Multimed Tools Appl (2017) 76:23291–23307
DOI 10.1007/s11042-016-4060-4

MapReduce-based clustering for near-duplicate image identification

Wanqing Zhao[1] · Hangzai Luo[1] · Jinye Peng[1] · Jianping Fan[2]

Received: 24 March 2016 / Revised: 29 September 2016 / Accepted: 10 October 2016/
Published online: 16 November 2016

Abstract In this paper, an effective algorithm is developed for tackling the problem of near-duplicate image identification from large-scale image sets, where the LLC (locality-constrained linear coding) method is seamlessly integrated with the maxIDF cut model to achieve more discriminative representations of images. By incorporating MapReduce framework for image clustering and pairwise merging, the near duplicates of images can be identified effectively from large-scale image sets. An intuitive strategy is also introduced to guide the process for parameter selection. Our experimental results on large-scale image sets have revealed that our algorithm can achieve significant improvement on both the accuracy rates and the computation efficiency as compared with other baseline methods.

Keywords Near-duplicate identification · Image clustering · Representative image · MapReduce · Large-scale photos

1 Introduction

The prevalence of digital cameras and the emerging trends of image sharing on the Internet lead to explosive growth of online images. As a result, many images on the Internet are near

✉ Hangzai Luo
hzluo@nwu.edu.cn

Wanqing Zhao
zhaowanqing@stumail.nwu.edu.cn

Jinye Peng
pjy@nwu.edu.cn

Jianping Fan
jfan@uncc.edu

[1] School of Information and Technology, Northwest University of China, Xi'an, China

[2] Department of Computer Science, UNC-Charlotte, Charlotte, NC 28223, USA

duplicates for the same object or scene. For many applications, such as personal photo management, image retrieval [6], image popularity ranking [12] and semantic extraction [18, 29], those near duplicate images must be clustered together to enable semantic computation, as shown in Fig. 1. Therefore, near-duplicate image clustering is a very important task for most image management systems.

Without considering the sizes of image sets, the near-duplicate image clustering problem can be treated as a restricted CBIR problem, and it is not very difficult to support identification of near-duplicate images from some small scale image sets. However, when the image scale increases to a very large level, several serious problems may emerge. First, the computation complexity for most clustering algorithms is $O\left(n^2\right)$ or even more, thus they can not directly be applying for identifying near-duplicate images from large-scale image sets. Second, most clustering algorithms are largely data dependent, which may not able to support parallel computing. Third, the features used in traditional CBIR algorithms are generally in high dimensions, which may result in high computational complexity.

Based on these observations, an effective algorithm is developed in this paper to identify the near-duplicate images from large-scale image sets. First, the locality-constrained linear coding (LLC) [25] is seamlessly integrated with the maxIDF cut model to achieve more discriminative representations of images. Second, a MapReduce-based clustering algorithm is developed to reduce the computational cost and identify the near-duplicate images effectively, which can achieve low complexity on image clustering but also scale with the sizes of image sets effectively.

This paper is organized as follows: the related works on near-duplicate images clustering and representative image selection are discussed in Section 2. In Section 3, we introduce our algorithm for parallel image clustering for near-duplicate image identification. The experimental results for algorithm evaluation are reported in Section 4, and we have compared our algorithm with multiple baseline methods. Finally we conclude our paper in Section 5.

2 Related works

Large scale near-duplicate image detection (NDID) generally contains two sub tasks: (a) selecting a suitable function for similarity characterization; and (b) performing image clustering. Some works [6, 9] focused on one of these two tasks while some [15, 22] closely-relevant algorithms to improve overall performance.

Many researchers focused on improving the similarity measurement of NDID. Chum et al. [6] proposed a method by using "bag of words" model for the local feature descriptors (SIFT) and exploiting the enhanced min-hash techniques. MinHash [2, 6] uses Jaccard coefficient as the similarity measurement,which also is a LSH scheme. It assigns the image pairs into the same hashing bucket with the probability proportional to the overlapping between their image features. In practice, MinHash can be used as a probabilistic clustering method that assigns two images into the same bucket with the probability $S\left(I_i, I_j\right) = \left|\frac{W_i \cap W_j}{W_i \cup W_j}\right|$ that is equal to the overlap similarity of their image features. In order to obtain more similar hashcode with the original image features, the learning-based hashing schemes have been received more attentions in large scale image retrieval and other related applications. The learning-based hashing schemes target on learning a compact and similarity preserving representation such that the similar images can be mapped into the nearby binary hash codes in the Hamming space. For example, Spectral Hashing [27] constructs the global graph with L2 distance and optimizes the graph Laplacian cost function in the Hamming space. Compact

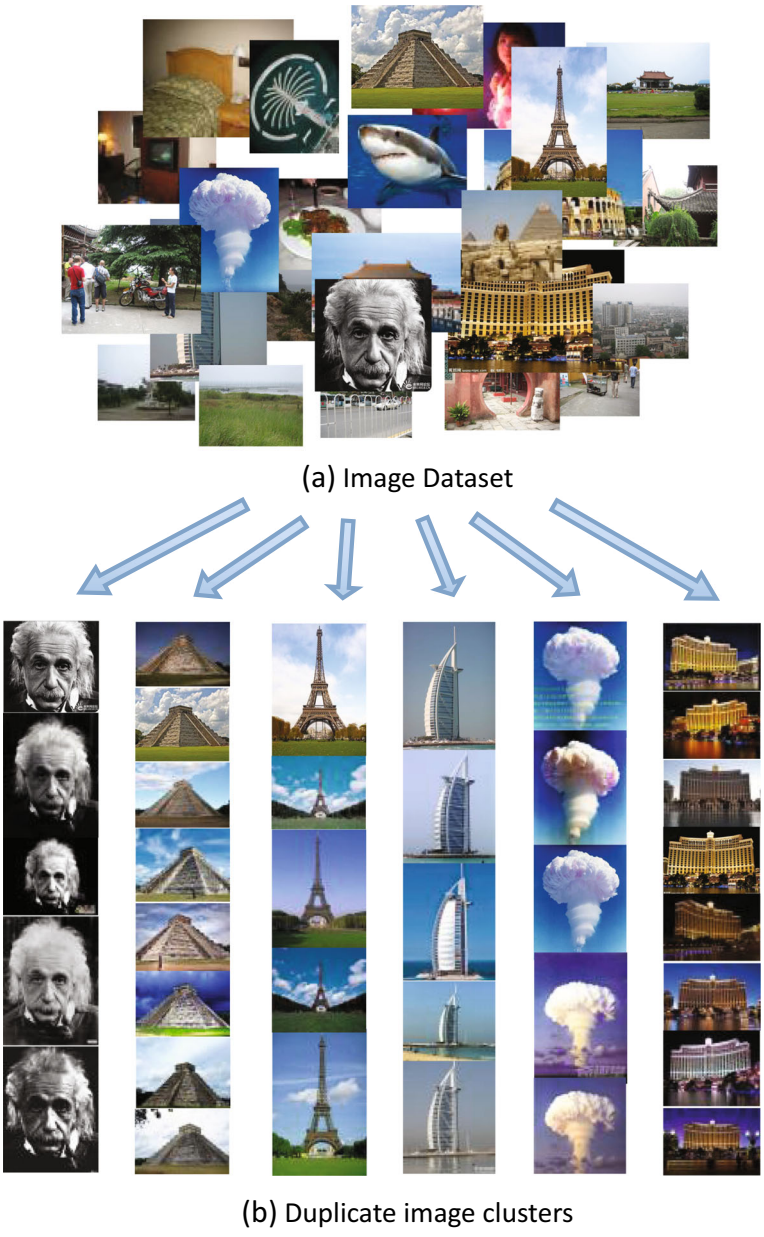

Fig. 1 The problem of large-scale image clustering is to identify a large number of clusters of near-duplicate images

sparse coding, the extension of the early work of robust sparse coding [4], adopts sparse codes to represent the database items: the atom indices, which correspond to nonzero codes, are used to build the inverted index, and the nonzero coefficients are used to reconstruct the database items and compute the approximate distances between the query and the database

items. Dong et al. [9] has shown that using entropy-based filtering to eliminate ambiguous SIFT features can retain high-quality features for supporting NDID, and they use graph cut for query expansion.

Although these approaches have shown that they are feasible on similarity measurement of NDID, it is still challenging to directly apply them to solve our problem on large-scale image clustering for near-duplicate detection. First, false positive is a serious problem for large-scale image clustering because encoding the original features into binary codes is not an easy issue. Second, the learning-based hash methods need multiple iterations for training the models, which may take a lot of time for a large image set. Third, these methods are all designed to run on a single machine. For the problem of large scale image clustering, large numbers of images cannot fit on one single machine and we cannot simply apply the traditional algorithms to support NDID.

Some researchers focused on improving the clustering task for NDID. Liu et al. [17] introduce an approximate nearest neighbor search algorithm by using spill trees for tackling the clustering problem. [11] has proposed a near-duplicate clustering method for near-duplicate text document clustering [3]. In more recent work, Wang and Zhang [26] have developed an interesting algorithm to combine both the local features and the global features for discovering the duplicate image clusters, where the global descriptors are used to discover the seed clusters and the local descriptors are used to grow the seeds to obtain good recall.Unfortunately, this algorithm also has some limitations inherently by global features.

As the tasks for similarity measurement and clustering are closely relevant for NDID, many researchers also optimize similarity measurement and clustering together to improve the overall performance. Hsieh et al. [14] proposed a two-stage sparse graph construction method. The first stage assigns the images into a set of overlapping groups by using Min-Hash method, and the second stage computes the pairwise similarities. Unfortunately, the false positives may seriously decrease the accuracy rates, meanwhile, the computation complexity for seed growing is still high for large-scale image sets. Xie and Tian [28] calculated the common features from every image pair and using the affinity propagation for image clustering, but parallelization is not supported which may result in low scalability.

Parallel Computing Framework: The MapReduce [8] can provide a stable and efficient parallel computing framework which also hiding many necessary details for handling a large number of machines. An operation in the MapReduce framework accepts an input of a collection of entries in the form of <key, value> pairs, and produces a collection of outputs in the same format. The operation of MapReduce contains two steps: the map step outputs <key, value> pairs after processing the input data and the reduce step collects and processes the <key, value> pairs that come from the map step with the same key. Based on the previous work to expand all of the similarity search [1], Elsayed et al. [10] used MapReduce to compute pairwise document similarity in large data collections and two MapReduce tasks are employed. The first task parsed the documents markup and generates the inverted lists. In the the second task, partial pairwise similarities for the pairs of the documents were calculated and those partial similarities for each pair to obtain final pairwise document similarity. Liu et al. introduced a MapReduce version of an approximate nearest neighbor search algorithms in [17] to explore the problem of image clustering. Hsieh et al. [13] proposed a MapReduce-based image graph construction method, and the image graph is used to improve the search result. Wang et al. [23] proposed a GPU-based MapReduce framework for near-duplicate video retrieval. However, those algorithms cannot be used to improve the effectiveness of duplicate discovery. In addition, the approach of [23] used bag of feature

and K-Means for clustering directly and the [17] tends to connect all of the points in the graph, which all may lead to lower efficiency.

3 MapReduce-based clustering for identifying near-duplicate images

In this section, a parallel system is developed for Near-duplicate images clustering. The images are represented as high dimensional sparse feature vectors by integrating the locality-constrained linear coding (LLC) with the maxIDF-cut strategy. The sparse features are used to divide the image set into a set of overlapping buckets and the near-duplicate images in the same bucket have the same significant feature atom. Then we check near duplicates in every bucket and merge the similar image pairs to generate the structure of the duplicate clusters. To make the entire process efficient for handling large-scale image sets, we use the MapReduce for parallel computation and HDFS for storage. We finally conduct our comparative experiments on the Yahoo! Webscope Yahoo Flickr Creative Commons 100M (YFCC-100M) dataset [21] and our labeled set. For the efficiency of generating visual vocabulary, we extracted a subset from the whole database as training set for K-means algorithm. Figure 2 shows the flowchart of our method. The details of each part are illustrated as follows.

3.1 Image representation with LLC and MaxIDF-cut

Given N image features $\{x_1, x_2, ..., x_N\}$ with $x_i \in \Re^d$, which forms the columns of the data matrix $X \in \Re^{d\times n}$. Let $V_1 ... V_M$ be a codebook learned by using k-means form X. LLC projects each image feature x_i down to the local linear subspace spanned by the $k (k \ll m)$ words from codebook closest to x_i. Specifically, the LLC uses the following criteria:

$$\begin{aligned} &\max_{S} \sum_{i=1}^{N} \|x_i - Bs_i\| + \lambda \|d_i \otimes s_i\|^2 \\ &s.t. 1^\top s_i = 1, \forall i \end{aligned} \tag{1}$$

where $\otimes$ denotes the element-wise multiplication, and d_i is the Euclidean distance between x_i and each word in the codebook. Note that the LLC in (1) is not sparse in the sense of l^0 norm, but it is sparse in the sense that the solution only has few significant values. Specifically, Let $u_i ... u_K$ be the indices of the K codewords closer to x_i (e.g., K-NN search in Euclidean distance) and denote them collectively as $B = [V_{u1} ... V_{uK}]$. Let $s_i \in \Re^k$ be the coefficients of the approximation $x_i \approx Bs_i$. LLC encoding of the descriptor x_i is the M-dimensional vector $f_{LLC}(x_i)$ of all zeros except for the k components $[f_{LLC}(x_i)]_{coding} = s_i$. In other words, the parameter k affects the degree of sparse code.

Nevertheless, because the feature distribution of the whole dataset is not considered, the reconstruction by LLC may lead to an image representation that is not discriminative enough for near-duplicate images clustering.

In this work, an alternative maxIDF-cut strategy is developed for LLC. The inverse document frequency(IDF) captures the informativeness of the words that appear in many different images, which are less informative than those that appear rarely. In our LLC framework, we used max pooling combined with ℓ_2 normalization for the final image feature representation. In order to obtain an image feature space which is discrimination and easy to be partitioned,our method employs the maxIDF [30] to perform the estimation, which is

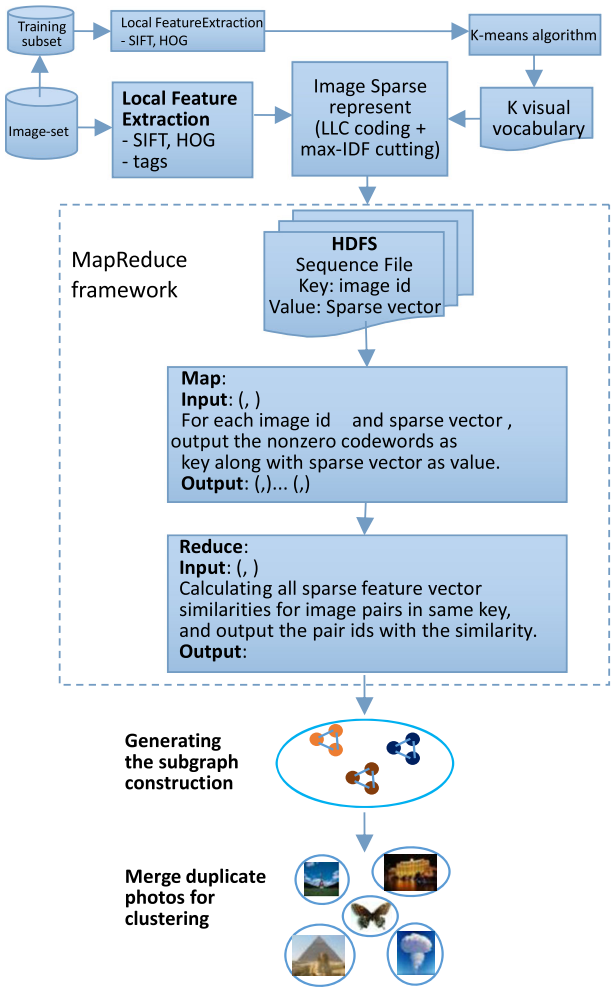

Fig. 2 The proposed near duplicate clustering system. We represent images by using the LLC and max-idf model. And we use the sparse features to divide the image set into overlapping subspaces. Then we check near duplicates in every subspace and merge the similar image pairs to generate the duplicate clusters

a variant of $\mathcal{L}_p$-norm. It can obtain a more accurate estimation about the frequency accumulates of each visual word when using the max pooling method for feature representation. The maxIDF weight of a word V_j is denoted as:

$$mIDF(V_j) = log \frac{N}{\max\limits_i s_i^j} \tag{2}$$

where N denotes the total number of images in the collection, and $\max\limits_i s_i^j$ encodes the max value of the database where word V_j occurs. Then the maxIDF-cut strategy recalculates

each LLC representation by using the maxIDF weight and encodes the small real-value into 0. Specifically, the maxIDF-cut uses the following criteria:

$$\begin{aligned} &maxIDF - cut(s_i) = \left\{ s_i^0 \cdot mIDF ... s_i^K \cdot mIDF \right\} \\ &if \left(s_i^j \cdot mIDF < t \cdot \sum_k s_i^k \cdot mIDF \right) \\ &s_i^j \cdot mIDF = 0 \end{aligned} \tag{3}$$

where mIDF mean the corresponding visual word frequency, which estimated by maxIDF. Parameter t determines how low the contribution of the word V_j in an image can be discarded (e.g., $t = 0.05$ in our experiment). The maxIDF cut strategy increases the discriminative terms and reduces the non-discriminative terms for sparse feature values. And it makes a more balanced distribution with more sparse values for image space partitioning and greatly increases the efficiency of similarity calculation. In our experiment, using maxIDF cut can achieve better performance for duplicate clustering.

3.2 Image space partitioning

A natural option to split a large-scale database into subspaces is hashing. Though the ultimate goal of most hash methods is to learn compact binary codes of representation vector whose Hamming distances correlate with the semantic similarities, so that search can be performed efficiency and lower memory expend [19, 24, 27]. Existing semantic hashing approaches require high computation cost for training the hash function and result in lower efficiency in large scale image sets. In this work, we represent the images as the low dimensional sparse feature vectors by using the LLC coding and the max-IDF cut model which has an approximated solution for fast encoding. In order to obtain higher accuracy, we do not convert the sparse data into compact Hamming space by any binarization rule. Since those features are very sparse and distinctive, we can simply extract prominent signatures for hash. The images, which have identical signatures, are assigned into the same bucket.

For tackling the problem of space partitioning for large-scale images, the MapReduce framework is applied. In the map function, the inputs are the sparse feature vectors of the images. For an image I_i, in order to improve the efficiency, only the non-zero item with form <item-id,item-value> is inputed. Then the map function sends a key/value pair in which the key is the codeword id and the value is the image id and the sparse feature value (I_i, s_i). Thus, in the reduce phase, the low dimensional sparse feature values of all the images, which have the same codeword, are aggregated to the same reduce function. The reduce function then calculates the feature vector similarity for every image pairs with the same key. And only the image id pair, which the similarity is higher than the cutting threshold, is identified as a duplicate pair.

We use LLC coding and maxIDF cut model to generate the image representation, thus the feature values are disperse and distinctive, thus the map function can simply partition the images with the same signature. The advantage of our technique lies in that the low cost for training the hash function, which is much lower than that of the existing hashing approaches and does not need (to build or use) any precomputed indexing structure, and it is efficient and feasible on handling large-scale datasets.

3.3 Computing pairwise similarity

Measure the distance between two images is calculated as the similarity of the local descriptor sets A_1 and A_2, which is defined as the ratio of the number of elements in the intersection over the union:

$$Sim_s(A_1, A_2) = \frac{|A_1 \cap A_2|}{|A_1 \cup A_2|} \tag{4}$$

Because our images are represented in high dimensional sparse features by using LLC and and max IDF pooling model, we find that using Histogram intersection as the similarity measure to calculate the nonzero elements can achieve better performance. Considering the similarity between the images I_i and I_j, the Histogram intersection is defined as:

$$Sim_h(I_i, I_j) = \frac{\sum_w \min\left(s_i^w, s_j^w\right)}{\sum_w \max\left(s_i^w, s_j^w\right)} \tag{5}$$

where v_i is the feature vector of the image I_i where each coordinate s_i^w is the number of visual words V_w present in the image I_i. And the previous definition of similarity in (4) is similar the definition in (5). The size of set intersection $|A_1 \cap A_2|$ is equal to $\sum_w \min\left(s_i^w, s_j^w\right)$ and the size of the $|A_1 \cup A_2|$ is equal to $\sum_w \max\left(s_i^w, s_j^w\right)$. Applying these equalities to sims (4), we can directly obtain (5). So we can get the similarity form (5) for the image pairs straightforward.

3.4 Pairwise merging for clustering

The above steps can generate similar image pairs with high accuracy, but the near duplicates pairs may have large variances and they may inevitably scatter across multiple buckets due to the lack of semantics in the function for similarity characterization and the process for image clustering, and our task is to discovery the near duplicate clusters form large scale image sets. Therefore, we perform clustering over the similar image pairs form all the buckets and eliminate the outliers to improve the recall rate.

A natural selection is to merge the pairs which have the same image id. By this method, we can get similar image clusters or graph construction with high recall and accuracy rates.

4 Algorithm evaluation

In this section, we present our quantitative evaluations of our method by using two image sets: (a) our own labeled image set which are collected from the web; and (b) the Flickr-100M image set.

4.1 Experimental setup

There are two main image sets for these experiments: (a) one contains the images which were hand labeled for setting various algorithm parameters and evaluating the accuracy

of the methods, and (b) the second larger set is used to evaluate efficiency.We used only a single descriptor for LLC, the Histogram of Oriented Gradient (HOG) , throughout the experiment. In our setup, the HOG features were extracted from patches densely located by every 8 pixels on the image, under three scales, 16*16, 25*25 and 31*31 respectively. The dimension of each HOG descriptor is 128.

(a) The labeled set: This set is an image set consisting of 20K images. It consists of 3000 near duplicate images in 30 categories and 17K non-duplicate images randomly sampled from Flickr-100M. This near duplicate set was produced by collecting the first 100 results for each query on several search engines base text-based image search queries and manual filtering.

(b) Flickr-100M: A large scale image set that consists of one hundred Million Flickr images. In order to test the system performance in different sizes, we continuously extract the distractors from Flickr-100M.

We tested the quality of image pairs that are discovered by counting the number of true pairs and false pairs. Recall is computed as the number of discovered true pairs divided by the total number of true pairs. Precision is computed as the percentage of true pairs among all retrieved pairs. F-measure is computed as the harmonic mean of precision and recall.

We implement MapReduce programming model for our parallel algorithm in this paper. Those experiments are run on two middle Hadoop clusters that consist of 10 and 20 nodes, respectively. Version 2.4.0 Hadoop is used on both clusters.

4.2 Parameter analysis

This section is aimed at finding a proper set of parameters for duplicate image clustering. For this, we adopt two intuitive tradeoff strategies to guide the process for parameter selection, one between precision and recall, and the other between time complexity and search accuracy. Finally, the accurate discovery results are achieved with very low time consumptions.

There are several adjustable parameters that affect the duplicate clustering stage, including the parameter k, the cutting threshold d and the length of the sparse code in the initial clustering process. The different sets of the parameters could result in contrasting the clustering accuracies and time costs, and our goal is to find a proper set of the parameters that can produce accurate clustering results efficiently.

First we consider the setting of parameter k and the cutting threshold d in the our method. A larger k causes more candidates are assigned into the reduce phase. Intuitively, the increasing of k helps to improve the recall and precision rates, at the same time, there are more image pairs for calculating the similarity. There always exists a precision-recall tradeoff in both document and image duplicate clustering systems. For example, decreasing the cutting threshold may produce more accurate pairs of near duplicates to improve the clustering precision, but also it may reject some correct pairs of near duplicates and therefore degrades the recall.

To be clear, we test different combinations of k and d, and plot the recall, precision, F-measure and time values under different settings in Fig. 3. It is easy to observe that although large k helps to improve the recall and precision rates, the precision value does not always increase when the k goes up. In fact, we can observe from Fig. 3(d) that if k becomes too

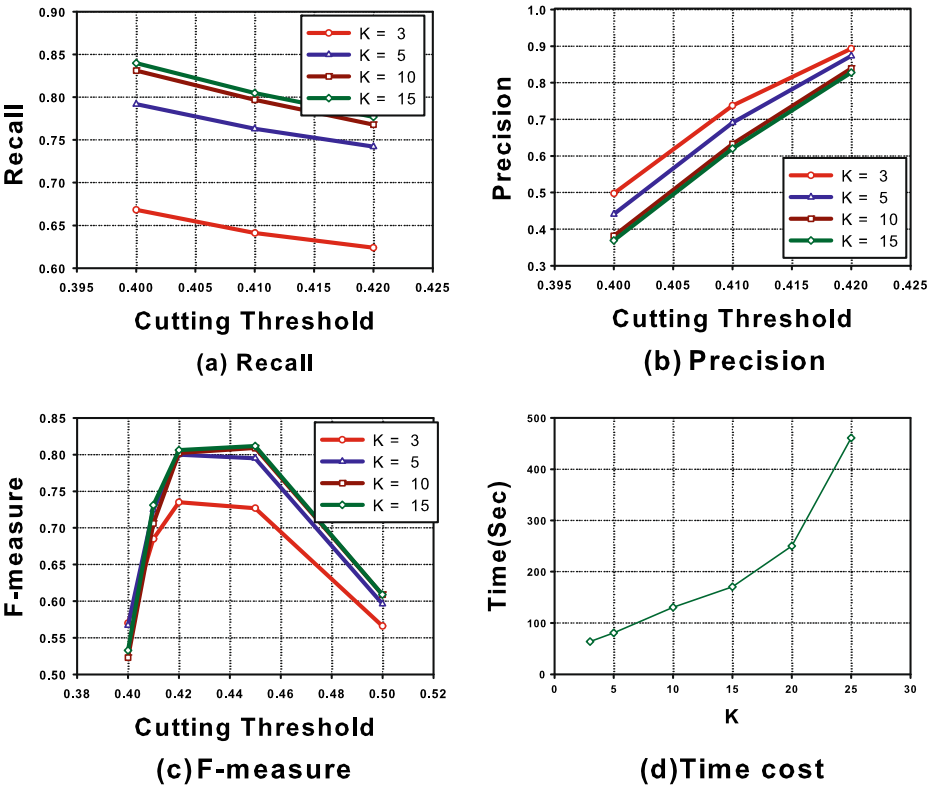

Fig. 3 Performance on our labeled dataset under different K. F-measure is the harmonic mean of precision and recall

large, the time cost will grow exponentially. On the other hand, an improper value of d could also cause the drop of the clustering accuracy rate.

Next we configure the setting of the length of sparse code in photo representation. Because the images, which have identical signatures, are assigned into the same buckets in our algorithm, the length of the sparse code may affect how many candidate images can be assigned into one bucket directly. Figure 4 shows the trade-off between the sparse code length and quality for our labeled photo-set. Due to only the value of the nonzero items were stored and computed, longer sparse code may not result in more time cost significantly. One can observe that the increasing of the length of sparse code may promote the slightly and reduce the time cost rapidly.

4.3 Experimental results

In parameter analysis part, we have tested the effect of parameter k, cutting threshold d and the length of the sparse code, by varying them from 3 to 25 , 0.35 to 0.5 and 1K to 8K, respectively. When we choose $k = 15$, $d = 0.4$ and the size of codebook is $8K$, we can

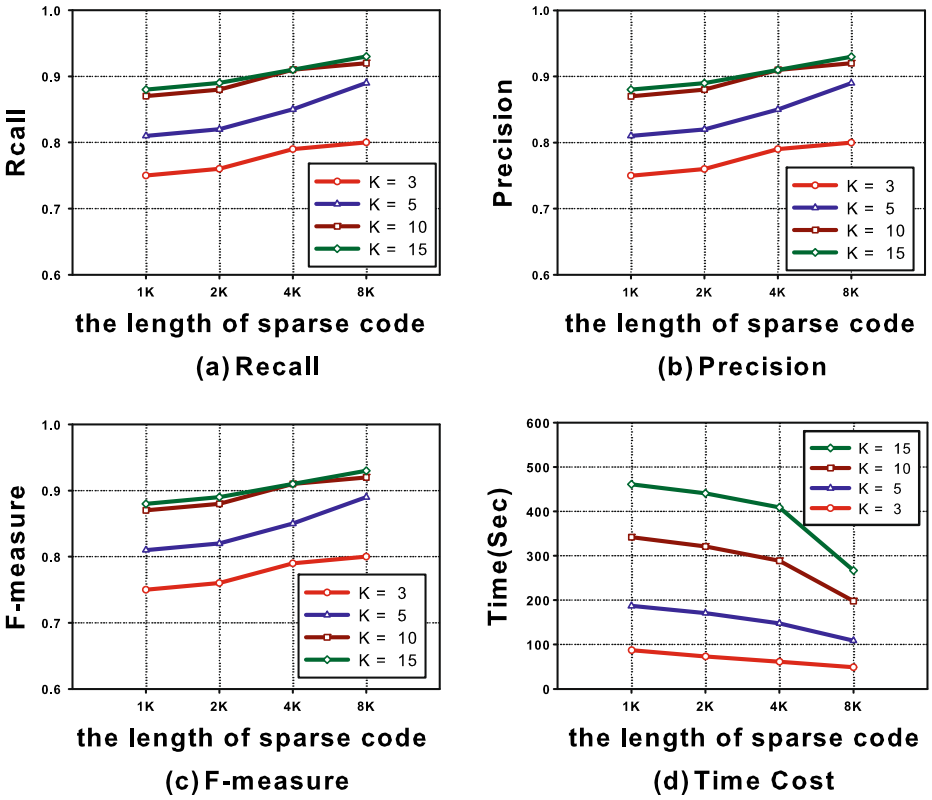

Fig. 4 Performance on our labeled dataset under different length of sparse code

get the best F-measure for labeled image-set. We shall inherit these parameters in the later experiments.

We have tested and compared four near duplicate detection methods. The first one is the Bag-of-Visual-Words approach with visual vocabulary tree [20], denoted as the baseline approach. The second one uses a standard LSH(st.LSH) [7], the random projection based hash function in Euclidean. The third one is min-hash(mH) method [6] which describes the images by selecting independent visual words as the global descriptors and group them into s-tuples(sketches) for hashing. The fourth one is Geometric min-Hash algorithm(GmH) [5], it improves upon standard min-hash by considering the dependency among visual words. The fifth one is Partition min-Hash algorithm(PmH) [16] which first divided image into overlapping partitions and then applied independently to the visual words within each partition to compute a min-hash value. They were compared under two scenarios: constant number of distractors with different code sizes (Fig. 5) and constant runtime (Fig. 6). The distractors were combined by the labeled set and the Flickr-100M. One can observe from Fig. 5 that our approach can significantly outperform all the other methods when the sizes of datasets are increased.With less code size, our method is significantly superior to oth-

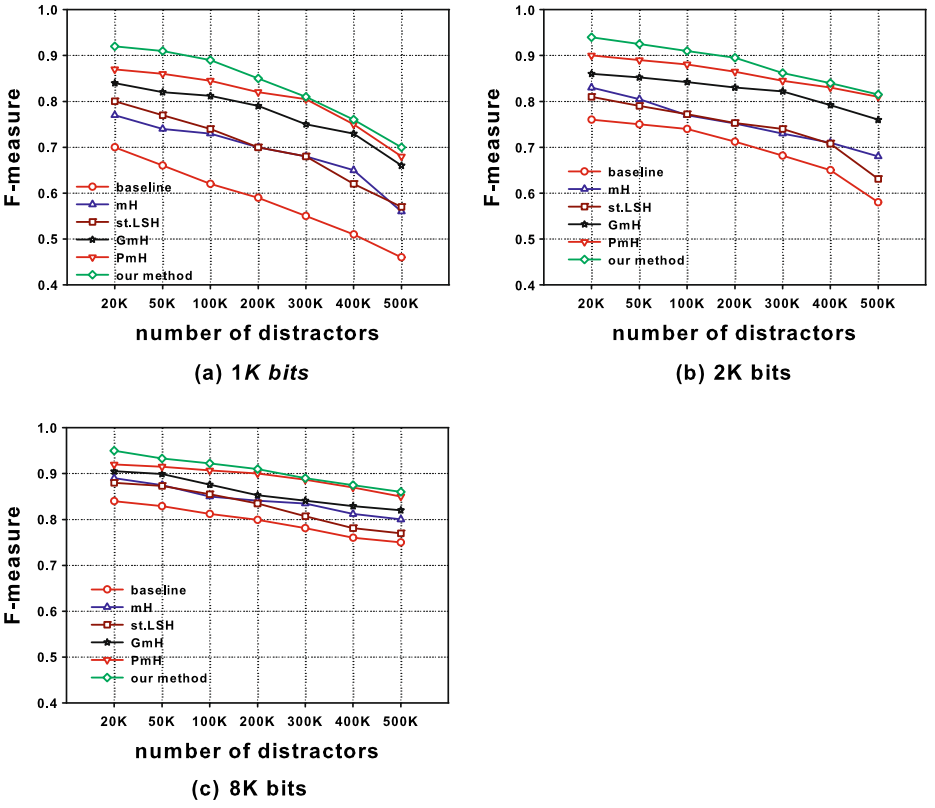

Fig. 5 Performance comparison using different code lengths (**a**) 1K bits, (**b**) 2K bits and (**c**) 8K bits on the our labeled dataset + Flicker100M with different numbers of distractors

ers.The second best is PmH or GmH. We demonstrate the efficiency of our proposed method by comparing it with the candidate systems on a single machine. Because the candidate methods require to fetch all the pairs of the images and compute their similarities, we can not load all the image pairs into the memory of one single machine. For the larger dataset, we only report our method based on MapReduce.And the different number of nodes will not affect the accuracy of experiment result. As shown in the Fig. 6, compared with other candidate methods, discovering duplicate images by using our method can reduce the time cost significantly. With 500K distractors, our method on the 10-nodes Hadoop cluster can achieve 5.02 times faster and 7.1 times on the 20-nodes than the best candidate method, mH.

Figure 7 shows some qualitative results on the labeled set, where our method can consistently obtain more accurate grouping results of near-duplicates than the other methods. Because there is no ground truth on labeling for the larger Flickr-100M set, we could not to perform objective evaluation, we show some subsets of the actual clusters in Fig. 8.

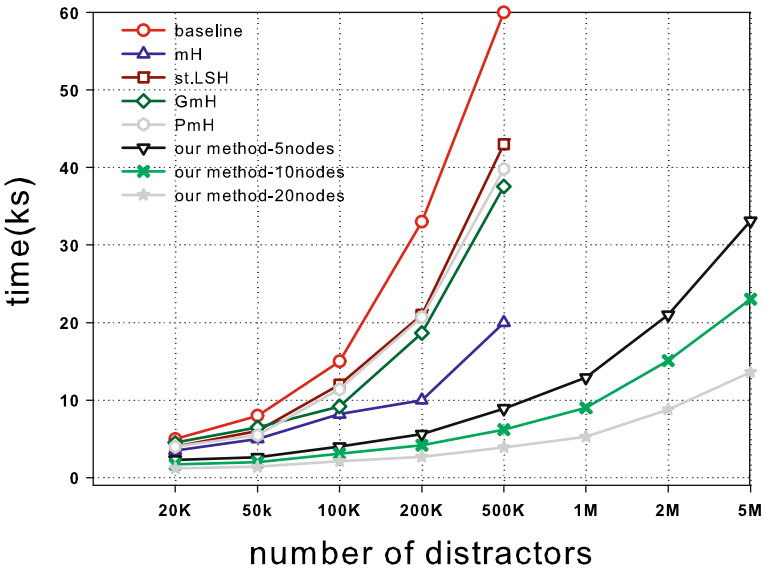

Fig. 6 Comparison of the runtime (kilosecond) with different numbers of distractors

Fig. 7 The partial result of four examples near-duplicate group on our labeled set. Red border denotes false positive, F denotes F-measure of sample category

Fig. 8 Five examples of near-duplicate clusters on Flicker images by our approach

5 Conclusions

In this paper, the problem of near-duplicate image clustering is invested, and an efficient and effective solution is developed. We use LLC with maxIDF cut model to represent the image feature. By applying MapReduce-based image partitioning method and pairwise merging, the image duplicates can be identified effectively from large scale image set. An intuitive tradeoff strategy is also introduced to guide the process for parameter selection.

Experimental results on the large-scale image sets have revealed that our algorithm achieves significant accuracy improvement compared to the baseline methods. Moreover, by integrating MapReduce framework to enable parallel computation, our method can obtain the clusters of image duplicates with 20 times speed-up on a Hadoop cluster that consists of 10 nodes, compared with baseline methods that are performed on one single machine.

Acknowledgments This research is partly supported by National Science Foundation of China under Grant 61272285, National High-Technology Program of China (863 Program, Grant No.2014AA015201), Program for Changjiang Scholars and Innovative Research Team in University (No.IRT13090), and Program of Shaanxi Province Innovative Research Team (No.2014KCT-17).

References

1. Bayardo RJ, Ma Y, Srikant R (2007) Scaling up all pairs similarity search. In: Proceedings of the 16th international conference on World Wide Web, pp. 131–140. ACM
2. Broder AZ (1997) On the resemblance and containment of documents. In: Compression and Complexity of Sequences 1997. Proceedings, pp. 21–29. IEEE

3. Broder AZ, Glassman SC, Manasse MS, Zweig G (1997) Syntactic clustering of the web. Computer Networks and Isdn Systems 29(8-13):1157–1166
4. Cherian A, Morellas V, Papanikolopoulos N (2012) Robust sparse hashing. In: Proceedings / ICIP.. International Conference on Image Processing, pp. 2417–2420
5. Chum O, Perdoch M, Matas J (2009) Geometric min-hashing: Finding a (thick) needle in a haystack. In: Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on, pp. 17–24. IEEE
6. Chum O, Philbin J, Zisserman A, et al. (2008) Near duplicate image detection: min-hash and tf-idf weighting. In: BMVC, vol. 810, pp. 812–815
7. Datar M, Immorlica N, Indyk P, Mirrokni VS (2004) Locality-sensitive hashing scheme based on p-stable distributions. In: Proceedings of the twentieth annual symposium on Computational geometry, pp. 253–262. ACM
8. Dean J, Ghemawat S (2008) Mapreduce: simplified data processing on large clusters. Commun ACM 51(1):107–113
9. Dong W, Wang Z, Charikar M, Li K (2012) High-confidence near-duplicate image detection. In: Proceedings of the 2nd ACM International Conference on Multimedia Retrieval
10. Elsayed T, Lin J, Oard DW (2008) Pairwise document similarity in large collections with mapreduce. In: Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics on Human Language Technologies: Short Papers, pp. 265–268. Association for Computational Linguistics
11. Foo JJ, Zobel J, Sinha R (2007) Clustering near-duplicate images in large collections. In: Proceedings of the international workshop on Workshop on multimedia information retrieval, pp. 21–30
12. Hama H, Zin TT, Tin P (2009) A hybrid ranking of link and popularity for novel search engine. International Journal of Innovative Computing. Inf Control 5(11):4041–4049
13. Hsieh LC, Wu GL, Hsu YM, Hsu W (2014) Online image search result grouping with mapreduce-based image clustering and graph construction for large-scale photos. J Vis Commun Image Represent 25(2):384–395
14. Hsieh LC, Wu GL, Lee WY, Hsu W (2012) Two-stage sparse graph construction using minhash on mapreduce. In: IEEE International Conference on Acoustics, pp. 1013–1016
15. Kim S, Wang XJ, Zhang L, Choi S (2015) Near duplicate image discovery on one billion images. In: 2015 IEEE Winter Conference on, Applications of Computer Vision (WACV), pp. 943–950
16. Lee DC, Ke Q, Isard M (2010) Partition min-hash for partial duplicate image discovery. In: European Conference on Computer Vision, pp. 648–662. Springer
17. Liu T, Rosenberg C, Rowley H, et al. (2007) Clustering billions of images with large scale nearest neighbor search. In: Applications of Computer Vision, 2007. WACV'07. IEEE Workshop on, pp. 28–28. IEEE
18. Peng J, Shen Y, Fan J (2013) Cross-modal social image clustering and tag cleansing. J Vis Commun Image Represent 24(7):895–910
19. Salakhutdinov R, Hinton GE (2007) Learning a nonlinear embedding by preserving class neighbourhood structure. In: International Conference on Artificial Intelligence and Statistics, pp. 412–419
20. Sivic J, Zisserman A (2003) Video google: A text retrieval approach to object matching in videos. In: Computer Vision, 2003. Proceedings. Ninth IEEE International Conference on, pp. 1470–1477. IEEE
21. Thomee B, Shamma DA, Friedland G, Elizalde B, Ni K, Poland D, Borth D, Li LJ (2015) The new data and new challenges in multimedia research. arXiv preprint. arXiv:1503.01817
22. Vonikakis V, Jinda-Apiraksa A, Winkler S (2014) Photocluster: A multi-clustering technique for near-duplicate detection in personal photo collections. In: Computer Vision Theory and Applications (VISAPP), 2014 International Conference on, pp. 153–161
23. Wang H, Zhu F, Xiao B, Wang L, Jiang YG (2014) Gpu-based mapreduce for large-scale near-duplicate video retrieval. Multimedia Tools & Applications 74(23):10,515–10,534
24. Wang J, Kumar S, Chang SF (2012) Semi-supervised hashing for large-scale search. Pattern Analysis and Machine Intelligence. Tran IEEE 34(12):2393–2406
25. Wang J, Yang J, Yu K, Lv F, Huang T, Gong Y (2010) Locality-constrained linear coding for image classification. In: Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on, pp. 3360–3367. IEEE
26. Wang XJ, Zhang L, Liu C (2013) Duplicate discovery on 2 billion internet images. In: Computer Vision and Pattern Recognition Workshops (CVPRW), 2013 IEEE Conference on, pp. 429–436
27. Weiss Y, Torralba A, Fergus R (2009) Spectral hashing. In: Advances in neural information processing systems, pp. 1753–1760
28. Xie L, Tian Q, Zhou W, Zhang B (2014) Fast and accurate near-duplicate image search with affinity propagation on the imageweb. Comput Vis Image Underst 124:31–41

29. Yang C, Peng J, Fan J (2012) Image collection summarization via dictionary learning for sparse representation. In: 2012 IEEE Conference on Computer Vision and Pattern Recognition, pp. 1122–1129
30. Zheng L, Wang S, Liu Z, Tian Q (2013) Lp-norm idf for large scale image search. In: Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on, pp. 1626–1633. IEEE

Wanqing Zhao is currently working toward the Ph.D degree in Northwest Univercity of China with the School of Information and Technology. His research interests include multimedia data mining, machine learning and parallel computation.

Hangzai Luo is a professor of Northwest University of China. He received his PhD degree at The University of North Carolina at Charlotte in 2007. His research focuses on techniques related to mining critical information in large-scale multimedia databases.

Jinye Peng is a professor at Northwest University of China. He received PhD degree from Northwestern Polytechnical University of China. His research interests include computer vision, pattern recognition and signal processing.

Jianping Fan is a professor at University of North Carolina at Charlotte. He received his Ph.D degree in computer science and technology from Shanghai Institute of Optics and Fine Mechanics of CAS. His research interests include semantic image and video analysis, computer vision, cross-media analysis, and statistical machine learning.