations $\vec{C_i}$ for user $i$
$\Delta L_i \leftarrow \vec{0}$ ;
label($\vec{x}$) $\leftarrow$ voting(LSH($\vec{x}$) ;
$\Delta L_i \leftarrow \Delta L_i$+label($\vec{x}$) ;
$L_i \leftarrow L_i^{'} + \Delta L_i$ ;
**for** $L_j \in \mathbf{N}^{(u)}$ **do**
    $S(L_i, L_j) \leftarrow$ updateSim($S(L_i^{'}, L_j)$) ;
**end**
$\vec{C_i}$ = topJConnections($L_j$) ;
```

4.2 Offline Computation

As shown in Fig. 3(b), computationally intensive tasks are performed periodically in the offline cloud platform, Spark[1] is used in this paper. As offline computation can have a relaxed time requirement in the whole framework, the LSH is constructed in offline and used by online connection discovery. Moreover, the offline computation 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 such that a sorted list is always available even there is no image input. This section will present the detail of the offline implementation on Spark.

4.2.1 Clustering on Cloud Platform. This section presents a distributed framework of the clustering algorithm, k-means, on Spark cluster. The Spark 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 refers to step 1 of Fig. 2. Each numeric vector represents an image. Originally the large file is split and stored in the Hadoop Distributed File System (HDFS) and then Spark will read it as a distributed object in a cluster. The k centroids are randomly generated from all the image vectors and it becomes a variable saved in memory. In Spark, this variable is broadcasted to all the slave nodes so that each slave can help calculated the distance between centroids and all the points. The first process is a *map* transformation in Spark as shown in step 1 in Fig. 3(b). 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 assignment for all vectors are done, the *reduce* step as in step 2 of Fig. 3(b) can be performed to calculate the new centroid for each cluster by combining the points within the same cluster. As Spark allows developers to store the results in memory, it is very useful for iterative algorithms like k-means in this paper. Thus the clustering will iterate until the centroids converge. Once the clustering result is obtained, the offline computation will send the result to online and continue distribution calculation as shown in step 3 of Fig. 3(b).

4.2.2 Generating the Sorted 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. 3(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 machine generated labels will be assigned to the users who have the corresponding images. With these machine generated labels, the framework can easily construct the user distributions in a vector format by counting the occurrences of each label.

From Section 3.2, the similarity between two users L_i and L_j is updated periodically based the cosine similarity in eq. 2. The sorted list for each user L can then be generated descending sorting according to the similarity between L and all other users. In this way, the framework can drag the discovered connections out at any time and no need to wait images come in.

4.3 Communications between Online and Offline

This section describes how the online and offline computations are connected and how they facilitate an efficient connection discovery. As shown in Fig. 3, 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

[1]Spark: https://spark.apache.org/.

offline 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 process from step 1 to step 5 of Fig. 3(b).

While the offline computation is generating the sorted lists for all users, the clustering results will be sent to the online (step 3 of Fig. 3(b)) for voting once step 2 of Fig. 3(b) is done and the offline computation will still continue distribution calculation without pausing. 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. The clustering results or in other words the labeling results of images are used in voting the label for the incoming images (step 2 of Fig. 3(a)). Because each image has its corresponding label in offline, the LSH select the neighbors for voting and the neighbors also have the corresponding labels from the clustering results. Once the offline computation is done, the sorted lists will also synchronize for all users in online so that the framework is also available for the first kind of request describe in Section 4.1.

5 PERFORMANCE EVALUATION

With the discovered connections, relationships between users can be predicted and many applications such as gender prediction, collaborative recommendation, user-targeted advertising and user search can be applied on. In this section, the discovered connections are evaluated with the application of follower/followee recommendation on social networks. This section first presents the setup for the experiments and then introduces the two social network datasets used in the experiments. Some methods are implemented as the baseline to compare the performance of the proposed framework by four metrics defined in this section. Finally, the evaluation results are presented to shows the efficiency of the proposed framework.

5.1 Experimental Setup

There are two datasets with a large amount of user-shared images collected from Skyrock and Twitter involved in the experiments. Skyrock and Twitter are general social networks which allow users to post text and images, and even other emerging forms such as videos. The Skyrock dataset comprises 176,547 images uploaded by 722 unique users, where all the users are randomly selected from a single query page of an image keyword search and their corresponding images and follower/followee relationships are scraped. There are 2,439,058 follower/followee relationships in total, including 7348 existing follower/followee relationships within these 722 users. The second dataset, Twitter, consists of 150,696 images uploaded by 462 unique users, scraped with the same method as Skyrock dataset. The total follower/followee relationships involved are 14,487,045, including 1364 existing follower/followee relationships within these 462 users. The statistic distribution in terms of the number of friends and the number of shared images for both datasets are shown in Fig. 6 and Fig. 7. All users from Skyrock and Twitter are selected randomly using the official API, in which the network densities are only 1.41% and 0.65% on Skyrock and Twitter, respectively. The datasets with 300K+ images from two social networks are large enough to show such behavior for evaluation of connections discovered and its application in follower/followee recommendation. Other possible results such as gender identification using similar dataset such as Flickr are shown in [6]. The results can be well extended when the analytics are applied in real-world platforms where billions of users are involved.

The experiments are performed in the Spark cluster, which consists of 9 VMs, one as the master node and another 8 VMs as slave nodes. Each node has two virtual CPU running at 2.4GHz, with a 8 GB RAM, a 20GB hard disk and a Gigabit network interface card. The cluster was built on top of

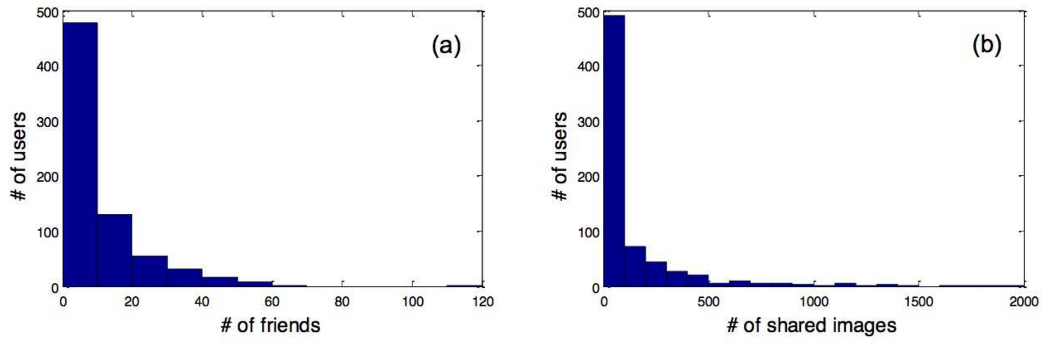


Fig. 6. Statistic distribution of Skyrock dataset. (a): number of friends. (b): the number of shared images

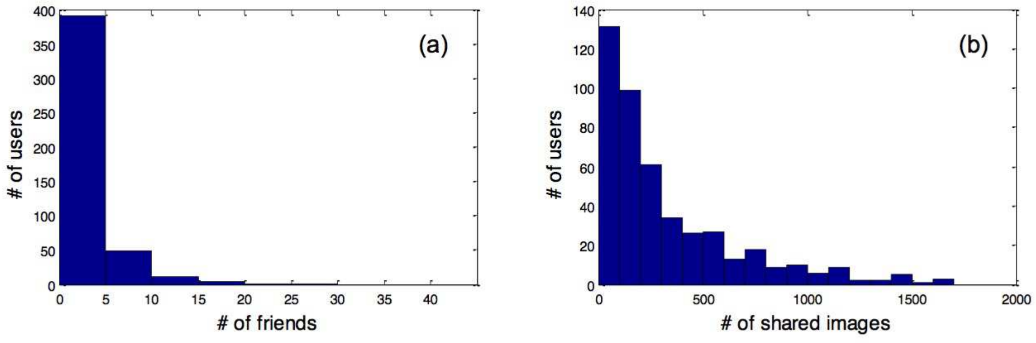


Fig. 7. Statistic distribution of Twitter dataset. (a): number of friends. (b): the number of shared images

Hadoop and the dataset is stored in Hadoop Distributed File System (HDFS). In order to evaluate the framework, there are some existing frameworks that for comparisons:

- **Stand-alone BoFT (SABoFT)**: The original BoFT connection discovery framework shown in Fig. 2 is implemented on a stand-alone machine to be compared with the proposed framework. SABoFT runs the complete process on a single machine and did not distribute any parts of computations in BoFT.
- **Cloud-Assisted BoFT (CABoFT)**: The cloud-assisted BoFT framework, proposed by [21], is also implemented for comparisons. The Cloud-assisted BoFT framework distributed the clustering process via Hadoop MapReduce. It further improved the scalability by distributing the distributions calculation and similarity calculation by parallel computing for connection discovery.

In SABoFT, when more images are uploaded, the runtime increases linearly as there are more images to cluster completely with limited computing resources such as CPU. In CABoFT, Although plenty of computing resources are available, the runtime is limited by the communication and those operations that can not be computed in parallel. These limitations increase the runtime. The proposed BoFT computation framework handles the computational parts (i.e. clustering, label distribution calculation, similarity calculation and top-M connection discovery) periodically in offline, which can make the framework persist the static information that can be sent back to

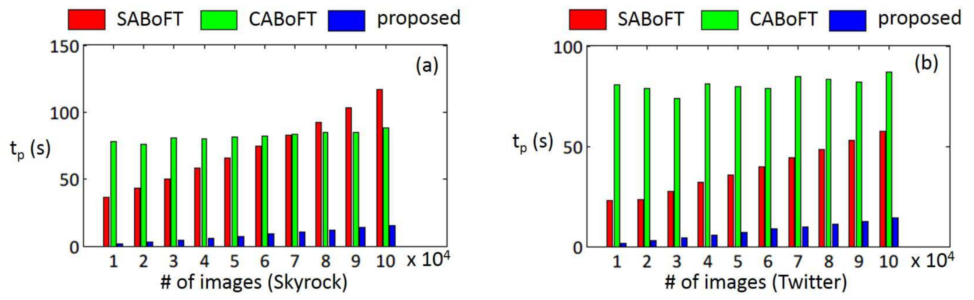


Fig. 8. Processing Time on two social networks. (a): Processing Time on Skyrock. (b): Processing Time on Twitter

users. When users trigger a request by posting images, the framework then quickly respond to a user through the online connection discovery. As shown in Fig. 3, the offline computation uses hash functions to achieve random projections, so there are some parameter settings in LSH before running the experiments. The implementation is using the following settings of parameters:

- N_h: the number of hash functions in LSH. Within each set of N_h dot products, the projection achieves success if the query and the nearest neighbors are in the same bin in all N_h dot products. N_h is empirically set to 10 [36] in this experiment.
- N_b: the number of buckets. By increasing the number of random projections, LSH can find the true nearest neighbor with arbitrarily high probability. N_t is set to 100 [36] in this experiment.
- β: the percentage number of all neighbors that to be chosen in majority voting (i.e. the number of neighbors that is selected from LSH). Here it is set to 50% and the experiment will measure the influence on the precision in discussions. Again, the neighbors selected are the top 50% nearest neighbors from all neighbors extracted in LSH.

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 an image vector is received to the time that recommendation is made as shown in Fig. 3. The second one is the precision for evaluating how accurate the recommendation are. Precision is defined as the number of truly predicted connections (T_p) divided by the total number of predicted connections ($T_p + F_p$):

$$Precision = \frac{T_p}{T_p + F_p}, \tag{7}$$

where F_p is the number of false predicted connections. There is another variable which is the number of neighbors in LSH in the proposed online connection discovery, as shown in Fig. 3. LSH will help pick all the similar neighbors for the voting process. But whether a large number of neighbors or a small number of neighbors would affect the performance. The last metric is to measure the processing time per image $t_{pi} = \frac{t_p}{n}$. As this paper claims that the framework towards an efficient connection discovery, the framework is expected to have a fast and stable processing time for each image input.

5.2.1 Processing Time. To process the Skyrock dataset, the proposed framework, SABoFT and CABoFT first trained initial 76,547 images in offline and then test the framework with 100,000 images. For Twitter, the framework also trained initial 50,696 images in offline first.On top of the

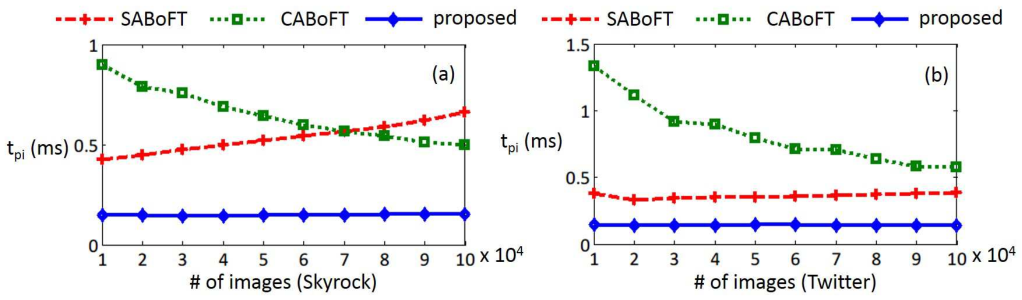


Fig. 9. Processing time per image on two social networks. (a): Processing time per image on Skyrock. (b): Processing time per image on Twitter

Table 1. Comparison of Processing Time on Skyrock

Time(s)	Label Generation	Profile Update	Similarity Update
SABoFT	102.534	0.258	3.880
CABoFT	80.402	0.324	3.763
Proposed	9.440	0.011	5.405

initial 76,547 and 50,696 images, the experiment examined the time as a function of the number of testing images from 10,000 to 100,000 with the increase of 10,000. Fig. 8 shows the processing time of online computation of BoFT in all the three frameworks introduced above. Fig. 8(a) and Fig. 8(b) show the processing time on Skyrock and Twitter dataset, respectively. It is observed that the proposed framework outperforms the stand-alone and cloud-assisted BoFT with increasing volume of images for both Skyrock and Twitter due to the low complexity of the online computation.

The processing time of all the three frameworks increases with more images. The processing time of SABoFT increases exponentially with more images in both social networks because of the low scalability of the stand-alone platform. In terms of the CABoFT, the processing time is originally very large since the volume of data is small and then the overhead in the cloud will dominate most of the time. However, as shown in Fig. 8 in Skyrock dataset, the time increased slowly when there are large amount of data and outperformed SABoFT after the number of images is over 70,000. The proposed framework reduces 90.0% and 88.37% as much time as SABoFT and CABoFT on average in terms of the processing time on all images, respectively. In Table 1 and 2, the comparison of the processing time of individual computation units for the three approaches are made. The major computation units include label generation, profile update and similarity update. The experiments are conducted for testing images of 100,000. As it can be seen that, for both Skyrock and Twitter datasets, the largest benefit of the proposed framework is in label generation. The proposed framework generates labels for incoming images through LSH and voting scheme without the necessity of clustering from scratch, which reduces a significant amount of computation time compared to SABoFT and CABoFT. For the profile update, the proposed framework also speed up a lot due to incrementally update. Although it might seem that the similarity update costs more time for the proposed framework, it’s actually because the similarity update is performed in a streaming manner where the recommendation can be made directly after an incoming image is processed. Whereas for SABoFT and CABoFT, the recommendation is made after all 100,000 images are processed. Therefore, the similarity update actually speeds up for the proposed framework.

Table 2. Comparison of Processing Time on Twitter

Time(s)	Label Generation	Profile Update	Similarity Update
SABoFT	50.176	0.393	2.141
CABoFT	80.56	0.337	1.918
Proposed	9.069	0.009	3.909

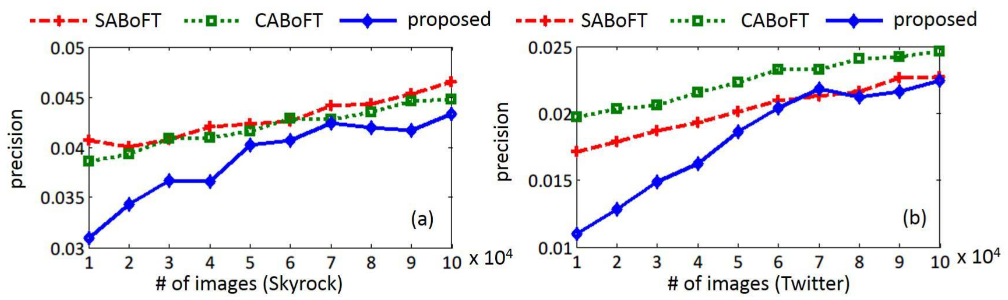


Fig. 10. Precision on two social networks. (a): Precision on Skyrock. (b): Precision on Twitter

5.2.2 Processing Time Per Image. As described in the beginning, t_{pi} is the processing time divided by the number of images. Fig. 9 shows the average processing time per image of three frameworks. The average processing time measures the time from when an image feature vector enters into the framework to when connections are found. The following description in this section will simply use the average processing time to denote processing time per image. On both social networks, the average processing time of SABoFT is shorter than the one of CABoFT in the beginning because the overhead in cloud dominates most of the time in CABoFT. With more data comes in, the average processing time drops down quickly and becomes more stable because processing data comes to the major part who consumes the time and it costs not too much time to process an image. With the high scalability of the cloud, the average processing time becomes stable. As shown in Fig. 9, the average processing time of CABoFT is shorter than SABoFT after the number of images surpasses 70,000, which also indicates the higher scalability of CABoFT than SABoFT. Because the waiting time for each image is increasing with more data in a stand-alone platform, the average processing time per image will also increase. Although the average processing time of CABoFT always higher than SABoFT in the Twitter dataset as shown in Fig. 9, the trend is still obvious that the time of CABoFT drops and becomes stable and the time of SABoFT increases with more and more data. Because of the low scalability of SABoFT, the average processing time should finally longer than the time of CABoFT.

With more and more streaming input, the time to process each image is almost equal as shown in Fig. 9 (a) and Fig. 9 (b). The average processing time per image of the proposed framework is always shorter than SABoFT and CABoFT, and those two methods have unstable performance when the number of images is different. 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 have a satisfied user experience and attract more users to join [1, 19]. With the stable and low processing time per image, the framework can easily scale and handle a large amount of data in an efficient fashion.

5.2.3 Precision. This experiment evaluates the precision of recommended relationships compared with ground truth on both Skyrock and Twitter dataset. Same as the previous experiments, the proposed framework first trained 76,547 and 50,696 images in advance and then test the remaining 100,000 images for comparisons by increasing the images by 10000. If the number of inserted images now is 30,000, then SABoFT and CABoFT will cluster these 30,000 images in addition to the initial 76,547 and 50,696 images, with the total of 106,547 and 80,696 images, and recommend relationships for the newly 10,000 images which is just posted by users, as in Fig. 10. Thus the size of test data at each time is 10,000 images posted by users. The experiments imitate the real-world applications, in which the offline computation of the proposed framework will update periodically. Basically, there is not much difference in the performance between SABoFT and CABoFT because they both applied the same clustering technique, k-means. The only differences are CABoFT make it scalable on cloud and the initial and random centers picked in the k-means algorithm. So they have very similar performance and the precision increases as more and more data came in to make the clustering more reliable. With the increasing number of posted images, the advantages of the proposed framework show up. It achieves 96.8% and 91.2% of the precision in Skyrock and Twitter, respectively. The proposed framework can have a comparable precision with stand-alone and cloud-assisted BoFT frameworks while it can have a much faster speed than them, while achieving a similar precision. The performance achieved in the two datasets can be extended when applying to real-world platforms.

It is proven that the proposed framework can achieve a comparable precision with the original BoFT while saving much more time than the stand-alone and cloud-assisted mode of BoFT. The online connection discovery can be done much faster since it is not necessary to wait for the clustering result, which can be updated in offline. Furthermore, the efficiency of processing time would not be influenced by whatever clustering the framework used because the online computation only utilizes the result of the offline clustering. Thus the offline clustering can be improved without considering whether the processing time will be affected or not. By using this architecture, developers can separately improve the precision on online and offline computations, which makes it more independent.

6 DISCUSSIONS

This section discusses the influence of the number of neighbors on the proposed framework and the trade-off value between the precision and the processing time, and then raises some directions that worth for future works.

6.1 Influence of the number of neighbors

As described in Section 4.1.1, LSH will help find out all the points which are near to the input. Because of the concern on the efficiency, the framework will select β nearest neighbors to avoid the fluctuation in the processing time. However, vary numbers of neighbors generated from LSH can have different impact on the framework. This experiment measures the influence of different percentages of neighbors extracted from LSH on precision and processing time, as shown in Fig. 11. The percentage β starts from 10% to 100% and the precision reached the peak at $\beta = 50\%$ in Skyrock and Twitter. From Fig. 11 (a) and Fig. 11 (b), the number of neighbors does not have a monotone-increasing effect on precision. With the fixed value of N_h and N_b, the range of extracted neighbors is also fixed. The results as shown in Fig. 11 (a) and Fig. 11 (b) indicate the range is still large because extracting all neighbors does not lead to a high precision but only a percentage of all neighbors. Based on this reason, either a very small number of neighbors or a very large number of neighbors in this experiment cannot lead to a very good result. Intuitively, it is not reliable to judge which label that the incoming image belongs to when the number of neighbors is too small.

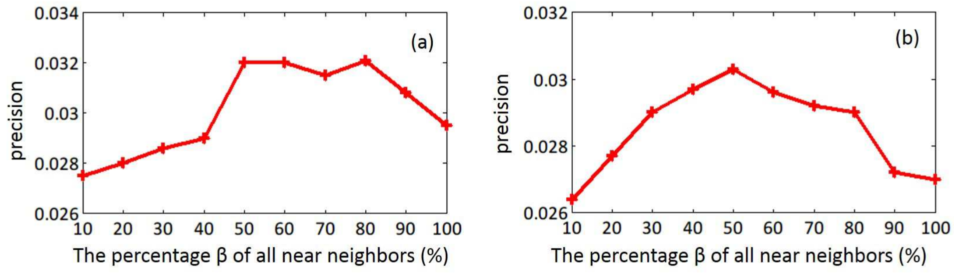


Fig. 11. Precision on two social networks. (a): Influence of different percentage of neighbors on Skyrock. (b): Influence of different percentage of neighbors on Twitter.

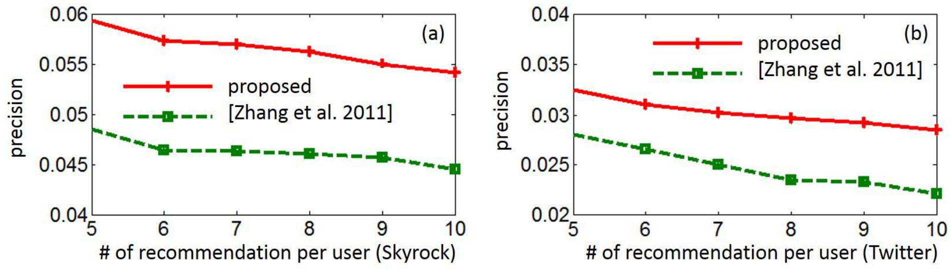


Fig. 12. Results of follower/followee recommendation on: (a) Skyrock, (b) Twitter.

In a similar way, it is also not reliable to decide the label when the number of neighbors is too large because a wide range can lead to the wrong classification. However, the different dataset will have a different optimal β value for connection discovery in terms of the precision. Thus it is still challenging to calculate an optimal β value for the whole process.

6.2 Comparing another framework

Although the focus of this paper is to demonstrate how a real-time response is possible using machine generated labels for connection discovery, it is also interesting to compare other approaches using user shared images for follower/followee recommendation. The approach in [46] is implemented, in which the user profile of users is not the distribution of the occurrence of machine generated labels, but the mean vector of feature vectors obtained from images. Fig. 12 shows the comparisons on two datasets with the number of recommendations to each user is 5 to 10. It is observed that the proposed approach is 20% better than the approach in [46].

6.3 Possible Research Directions

There are directions to improve the framework in terms of the precision. The first one is the burn-in period, where users share only a few images that the results is not reliable. One of the options is to predict the connections based on conventional approaches, such as using user online profile. The second one is the quality of the clustering results, a better clustering results, such a the best K, can enhance connection discovery. In online connection discovery, new clusters may be needed when an image comes. It is worth investigating whether forming new clusters in online can help to further improve the precision. The last issue is about the number of online servers in the framework, one in this experiments. In real-world applications, there can be many client servers to receive the images from users, and increasing the number of servers can help to speed up the processing. It is interesting to investigate how the number of servers affects the processing, as well as how to design an effective algorithm for the processing on multiple machines.

7 CONCLUSION

In this paper, an efficient computation framework is proposed for connection discovery to speed up the BoFT analytics using user shared images. The framework is designed separately by online and offline computations. The computationally intensive parts are handled in offline on the cloud platform and the online computation utilizes the results from offline to make connection discovery in an efficient fashion. The online connection discovery then uses voting to select the results from offline. The performance of the connection discovery is evaluated in the application of follower/followee recommendation with 300K+ user-shared images on two social networks. The experiments showed the proposed framework has a much shorter processing time compared to the previous works of BoFT frameworks on two real social network datasets. The proposed framework reduces on average 90.0% as much time as stand-alone BoFT and 88.7% of the time on cloud-assisted BoFT in terms of the processing time on all images, with 90% as accurate as other frameworks. The experiments further proved the effectiveness of the proposed framework with more stable and much shorter processing time per image, which makes it satisfy the requirement of real time in real-world applications.

ACKNOWLEDGMENTS

This work is supported by HKUST-NIE Social Media Lab., HKUST. The authors would like to thanks ZHANMING JIE for improving this paper.

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] Xavie Amatriain and Justin Basilico. 2013. System architectures for personalization and recommendation. *the Netflix Techblog: http://techblog. netflix. com/2013/03/system-architectures-for. html* (2013).

[4] Jeremy Buhler. 2001. Efficient large-scale sequence comparison by locality-sensitive hashing. *Bioinformatics* 17, 5 (2001), 419–428.

[5] 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.

[6] Ming Cheung, James She, and Zhanming Jie. 2015. Connection Discovery using Big Data of User Shared Images in Social Media. *IEEE Transactions on Multimedia* (2015). Accepted.

[7] Ming Cheung, James She, and Li Xiaopeng. 2015. Non-user Generated Annotation on User Shared Images for Connection Discovery. In *Proceedings of The IEEE International Conference on Cyber, Physical and Social Computing (CPSCom 15)*.

[8] Edith Cohen, Mayur Datar, Shinji Fujiwara, Aristides Gionis, Piotr Indyk, Rajeev Motwani, Jeffrey D Ullman, and Cheng Yang. 2001. Finding interesting associations without support pruning. *Knowledge and Data Engineering, IEEE Transactions on* 13, 1 (2001), 64–78.

[9] Corinna Cortes and Vladimir Vapnik. 1995. Support-vector networks. *Machine learning* 20, 3 (1995), 273–297.

[10] Weijia Dai, Ginger Z Jin, Jungmin Lee, and Michael Luca. 2012. *Optimal aggregation of consumer ratings: an application to yelp. com.* Technical Report. National Bureau of Economic Research.

[11] Abhinandan S Das, Mayur Datar, Ashutosh Garg, and Shyam Rajaram. 2007. Google news personalization: scalable online collaborative filtering. In *Proceedings of the 16th international conference on World Wide Web*. ACM, 271–280.

[12] Claudia Diamantini, Domenico Potena, Alessandro Sabelli, and Samuele Scattolini. 2014. An integrated system for social information discovery. In *International Conference on Collaboration Technologies and Systems (CTS), 2014*. IEEE,

353–360.

[13] 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.

[14] 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.

[15] Aristides Gionis, Piotr Indyk, Rajeev Motwani, et al. 1999. Similarity search in high dimensions via hashing. In *VLDB*, Vol. 99. 518–529.

[16] 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.

[17] 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.

[18] Piotr Indyk and Rajeev Motwani. 1998. Approximate nearest neighbors: towards removing the curse of dimensionality. In *Proceedings of the thirtieth annual ACM symposium on Theory of computing*. ACM, 604–613.

[19] 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.

[20] Shuhui Jiang, Xueming Qian, Jialie Shen, Yun Fu, and Tao Mei. 2015. Author topic model-based collaborative filtering for personalized POI recommendations. *IEEE transactions on multimedia* 17, 6 (2015), 907–918.

[21] Zhanming Jie, Ming Cheung, and James She. 2015. A Cloud-assisted Framework for Bag-of-Features Tagging in Social Networks. In *IEEE 4th Symposium on Network Cloud Computing and Applications*. Accepted.

[22] 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.

[23] I Kanellopoulos and GG Wilkinson. 1997. Strategies and best practice for neural network image classification. *International Journal of Remote Sensing* 18, 4 (1997), 711–725.

[24] Xiaojiang Lei, Xueming Qian, and Guoshuai Zhao. 2016. Rating prediction based on social sentiment from textual reviews. *IEEE Transactions on Multimedia* 18, 9 (2016), 1910–1921.

[25] Ian XY Leung, Pan Hui, Pietro Lio, and Jon Crowcroft. 2009. Towards real-time community detection in large networks. *Physical Review E* 79, 6 (2009), 066107.

[26] Zhenyu Li, Jiali Lin, Kave Salamatian, and Gaogang Xie. 2013. Social connections in user-generated content video systems: Analysis and recommendation. *IEEE Transactions on network and service management* 10, 1 (2013), 70–83.

[27] Andy Liaw and Matthew Wiener. 2002. Classification and regression by randomForest. *R news* 2, 3 (2002), 18–22.

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

[29] 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.

[30] Xiuquan Qiao, Jianchong Su, Jinsong Zhang, Wangli Xu, Budan Wu, Sida Xue, and Junliang Chen. 2014. Recommending friends instantly in location-based mobile social networks. *Communications, China* 11, 2 (2014), 109–127.

[31] Lara Quijano-Sanchez, Juan A Recio-Garcia, and Belen Diaz-Agudo. 2011. Happymovie: A facebook application for recommending movies to groups. In *Tools with Artificial Intelligence (ICTAI), 2011 23rd IEEE International Conference on*. IEEE, 239–244.

[32] 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.

[33] Jitao Sang and Changsheng Xu. 2013. Social influence analysis and application on multimedia sharing websites. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)* 9, 1s (2013), 53.

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

[35] Meredith M Skeels and Jonathan Grudin. 2009. When social networks cross boundaries: a case study of workplace use of facebook and linkedin. In *Proceedings of the ACM 2009 international conference on Supporting group work*. ACM, 95–104.

[36] Malcolm Slaney and Michael Casey. 2008. Locality-sensitive hashing for finding nearest neighbors [lecture notes]. *Signal Processing Magazine, IEEE* 25, 2 (2008), 128–131.

[37] 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.

[38] DT Tsai, Allen Y Chang, S Chung, and You Sheng Li. 2010. A Proxybased Real-time Protection Mechanism for Social Networking Sites. *Proc. ICCST* (2010).

[39] Stanley Wasserman. 1994. *Social network analysis: Methods and applications*. Vol. 8. Cambridge university press.

[40] Zhipeng Wu, Shuqiang Jiang, and Qingming Huang. 2009. Friend recommendation according to appearances on photos. In *Proceedings of the 17th ACM international conference on Multimedia*. ACM, 987–988.

[41] 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.

[42] 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.

[43] Xiwang Yang, Yang Guo, and Yong Liu. 2013. Bayesian-inference-based recommendation in online social networks. *IEEE Transactions on Parallel and Distributed Systems* 24, 4 (2013), 642–651.

[44] Ting Yao, Chong-Wah Ngo, and Tao Mei. 2011. Context-based friend suggestion in online photo-sharing community. In *Proceedings of the 19th ACM international conference on Multimedia*. ACM, 945–948.

[45] Guoshuai Zhao, Xueming Qian, and Xing Xie. 2016. User-service rating prediction by exploring social users' rating behaviors. *IEEE Transactions on Multimedia* 18, 3 (2016), 496–506.

[46] 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