Near Duplicate Image Discovery on One Billion Images

Saehoon Kim *
Department of Computer Science,
POSTECH, Korea
kshkawa@postech.ac.kr

Xin-Jing Wang
Web Search and Mining Group
Microsoft Research Asia, Beijing
xjwang@microsoft.com

Lei Zhang
Web Search and Mining Group
Microsoft Research Asia, Beijing
leizhang@microsoft.com

Seungjin Choi
Department of Computer Science,
POSTECH, Korea
seungjin@postech.ac.kr

Abstract

Near-duplicate image discovery is the task of detecting all clusters of images which duplicate at a significant region. Previous work generally take divide and conquer approaches composed of two steps: generating cluster seeds using min-hashing, and growing the seeds by searching the entire image space with the seeds as queries. Since the computational complexity of the seed growing step is generally $O(NL)$ where N and L are the number of images and seeds respectively, existing work can hardly be scaled up to a billion-scale dataset because L is typically millions. In this paper, we study a feasible solution of near-duplicate image discovery on one billion images, which is easily implemented on MapReduce framework. The major contribution of this work is to introduce the seed growing step designed to efficiently reduce the number of false positives among cluster seeds with $O(cNL)$ time complexity, where c is small enough for a billion-scale dataset. The basis component of the seed growing step is a bottom-k min-hash, which generates different signatures in a sketch to remove all candidate images that share only one common visual word with a cluster seed. Our evaluations suggest that the proposed method can discover near-duplicate clusters with high precision and recall, and represent some interesting properties of our 1 billion dataset.

1. Introduction

Near-duplicate image discovery is to detect all clusters composed of images which duplicate at a significant region. Let T be a partition of an image space. The task is to minimize the cost with an optimal T^*, so that the sum of distances between images and their cluster centers is minimized, as denoted in Eq.1:

$$T^* = \arg\min_{T} \sum_{i=1}^{|T|} \sum_{x_j \in T_i} dist(x_j, c_i) \quad (1)$$

where $dist(\cdot,\cdot)$ is a distance measure, x_j is an image, and c_i is the center of subspace T_i. For a distance measure, state-of-the-art methods [5, 6] utilize Jaccard similarity, measuring the overlapping ratio of the two sets. An image is typically represented as a set of visual words, and henceforth Jaccard similarity is well suited to define near-duplicate images. Two images are defined as near-duplicates if $dist(\cdot) \leq \theta$, where θ is a judiciously selected threshold.

Given a billion-scale dataset, near-duplicate image discovery can be an essential element of computer vision applications: removing redundancy in image search index to save cost, discovering spatially related images [5], generating mid-level instance-specific visual descriptors [14, 18], etc. Figure 1 suggests that duplicated regions marked out

Figure 1. Near-duplicate clusters discovered from our one billion dataset with the repeated visual words marked as red circles. The repeated pattern has the potential to generate mid-level instance-specific features.

*This work was done during an internship at Microsoft Research Asia.

by repeated visual words can be used to generate mid-level (instance-specific) features.

State-of-the-art approaches [5, 6] optimize Eq.1 with two steps: cluster seeds (analogous with cluster centers) generation and seed growing. An efficient, nonparametric approach is generally adopted for the seed generation step as a good guess of c_i. Specifically, min-hashing [1] is used to partition $\mathbb{R}^d$ into subspaces, and a number of cluster seeds c_i are generated from each subspace, in which c_i is a set of repeated visual words from an image pair. Seed growing is performed by image retrieval using c_i as queries, which is an $O(NL)$ approach, where there are N images and $L = |T|$ seeds.

We argue that the retrieval-based seed growing method cannot be scaled up to a billion-scale image dataset. According to our observation, L is typically in million-scale when $N \geq 1$ billion, and basic knowledge on algorithm tells us that it is too high computational complexity.

In this paper, we propose a novel near-duplicate image discovery approach on a billion-scale dataset. While we adopt the same divide-and-conquer idea of [5, 6], the major difference lies in the seed growing step. Instead of performing image retrieval, we solve seed growing with multiple min-hash filters of which the computational complexity is $O(cNL)$, where c is very small for a billion-scale dataset. The major challenge we met during our implementation is the high false positive ratio (i.e. candidate images for a cluster seed we need to verify based on the original image descriptors). We fight the challenge with two methods, which are our major contributions:

- a novel seed growing function which is highly scalable and easily implemented on the MapReduce framework [9]. It is an ensemble of multiple min-hash filters that efficiently and effectively reduces false positive candidate images of a cluster seed.
- bottom-k min-hash [8], which specifically targets at removing candidate images which share only one common visual words with a seed.

In this paper, we abbreviate million as M and billion as B to save space.

2. Related Work

In this section, we review some related work for near-duplicate image discovery. To our best knowledge, there is no previous work to propose a feasible solution for near-duplicate image discovery on a billion-scale dataset.

Large-scale global duplicates discovery. The primary goal of prior work is to build up efficient algorithms to discover all global duplicates on a billion-scale dataset. [15] proposes an approximate nearest neighbor search method for clustering billions of images. The method works since images are represented by global features[1]. As a consequence, the approach is good at clustering global duplicates but is vulnerable to near-duplicates. Figure 2 shows a few examples. Recently, [19] proposes an efficient algorithm to discover all global duplicate clusters on a 2 billion dataset, where PCA-based hashing discovers initial clusters and the clusters are growing until discovering all possible global duplicates. Unfortunately, this algorithm also has some limitations inherently by global features.

Figure 2. Examples of duplicate images discovered by our approach. We name (a) and (b) as *global duplicates*, which can be reasonably discovered by global feature based duplicate discovery techniques (e.g. [15]), whereas (c) and (d) as *near-duplicates* which can only be reasonably addressed by local feature based methods.

Near-duplicate image search. Given a query image, the existing algorithms [7, 10] efficiently search near-duplicate images from a large-scale dataset, where the time complexity of search is reduced from $O(N)$ into $O(cN)$, where c is a small constant. One may want to issue every image as a query to discover all near-duplicate images. Although [7, 10] reduces several orders for searching, near-duplicate image discovery still requires $O(cN^2)$, which suffers from heavy computational cost.

Near-duplicate image discovery. Previous work typically tries to discover spatially related images on a small-scale (million-scale) dataset, which can be easily adapted for near-duplicate image discovery. [5] is the first work for discovering spatially related images from a million-scale dataset. In this work, the authors utilize min-hashing to implement cluster seed generation and cluster seed growing steps, reducing the time complexity of $O(N^2)$ into $O(NL)$, where N is the number of images and L is the number of cluster seed. Unfortunately, as we will observe, there are millions of cluster seeds to make the algorithm infeasible on a billion-scale dataset.

[1]Typical global feature vectors are about hundreds or thousands of dimensions, whereas bag-of-visual-words descriptors are typically defined on a visual codebook with millions of visual words when dealing with billions of images.

3. Background

In this section, we review the properties of min-hashing and our definition of near-duplicate images.

3.1. Min-Hashing

Min-hashing [2] is a randomized algorithm to preserve Jaccard similarity, where the images with high similarity are collided into the same hash bucket. More specifically, given the example represented by a set $\boldsymbol{x} = \{x_1, ..., x_n\}$, a min-hash signature is defined as $h(\boldsymbol{x}) = \arg\min_{x_i} \pi(x_i)$, where $\pi(\cdot)$ is a random permutation. Given the two sets, $\boldsymbol{x}$ and $\boldsymbol{y}$, the Jaccard similarity is defined as

$$J(\boldsymbol{x}, \boldsymbol{y}) = \frac{|\boldsymbol{x} \cap \boldsymbol{y}|}{|\boldsymbol{x} \cup \boldsymbol{y}|}. \tag{2}$$

The collision probability of a min-hash signature is the same with Jaccard similarity:

$$P(h(\boldsymbol{x}) = h(\boldsymbol{y})) = J(\boldsymbol{x}, \boldsymbol{y}). \tag{3}$$

If a single min-hash signature is used, many false positives are resided into the same hash bucket. To remove the false positives, a couple of independent min-hash signatures are grouped together, which is called as a sketch. The collision probability is decreased by the sketch length, where

$$P(s(\boldsymbol{x}) = s(\boldsymbol{y})) = J(\boldsymbol{x}, \boldsymbol{y})^k, \tag{4}$$

where $s(\cdot)$ is a sketch and k is a sketch length.

If a large sketch length is used, the number of false positive is decreased, but the number of false negative is increased, i.e. recall is decreased. To improve recall. multiple sketches are introduced, where two examples are reduced in the same hash bucket if they share at least one of sketches. The collision probability when the sketch size is k and the number of sketch is M is described as

$$P(collsion) = 1 - (1 - J(\boldsymbol{x}, \boldsymbol{y})^k)^M. \tag{5}$$

For the random permutation, a linear hash function is commonly used [13, 3]:

$$\pi(x) = (a \times x + b) \bmod n, \tag{6}$$

where a and b are random integers and n is a prime number larger than the number of visual word.

3.2. Definition of Near-Duplicates

In this section, we introduce some information of our dataset and our definition[2] of near-duplicates, which helps to understand the proposed algorithm.

[2]A solid scientific measurement of near-duplicate images is beyond the scope of this paper.

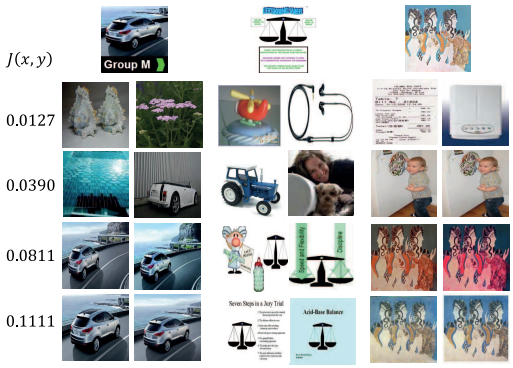


Figure 3. Three examples of the effect of $J(\boldsymbol{x}, \boldsymbol{y})$ on duplicate image suggestions. The query images are located on the top.

Our dataset consists of 1B images randomly collected from Bing. We represent an each image by a set of visual words as in [16]. Specifically, we build up 1M visual words (stop words are removed) by k-means clustering on an independent dataset. To represent an image by a set of visual words, we extract SIFT features of an image and select the 40 most salient ones similar to the approach of [10]. Then, using 1M visual words, an image is represented by a set of 40 visual words.

Since an image is represented by a discrete set [7, 5], it is natural to use Jaccard similarity to define near-duplicates. In our implementation, two images are assumed as a near-duplicate pair if their Jaccard Similarity $J(\cdot)$ satisfies Eq.7.

$$J(\boldsymbol{x}, \boldsymbol{y}) = \frac{|\boldsymbol{x} \cap \boldsymbol{y}|}{|\boldsymbol{x} \cup \boldsymbol{y}|} \geq \alpha. \tag{7}$$

where x, y are sets of visual words of two images respectively, and α is a threshold.

We observe that $\alpha = 0.0811$ is an empirically good threshold to discover near-duplicates, which means that 6 out of 40 visual words are common of two images. These settings are selected with tedious manual evaluations on a large near-duplicate image set.

Figure 3 visualizes the relationship between $J(\boldsymbol{x}, \boldsymbol{y})$ and the precision of duplicate images discovered. The four $J(\boldsymbol{x}, \boldsymbol{y})$ values correspond to $|\boldsymbol{x} \cap \boldsymbol{y}| = 1, 3, 6, 7$ respectively. The three images in the top row are the query images we used to retrieve the 1B dataset, and the images in the other rows are the detected "duplicate images". Each row of duplicated images corresponds to a specific value of $J(\boldsymbol{x}, \boldsymbol{y})$ labeled at the left-most of Figure 3. There are no explicit clues to suggest why those images are retrieved when they share only one visual word (i.e. $J(\boldsymbol{x}, \boldsymbol{y}) = 0.0127$). With 3 common visual words, images seem to share some visual clues such as visually similar objects (e.g. the white car in the first example, and the picture on the wall in the third example) and similar edges or layouts (e.g. the second example). Challenging near-duplicate images are

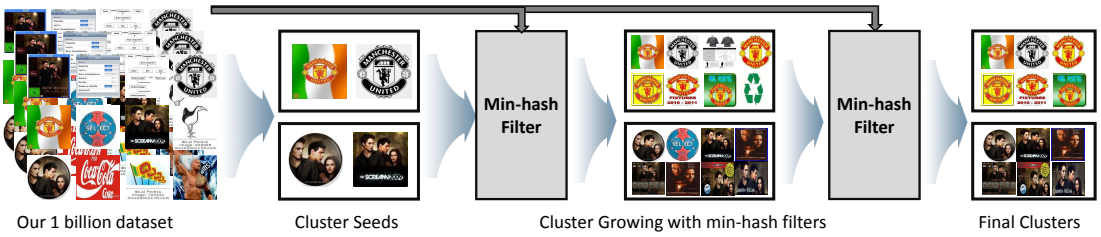


Figure 4. Brief sketch of our near-duplicate discovery approach. Images are first min-hashed into buckets to generate the cluster seeds. The seeds are then grown by enumerating all the images with an efficient seed growing function, which is built upon multiple min-hash filters.

discovered when $J(\boldsymbol{x}, \boldsymbol{y}) = 0.0811$, and diversity on image content reduces as $J(\boldsymbol{x}, \boldsymbol{y})$ increases.

4. The proposed solution

In this section, we detail our solution of discovering near-duplicate images from 1B images, which is summarized in Figure 4. Similar to [12, 11, 14], we also use min-hashing to generate cluster seeds, and then grow the seeds by matching them to the 1B images with multiple min-hash filters. The solution was implemented on the MapReduce framework and the pseudo-codes are summarized in the supplementary material.

4.1. Cluster Seed Generation

Min-hashing is used to efficiently partition images into buckets, where the images having the same sketch (i.e. in the same bucket) are candidates for the cluster seeds. Since images in a bucket are not necessarily near-duplicates, false positives should be removed to identify clean seeds. We form a graph whose nodes are defined by images and an edge occurs if Jaccard similarity between two images satisfy Eq.7 ($\alpha \geq 0.0811$). Then, all connected components of the graph are extracted, and each component is a candidate cluster seed.

A cluster seed is represented as the 40 most frequent visual words that occur in the corresponding connected component. This implementation is a intrinsically simple query expansion, whereas a more complex expansion model considers the spatial layouts of image pairs [5].

Note that 87.6M cluster seeds are generated through this step. Millions of cluster seeds make the cluster seed growing step used in [5, 6] impractical for our 1B dataset, because the time complexity is $O(NL)$, where there are N images and L seeds. In the next section, we state our cluster seed growing approach to effectively remove false positives.

4.2. Cluster Seed Growing

In this section, we describe our cluster seed growing, which is a practical approach for a billion-scale dataset. Cluster seed growing is to collect all near-duplicate images, given the cluster seeds. Before describing the details of our approach, we want to remark that cluster seed growing is the same as "similarity join", and discuss why the recent approaches of similarity join cannot be applied for a billion-scale dataset.

Existing similarity join algorithms have two steps: 1) candidate seed-item pairs generation step, and 2) false positive pairs elimination step (based on the original features). How to generate the small number of candidate seed-item pairs is the focus for similarity join, and our seed growing method as well.

Assuming all items as images for simplicity, one can naïvely collect all similar seed-image pairs by employing an inverted index, where a candidate image has to share at least one visual word with a cluster seed. However, we observed that the number of candidate images per cluster seed is more than 2M on the 1B dataset, estimated on randomly selected 5,000 cluster seeds. This phenomenon suggests that the naïve approach cannot be applied to a billion-scale dataset.

A better approach is to employ a filter [20, 21, 4], which removes false positives in an efficient way before computing the exact distance between a seed and an image. Prefix filtering [4] maintains a small subset of visual words (instead of all visual words) to select candidate images for a seed. However, the low similarity threshold (i.e. $\alpha = 0.0811$ used to define near-duplicates in section 3.2) makes prefix filtering impractical to our case, because we have to keep the most of visual words. Such low similarity threshold also limits advantages of the other filtering-based algorithms [20, 21].

4.2.1 The seed growing function

Our seed growing function measures the probability that an image is a near-duplicate candidate of a cluster seed. Given a cluster seed $\boldsymbol{c}_i$ and an image $\boldsymbol{x}$, Eq.8 gives our seed growing function, in which $p(\boldsymbol{c}_i, \boldsymbol{x})$ is defined in Eq.10 described in section 4.2.2. The core contribution of Eq.8 is the well-designed combination of min-hash functions [17] so that false positive and false negative ratios are well-balanced in the condition that $J(\boldsymbol{c}_i, \boldsymbol{x}) = 0.0811$ defines

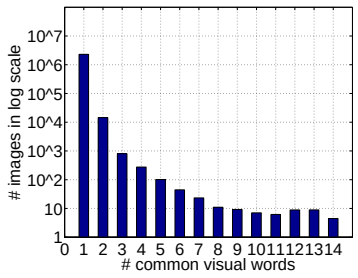
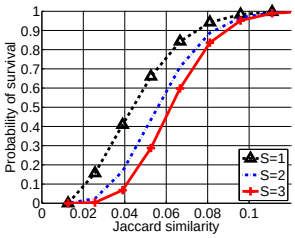


Figure 5. Histogram of the number of images sharing variant numbers of visual words with a cluster seed, evaluated on 5,000 random seeds.



(a)

$J(x,y)$	$p(x,y)$	$S=1$	$S=3$
0.0127	0	0	0
0.0256	3.3e-004	0.1568	0.0039
0.0390	0.0010	0.4086	0.0682
0.0526	0.0021	0.6600	0.2876
0.0667	0.0036	0.8425	0.5980
0.0811	0.0056	0.9422	0.8365
0.0959	0.0080	0.9836	0.9514

(b)

Figure 6. The probability of survival (a) of our seed growing function varying the number of min-hash filters (S), and the detailed probability (b) of the subfigure (a).

near-duplicates.

$$P(survival) = (1-(1-p(\boldsymbol{c}_i,\boldsymbol{x}))^M)^S \triangleq (P_{MHF})^S, \quad (8)$$

where $p(\boldsymbol{c}_i,\boldsymbol{x})$ is an estimator of the similarity between $\boldsymbol{c}_i$ and $\boldsymbol{x}$, P_{MHF} is denoted as a *min-hash filter*, S is the number of min-hash filter and M is a parameter for min-hash filter.

The motivation to multiply $S \geq 1$ min-hash filters for cluster seed growing is to reduce the number of false positive candidate images. Specifically, an image x is a candidate duplicate for a cluster seed c_i if and only if all min-hash filters judge that it is a candidate, i.e. $p(survival) > 0$.

For the min-hash filter, we can use the standard min-hash, but this will lead to a severe computational bottleneck due to the numerous number of images which share one visual word with a cluster. Figure 5 represents that the number of images which share only one visual word with a cluster seed is about 2M. We denote this phenomenon as *one-common-word problem*. If we use the standard min-hash with $k = 2$ and $M = 512$, the collision probability is 0.079 when $J(\boldsymbol{c}_i,\boldsymbol{x}) = 0.0127$ (i.e. sharing one common word), resulting in at least 2M $\times$ 0.079 $\approx$ 0.16M false positives per cluster seed. This requires 0.16M $\times$ 87.6M times of pair verification based on original image descriptors, where the number of cluster seeds is 87.6M.

One may want to increase S (i.e. the number of min-hash filters) to reduce the number of false positives rapidly. To implement the seed growing function, we should store the intermediate results for an individual min-hash filter and take the intersection to reduce the number of false positives. When we use the standard min-hash with k = 2 and M = 512, the intermediate results require at least 0.16M $\times$ 87.6M $\times$ 4bytes = 50.6TB storage, where the number of cluster seeds is 87.6M, the number of false positives is 0.16M, and 4bytes are used to represent an image. Therefore, it is no be useful to increase S.

Finally, one may want to increase k (i.e. sketch length) to eliminate false positives, but this setting should increase M to achieve a reasonable survival probability. Large M requires large memory footprint and high computational costs. Therefore, in the next section, we use a bottom-k min-hash as the basis of a min-hash filter to efficiently remove the false positives.

4.2.2 Min-Hash Filter

A bottom-k min-hash [8] is used as a min-hash filter to resolve the one-common-word problem. The standard min-hash formulation (Eq.5) can generate a sketch with the same signatures due to the independence assumption. The bottom-k min-hash breaks this independence in signature generation, which is defined as Eq.9.

$$h_i(\boldsymbol{x}) = \text{the i-th smallest element of } \pi(v), \quad i = 1,...,k, \quad (9)$$

where $\pi(v)$ defines a permutation on the set of visual words x. Eq.9 means that all the signatures in a sketch should be different.

Therefore, $p(\boldsymbol{x},\boldsymbol{y})$ in our seed growing function can be calculated as

$$p(\boldsymbol{x},\boldsymbol{y}) = \frac{\binom{\boldsymbol{x}\cap\boldsymbol{y}}{k}}{\binom{\boldsymbol{x}\cup\boldsymbol{y}}{k}}, \quad (10)$$

where $\binom{}{}$ is a binomial coefficient and $\binom{a}{b} = 0$ when $a < b$. If $k \geq 2$, $p(\boldsymbol{x},\boldsymbol{y}) = 0$ when $|\boldsymbol{x}\cap\boldsymbol{y}| = 1$, which obviously removes the candidate images sharing only one common word with a cluster seed.

Figure 6 shows the survival probability of the seed growing function with the bottom-k min-hash ($k = 2$ and $M = 512$), varying the number of min-hash filters (S). As in Figure 6, the survival probability with respect to small Jaccard similarity decreases rapidly by multiplying multiple min-hash filters, which means that false positive images are rapidly filtered out. For example, if $S = 3$, $P(survival) = 6.82\%$ when $J(\boldsymbol{x},\boldsymbol{y}) = 0.0390$, which removes about $83.3\% = (0.4086 - 0.0682)/0.4086$ candidates compared to a single min-hash filter. As a consequence, we observe that the storage cost of all candidate

Table 1. Cluster and image distributions on cluster size

cluster size	≥ 2	≥ 10	≥ 100	$\geq 1K$
the number of clusters	82.2M	4.5M	0.1M	1.7K
the number of images	344.7M	120.5M	29.5M	6.7M

Table 2. Cluster and image distributions on average Jaccard similarity

average $J(\boldsymbol{x}, \boldsymbol{y})$ of a cluster	≤ 0.2	≤ 0.4	≤ 0.7	≤ 1
the number of clusters	14.8M	39.1M	57.8M	82.2M
cluster distribution (%)	18.0%	47.6%	70.3%	100%
the number of images	76.9M	201.0M	282.0M	344.7M
image distribution (%)	22.3%	58.3%	81.8%	100%

seed-image pairs decreases from 625GB to 20GB when S increases from 1 to 3. Meanwhile, true positive images (i.e. $J(\boldsymbol{x}, \boldsymbol{y}) > 0.0811$) still have large chance of survival, e.g. $P(survival) = 95.14\%$ when $J(\boldsymbol{x}, \boldsymbol{y}) = 0.0959$.

Since the proposed seed growing function effectively removes the false positives, the time complexity for growing a single cluster seed is $O(cN)$, where c is a very small constant. Therefore, the time complexity for the cluster seed growing step is $O(cNL)$, which makes it practical for a billion-scale dataset.

4.3. Post-processing

We merge two clusters if there are 40% images in common and the Jaccard similarity of their corresponding cluster seeds exceeds 0.0811. In addition, the clusters whose member images have low visual consistency (such as highly textured or textual images) are removed, if the average Jaccard similarity of the member images is less than 0.06.

5. Experiments

We conducted a series of evaluations to measure our proposed method and compared it to several challenging state-of-the-art baselines. We implemented our algorithm on MapReduce framework with 250 nodes. The detailed pseudo code is described in the supplementary material.

5.1. The Distribution of Duplicate Images

From the 1B images, we discovered about 82.2M clusters, which contain about 344.7M near-duplicate images. Figure 8 shows the sample near-duplicate clusters discovered in our 1B images. The distributions of clusters and images are shown in Table 1. From this table, it can be seen that $94.5\% = (82.2M - 4.5M)/82.2M$ clusters are small which contain less than 9 images. These clusters hold $65.0\% = (344.7M - 120.5M)/344.7M$ images of all the images that have at least one duplicate. Contrarily, very large clusters which contain more than 1K images occupy only a very small proportion of 0.002% clusters, but correspond to 1.94% images.

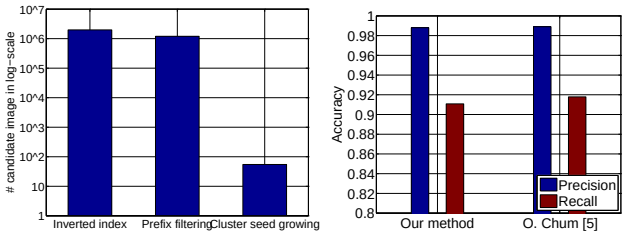
Table 2 illustrates the distributions of clusters and images versus the average Jaccard similarity of images in a cluster.

Table 3. Cluster distribution on average Jaccard similarity on the pseudo ground truth set

the averaged similarity	≤ 0.2	≤ 0.4	≤ 0.7	≤ 1
the number of cluster	955	3,569	5,931	7,868
Ratio (%)	12.1%	45.3%	75.3%	100%

Note that $J(\boldsymbol{x}, \boldsymbol{y}) \leq 0.2$ generally suggests that most of the images in a cluster are near-duplicates rather than global duplicates or exact duplicates, which have very large variance in appearances. Contrarily, $J(\boldsymbol{x}, \boldsymbol{y}) = 1$ generally suggests that almost all of the images in a cluster are exact duplicates. The larger the Jaccard similarity, the smaller the ratio of near-duplicates. From Table 2, we can see that clusters which tend to be full of near-duplicates occupy a population of 18.0% of all the clusters, whereas those which are dominated by global or exact duplicates (i.e. $J(\boldsymbol{x}, \boldsymbol{y}) \geq 0.7$) occupies $29.7\% = (82.2M - 57.8M)/82.2M$ of the cluster population. This suggests that near-duplicate images should be less popular than global or exact duplicates on the web.

5.2. Performances on Pseudo Ground Truths



(a) on 1B images (b) on 1M images

Figure 7. Performance comparison to state-of-the-art baselines. (a) Our method is much more powerful than existing similarity join methods in reducing false positive candidates. (b) Our solution achieves a comparable performance with [5] on a random 1M dataset.

We would like to measure the average precision and recall of the discovered image clusters, which requires a ground truth dataset. Since it is impossible to generate the true ground truth dataset from billions of images, we adopt an image retrieval approach to construct a pseudo ground truth dataset.

We randomly selected 7,868 images which have at least one global duplicate image by the global feature used in [19]. Each image is again represented as a set of visual words and is used as a query to retrieve near-duplicates from the 1B dataset. The similarity metric is $J(\boldsymbol{x}, \boldsymbol{y}) \geq 0.08$. By this means, we obtained the ground truth cluster T_{q_i} for a query image q_i. Table 3 sheds some lights on the ground truth dataset T_{q_i}. Comparing this table to Table 2, it can be seen that our sampling generates a similar cluster distribution as that on the entire 1B dataset, which means it is a reasonable sampling, so that the average precision and

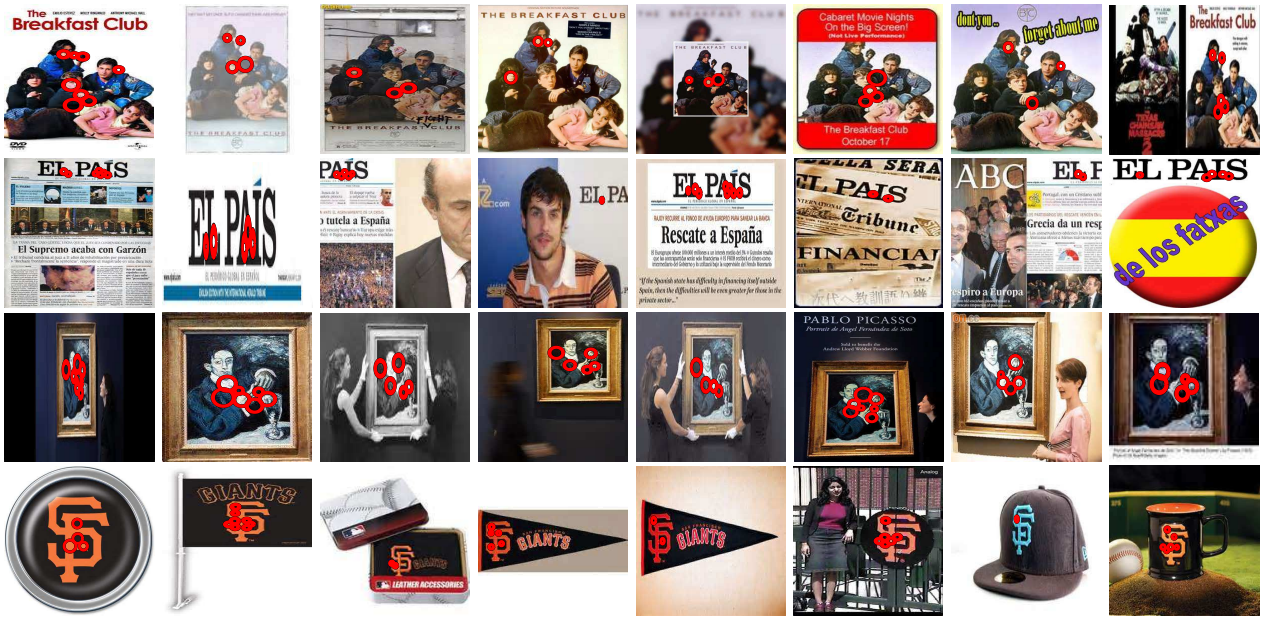


Figure 8. Samples of near-duplicate clusters whose average Jaccard similarity is less than 0.1. Red circles highlight the repeated local interest points which drive images into a cluster. More examples can be found in the supplementary material.

recall performances evaluated based on this ground truth set should reasonably represent the true performance of our method.

We use below criteria to measure the average precision and recall:

$$precision = \frac{1}{m}\sum_{i=1}^{m}\frac{|C_{q_i} \cap T_{q_i}|}{|C_{q_i}|} \quad (11)$$

$$recall = \frac{1}{m}\sum_{i=1}^{m}\frac{|C_{q_i} \cap T_{q_i}|}{|T_{q_i}|}, \quad (12)$$

where C_{q_i} is a discovered image cluster which contains q_i.

We compare our method to several state-of-the-art baselines. Firstly, we compare the power of reducing false positive images to the two similarity join methods, i.e. inverted index and prefix filtering, as shown in Figure 7(a). It can be seen that, prefix filtering slightly improves inverted index, whereas our method greatly outperforms the two baselines.

Secondly, we randomly selected 1M distracter images from the 1B dataset and mixed them with the ground truth set, on which we applied our method and Chum's method (without spatial verification) [5]. The result is shown in Figure 7(b). Our method achieved similar precision performance but slightly worse recall performance than [5]. The worse recall is because Chum's method [5] is about a linear scan on the entire database, whereas our method effectively removes false positives with small false negative ratio, as we discussed in Section 4.2.2. In addition, high precision and recall in Figure 7(b) suggest that our method discovers all full-duplicate clusters with additional near-duplicate clusters. This fact leads us to conclude that our method can discover much more clusters compared with the full duplicate image discovery method [19].

5.3. Evaluation by User Judgement

We conducted user study to measure that the images of a discovered near-duplicate cluster are really near-duplicate each other. A user scores a cluster from 5 (very good) to 0 (bad), and the scoring guideline is summarized as 5 (precision$\geq$ 95%), 4 ($\geq$ 90%), 3 ($\geq$ 80%), 2 ($\geq$ 70%), 1 ($\geq$ 60%), and 0 ($<$ 60%). Ten users were involved in the evaluation and each of them processed 100 clusters of which average Jaccard similarity is less than 0.2. The averaged scores for clusters whose $J(\boldsymbol{x}, \boldsymbol{y}) < 0.1$ and $0.1 \leq J(\boldsymbol{x}, \boldsymbol{y}) < 0.2$ are 3.73 and 4.6 respectively, which suggests nearly 90% and 95% precisions on the two types of clusters respectively.

5.4. Running time

In our experiments, our algorithm takes about 17 hours on 1 billion images with 250 nodes. We expect that [5] takes much more time than our algorithm, because it takes already more than 24 hours on 100 million images with 250 nodes. The major computational bottleneck of [5] is cluster seed growing step, taking O(NL) time complexity, where N is the number of image and L is the number of cluster seed. Moreover, we observed that the execution time for cluster seed growing is increased more than linearly in N (because L is also increased with N). Therefore, [5] is expected to

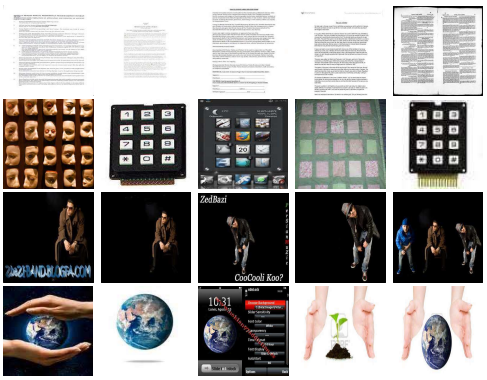


Figure 9. Failure cases of near-duplicate images. Each row shows a subset of discovered clusters. They are either due to the ineffectiveness of image descriptors (the first two clusters), or due to the errors brought by greedy seed generation.

take more than 240 hours on 1 billion images.

5.5. Discussions: Failure Cases

Four examples of noisy clusters (subsets due to space limit) are given in Figure 9 which represent the failure cases we discovered. The first two clusters (one cluster per row) capture the textual or texture patterns of images. Since no existing low-level global or local image descriptors can be generally effective on such images, to achieve high clustering precision on such images requires more powerful visual descriptors. The noises in the third and fourth examples are caused by the cluster seed generation approach (Section 4.1). In the third row, the left four images were grouped into one cluster because they are both near-duplicates for the last image. In the last row, four images are assumed as near-duplicates because of the object "earth", while the hands in the last earth image brings into the cluster the image of hands and trees.

6. Conclusions

We presented a scalable solution of near-duplicate image discovery on billions of images. To our best knowledge, this is the first achievement of local feature-based image clustering in such a scale. Our method divide-and-conquers the problem by first efficiently generating some cluster seeds with min-hashing, and then growing the seeds with a carefully designed growing function which removes the false positives in an efficient manner. We validated the proposed method quantitatively and qualitatively, and discussed several interesting properties of our 1B dataset.

Acknowledgments: This work was supported by the IT R&D Program of MSIP/IITP (14-824-09-014, Machine Learning Center), National Research Foundation (NRF) of Korea (NRF-2013R1A2A2A01067464), and NIPA-MSRA Creative IT/SW Research Project. Portion of this work was performed when SK was visiting Microsoft Research Asia.

References

[1] A. Broder. On the resemblance and containment of documents. In *Proceedings of Compression and Complexity of Sequences*, 1997.

[2] A. Z. Broder. On the resemblance and containment of documents. In *Compression and Complexity of Sequences (SEQUENCES'97)*, 1997.

[3] A. Z. Broder, M. Charikar, A. M. Frieze, and M. Mitzenmacher. Min-wise independent permutations. *Journal of Computer and System Sciences*, 60:630–650, 2000.

[4] S. Chaudhuri, V. Ganti, and R. Kaushik. A primitive operator for similarity joins in data cleaning. In *Proceedings of the 22Nd International Conference on Data Engineering*, 2006.

[5] O. Chum and J. Matas. Large-scale discovery of spatially related images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(2):371–377, 2010.

[6] O. Chum, M. Perdoch, and J. Matas. Geometric min-hashing: Finding a (thick) needle in a haystack. In *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, Miami, Florida, USA, 2009.

[7] O. Chum, J. Philbin, and A. Zisserman. Near duplicate image detection: Min-hash and tf-idf weighting. In *Proceedings of the British Machine Vision Conference*, 2008.

[8] E. Cohen and H. Kaplan. Summarizing data using bottom-k sketches. In *Proceedings of the twenty-sixth annual ACM symposium on Principles of distributed computing*, 2007.

[9] J. Dean and S. Ghemawat. MapReduce: Simplified data processing on large clusters. *Communications of the ACM*, 51(1):107–113, 2008.

[10] W. Dong, Z. Wang, M. Charikar, and K. Li. High-confidence near-duplicate image detection. In *Proceedings of the 2Nd ACM International Conference on Multimedia Retrieval*, 2012.

[11] A. Farhadi and M. A. Sadeghi. Recognition using visual phrases. In *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2011.

[12] R. Fergus, P. Perona, and A. Zisserman. Object class recognition by unsupervised scale-invariant learning. In *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2003.

[13] P. Indyk. A small approximately min-wise independent family of hash functions. In *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 1999.

[14] Q. V. Le, M. Ranzato, R. Monga, and et al. Building high-level features using large scale unsupervised learning. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2012.

[15] T. Liu, C. Rosenburg, and H. A. Rowley. Clustering billions of images with large scale nearest neighbor search. In *IEEE Workshop on Applications of Computer Vision (WACV)*, 2007.

[16] D. Nister and H. Stewenius. Scalable recognition with a vocabulary tree. In *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2006.

[17] A. Rajaraman and J. Ullman. *Mining of Massive Datasets*. Cambridge University Press, 2011.

[18] S. Singh, A. Gupta, and A. A. Efros. Unsupervised discovery of mid-level discriminative patches. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2012.

[19] X.-J. Wang, L. Zhang, and C. Liu. Duplicate discovery on 2 billion internet images. In *Big Data Computer Vision, in conjunction with CVPR*, 2013.

[20] C. Xiao, W. Wang, X. Lin, and J. X. Yu. Efficient similarity joins for near duplicate detection. In *Proceedings of the International Conference on World Wide Web (WWW)*, Beijing, China, 2008.

[21] J. Zhai, Y. Lou, and J. Gehrke. ATLAS: A probabilistic algorithm for high dimensional similarity search. In *Proceedings of the ACM SIGMOD Conference on Management of Data (SIGMOD)*, Athens, Greece, 2011.