Multimed Tools Appl
https://doi.org/10.1007/s11042-018-6178-z

Hierarchical one permutation hashing: efficient multimedia near duplicate detection

Chengyuan Zhang[1,2] · Yunwu Lin[1,2] · Lei Zhu[1,2] · XinPan Yuan[3] · Jun Long[1,2] · Fang Huang[1]

Received: 3 May 2018 / Revised: 17 May 2018 / Accepted: 21 May 2018

Abstract With advances in multimedia technologies and the proliferation of smart phone, digital cameras, storage devices, there are a rapidly growing massive amount of multimedia data collected in many applications such as multimedia retrieval and management system, in which the data element is composed of text, image, video and audio. Consequently, the study of multimedia near duplicate detection has attracted significant concern from research organizations and commercial communities. Traditional solution minwish hashing (MinWise) faces two challenges: expensive preprocessing time and lower comparison speed. Thus, this work first introduce a hashing method called one permutation hashing (OPH) to shun the costly preprocessing time. Based on OPH, a more efficient strategy group based one permutation hashing (GOPH) is developed to deal with the high comparison time. Based on the

✉ XinPan Yuan
xpyuan@hut.edu.cn

✉ Jun Long
jlong@csu.edu.cn

Chengyuan Zhang
cyzhang@csu.edu.cn

Yunwu Lin
lywcsu@csu.edu.cn

Lei Zhu
leizhu@csu.edu.cn

Fang Huang
hfang@csu.edu.cn

[1] School of Information Science and Engineering, Central South University, Changsha, People's Republic of China

[2] Big Data and Knowledge Engineering Institute, Central South University, Changsha, People's Republic of China

[3] School of Computer, Hunan University of Technology, Zhuzhou, People's Republic of China

Published online: 30 June 2018

fact that the similarity of most multimedia data is not very high, this work design an new hashing method namely hierarchical one permutation hashing (HOPH) to further improve the performance. Comprehensive experiments on real multimedia datasets clearly show that with similar accuracy HOPH is five to seven times faster than MinWise.

Keywords Hierarchical one permutation hashing · Multimedia data · Near duplicate detection

1 Introduction

Due to the proliferation of smart phone, video cameras, storage devices as well as the development of social networks, large volumes of multimedia data associating to text, image, video and audio have been generated by users every day. For instance, it is reported that there are over 95 million geo-tagged photos on Flickr with a daily growth rate of around 500,000 new geo-tagged photos. As a result, in recent years variety of multimedia information retrieval applications have emerged and become more and more popular.

As a vital component of multimedia information retrieval applications, near duplicate detection, also known as similarity search, is a well established research topic and still attracts lots of attention. An element is called a near-duplicate of a reference element if it is ”close”, according to some defined measure, to the reference element. For example, given an image and a collection, we want to find those candidates in the collections which are most similar to the input image based on their Jaccard similarity.

As efficient solution of multimedia near duplicate detection, MinWise approach has been paid substantial effort in recent research literatures. For example, in text near duplicate detection, k independent random permutations are generated to convert the words of each document into a set of fingerprints to accelerate similarity search; similarly, in image near duplicate detection, k independent random permutations are applied to convert the visual words of each image into a set of fingerprints.

Though MinWise method are successful in large scale similar multimedia searching, it still faced two critical challenges. Firstly, the preprocessing cost of MinWise can be very expensive. To ensure the accuracy of MinWise, hundreds of independent random permutations are generated. Secondly, MinWise is time consuming, because it has to compare all fingerprints to calculate the similarity.

In order to reduce preprocessing time of MinWise, we first introduce a hashing method OPH, which is widely used in text search area, to multimedia near duplicate detection. Different with MinWise, which requires hundreds of random permutation, OPH only require single permutation, which significantly reduces the time consumption during random permutation. Based on OPH, we further divide the fingerprints into several groups, and bring in the concept of small probability event. With the assistance of small probability event, our proposed GOPH can terminal the comparison earlier, but with similar accuracy. Meanwhile, we observe that most of the data are not similar to the input in real application. Thus, based on this observation, we purpose a new hashing method namely HOPH to further reduce the comparison time.

Contributions The principle contributions of this paper are summarized as follows.

– OPH is introduced to avoid the expensive preprocessing step of MinWise. Based on OPH, an efficient group based algorithm called GOPH is develop to reduce the comparison time.

– To further accelerate the search speed, we proposed a new hashing method HOPH.
– Comprehensive experiments on real and synthetic text and image datasets demonstrate that our proposed hashing method HOPH achieve substantial improvements over MinWise, OPH, but with similar accuracy.

Roadmap The rest of the paper is organized as follows. Section 2 formally defines the problem of multimedia near duplicate detection, followed by the introduction of the related work. Section 3 describes OPH, GOPH and GOPH's image application. Section 4 presents HOPH, some important theoretical analysis of HOPH and its image application. Extensive experiments are depicted in Section 5. Finally, Section 6 concludes the paper.

2 Related work

In this subsection, we present some existing techniques, such as MinWise, *b*-bit MinWise, for the problem of text near-duplicate detection. Then we also present some general techniques related to image similarity computation.

2.1 Near-duplicate detection for text

Near-duplicate detection is one of the key problems in the area of database and information retrieval. It aims to detect groups of documents with almost the same contents among a document collection. Two documents with a great amount of shared attributes do not necessarily count as near-duplicate. For the near-duplicate detection problem, Yang et al. [40] proposed an instance-level constrained clustering approach. Their framework incorporates information such as document attributes and content structure into the clustering process to form near-duplicate clusters. Yang et al. [8] emphasized that the gap between rare words' term frequency in two documents should be smaller than that between common words' and their best ranking is giving by a term weighting function biased toward rare terms. Hassanian-esfahani et al. [6] proposed a Sectional MinWise (S-MinWise) for the detection of near-duplicate documents to enhances the MinWise data structure with information about the location of the attributes in the document.

MinWise [3, 23] is a locality sensitive hashing for the Jaccard similarity, which is a most popular technique for efficiently text similarity computing. It has a wide range of applications such as duplicate detection [7], nearest neighbor search [9], large-scale learning [11], all-pairs similarity [1], etc.. Border et al. [2] invented the MinWise algorithm for near-duplicate web page detection and clustering. Li et al. [11] presented a simple effective solution to large-scale learning in massive and extremely high-dimensional datasets and indicated that *b*-bit MinWise is significantly more accurate than Vowpal Wabbit in binary data. The major defect of MinWise algorithm and *b*-bit MinWise is that they require an expensive preprocessing step, by conducting k permutations on the entire dataset [12]. Pagh et al. [17] studied and addressed the question: How many bits is it necessary to allocate to each summary in order to get an estimate with 1 relative error.

2.2 Similarity computation for image

Similarity computation [25, 26] for image is another important technique which is focused by a great many of researchers. The first significant problem concerning efficient similarity measure is image [31] representations. Scale Invariant Feature Transform (SIFT for

short) [15] proposed by Lowe is a classical approach in image recognition and computer vision area. It aims to detects and describes local features in images [14]. The SIFT descriptor is invariant to uniform scaling, orientation, illumination changes and partially invariant to affine distortion. Many studies of image processing [27, 34] and retrieval [28] use it as one of the basic methods. In order to identify and remove the most false positive matches, Zhou et al. [43] proposed to generate binary SIFT descriptor in a given pair of the images from the original SIFT descriptor. Zhou et al. [44] extended SIFT-based match kernels by integrating the match functions for SIFT and CNN features [36–39]. In order to improve the performance of image retrieval, they proposed a threshold exponential match kernel for CNN features to filter out the images whose the semantic similarity is lower than the threshold. To obtain enhanced performance, Zhang et al. [42] utilized 1000 semantic attributes to revise the vocabulary tree of SIFT descriptors in Bag-of-Word model (BoW for short), which stores only occurrence counts of vector quantized features [19]. The Bag-of-Visual-Word model (BoVW for short) represent a image by a set of visual words applying SIFT descriptor and K-means clustering algorithm [16, 18]. BoVW with $tf - idf$ weighting [21] has proven to be a very successful approach for image and particular object retrieval. Nister et al. proposed a recognition scheme by applying this modal, which scales efficiently to a large number of objects. There are two types of searching methods based on BoVW modal, namely the exact inverted file methods [45] and the hashing methods [4]. Hashing methods [29, 35] are used to solve the problem of image similarity measure [24] and multimedia retrieval [30, 32, 33], which are concerned by the community. The spectral hashing method proposed by Shao et al. [20] and the local sensitive hashing method designed by are belong to the unsupervised hashing methods. Jain et al. [10] a method that applied a Mahalanobis distance function that captures the imagespsila underlying relationships well. This approach combined the MinWise method with distance metric learning. Li et al. [13] presented a method to directly optimize the graph Laplacian by using spectral hashing combined with a distance learning.

For problem of detection of near duplicate images, Chum et al. [5] proposed an efficient way based on a MinWise method, which uses a visual vocabulary of vector quantized local feature descriptors and for retrieval exploits enhanced MinWise techniques. Zhang et al. [41] applied a parts-based representation of each scene by building Attributed Relational Graphs (ARG) between interest points. Based on Stochastic Attributed Relational Graph Matching, they compared the similarity of two images. Torralba et al. [22] proposed a method to learn short descriptors to retrieve similar images from a huge database, which is based on a dense 128D global image descriptor. Jain et al. [10] introduced a method for efficient extension of Locally Sensitive Hashing scheme for Mahalanobis distance. Apparently, both above approaches use bit strings as a fingerprint of the image. Philbin et al. [18] proposed a large-scale object retrieval system and compared different scalable methods for building a vocabulary. Besides, they introduced a novel quantization method based on randomized trees to enhance the performance of image-feature vocabularies construction.

3 Basic apporach

In this section, we first formally define the problem of multimedia near duplicate detection, and then review the hashing method OPH, which is widely used in text search area, in Section 3.2. We propose an advanced hashing method called GOPH to improve the performance of OPH in Section 3.3. Section 3.4 presents the image application of GOPH. Table 1 below summarizes the mathematical notations used throughout this paper.

Table 1 Notations

Notation	Definition
q	a multimedia data (query)
$\mathcal{D}$	the multimedia dataset
$\mathcal{T}$	the similarity threshold
ψ	a random permutation
$\mathcal{V}$	the whole vocabulary space
$\mathcal{N}_m b$	the number of "matched bins"
$\mathcal{N}_e b$	the number of "empty bins"
$\hat{\mathcal{R}}_{mb}$	unbiased estimator
$var(\hat{\mathcal{R}}_{mb})$	variance of OPH
$\mathcal{K}$	the bin number of the comparison part
$\mathcal{X}$	the number of times that the fingerprints are equal
T	the number of match bin after $\mathcal{K}$ comparisons
ϵ	the error tolerance
$\mathcal{I}$	the image dataset
$\mathcal{N}_{mb_h}$	The number of matched bin of HOPH
$\mathcal{N}_{eb_h}$	The number of empty bin of HOPH
$\hat{\mathcal{R}}_{mb_h}$	The unbiased estimator of HOPH
$\otimes$	the empty bin of OPH and HOPH

3.1 Problem definition

Near Duplicate. Given a multimedia data $q \in \mathcal{D}$, any multimedia data $p \in \mathcal{D}$ such that $sim(q, p) < \mathcal{T}$ is a near duplicate of q, where $\mathcal{T}$ is a similarity threshold. Among the available similarity comparison function **sim** which might be exploited, the Jaccard similarity has been chosed in this work, since it has been widely used in different applications. Without loss of generality, this paper mainly consider two types multimedia data, text and image.

3.2 One permutation hashing review

To reduce the times of random permutation of MinWise, Ping Li et al. propose an signature named one permutation hashing in [12]. To generate hundreds of samples, traditional signature MinWise requires hundreds of random permutation. However, only single permutation is enough for OPH, which significantly reduces the time consumption during random permutation.

One permutation hashing First, a random permutation ψ is generated. For each document $\mathcal{D}_i$ a one permutation hashing min $\psi(\mathcal{D}_i)$ is recorded. Consider $\mathcal{D}_1$, $\mathcal{D}_2$, $\mathcal{D}_3 \subseteq \mathcal{V}$ = {0, 1, ..., 16}. Assume $\mathcal{D}_1$ = {1, 2, 5, 10, 12, 15}, $\mathcal{D}_2$ = {1, 2, 6, 10, 12, 14}, $\mathcal{D}_3$ = {2, 9, 10, 12, 14}, we apply the permutation ψ on the three sets and present the corresponding $\psi(\mathcal{D}_1)$, $\psi(\mathcal{D}_2)$, $\psi(\mathcal{D}_3)$ as binary (0/1) vector as what is shown in Fig. 1.

Then, we divide the whole space $\mathcal{V}$ into k bins, and select the first non-zero element as a sample for each bin. In some special case, if there is no non-zero element in the bin, $\otimes$ is used to represent the empty bin. By this way, a random permutation of OPH can generate k sample through k bins. Assume k=4, the sample selected from $\psi(\mathcal{D}_1)$ is [1, 5, 10, 12],

	1				2				3				4			
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
$\psi(\mathcal{D}_1)$:	0	1	1	0	0	1	0	0	0	0	1	0	1	0	0	1
$\psi(\mathcal{D}_2)$:	0	1	1	0	0	0	1	0	0	0	1	0	1	0	1	0
$\psi(\mathcal{D}_3)$:	0	0	1	0	0	0	0	0	0	1	1	0	1	0	1	0

Fig. 1 One permutation hashing example

$\psi(\mathcal{D}_2)$ is [1, 6, 10, 12], and $\psi(\mathcal{D}_3)$ is [2, $\otimes$, 9, 12]. Because we only need to compare the sample within the same bin, we can use the smallest possible representation to represent the actual value. For example, for $\psi(\mathcal{D}_1)$, the final representation is [1, 1, 2, 0]; for $\psi(\mathcal{D}_2)$, the final representation is [1, 2, 2, 0]; similarly, for $\psi(\mathcal{D}_3)$, the final representation is [2, $\otimes$, 1, 0].

From the above example, the sets $\psi(\mathcal{D}_1)$ and $\psi(\mathcal{D}_2)$ have 3 identical smallest possible representations and the estimated similarity will be 0.75, while the exact similarity is 0.5. The sets $\psi(\mathcal{D}_1)$ and $\psi(\mathcal{D}_3)$ share one smallest possible representation and their similarity estimate is 0.333 (0.375 is exact).

Finally, we introduce some important properties of OPH. Without loss of generality, we consider two documents $\mathcal{D}_1$ and $\mathcal{D}_2$. Firstly, it is the two fundamental definition of OPH $\mathcal{N}_{mb}$ and $\mathcal{N}_{eb}$. $\mathcal{N}_{mb}$ and and $\mathcal{N}_{eb}$ represents the number of "matched bins" and the number of "empty bins" respectively:

$$\mathcal{N}_{mb} = \sum_{j=1}^{k} \mathcal{B}_{mb,i} \tag{1}$$

$$\mathcal{N}_{eb} = \sum_{j=1}^{k} \mathcal{B}_{eb,i} \tag{2}$$

where $\mathcal{B}_{mb,i}$ and $\mathcal{B}_{eb,i}$ are defined for the i-th bin, as

$$\mathcal{B}_{mb,i} = \begin{cases} 1 \; if \; min(\psi(\mathcal{D}_1)) = min(\psi(\mathcal{D}_2)) \& \psi(\mathcal{D}_1) \neq \otimes \& \psi(\mathcal{D}_1) \neq \otimes \; in \\ \quad the \; i-th \; bin \\ 0 \; otherwise \end{cases} \tag{3}$$

$$\mathcal{B}_{eb,i} = \begin{cases} 1 \; if \; \psi(\mathcal{D}_1) = \psi(\mathcal{D}_2) = \otimes \; in \; the \; i-th \; bin \\ 0 \; otherwise \end{cases} \tag{4}$$

Denote ψ a random permutation: $\psi : \mathcal{V} \rightarrow \mathcal{V}$. The hashed values are the two smallest possible representation sets after applying the permutation ψ on $\psi(\mathcal{D}_1)$ and $\psi(\mathcal{D}_2)$. The probability at which the two hashed values are equal is

$$\mathcal{R} = \mathbf{Pr}(min(\psi(\mathcal{D}_1)) = min(\psi(\mathcal{D}_2))) = \frac{\mathcal{D}_1 \cap \mathcal{D}_2}{\mathcal{D}_1 \cup \mathcal{D}_2} = Jaccard(\mathcal{D}_1, \mathcal{D}_2) \tag{5}$$

Then the unbiased estimator of one permutation hashing is

$$\hat{\mathcal{R}}_{mb} = \frac{\mathcal{N}_{mb}}{k - \mathcal{N}_{eb}} \tag{6}$$

$$E(\hat{\mathcal{R}}_{mb}) = \mathcal{R} \tag{7}$$

Based on unbiased estimator, assume $g = |\mathcal{D}_1 \cup \mathcal{D}_2|$ the variance of OPH is

$$var(\hat{\mathcal{R}}_{mb}) = \mathcal{R}(1 - \mathcal{R})(E(\frac{1}{k - \mathcal{N}_{eb}})(1 + \frac{1}{g-1}) - \frac{1}{g-1}) \tag{8}$$

3.3 Group based one permutation hashing

Motivation While calculating the similarity, we have to compare the fingerprints one by one in one permutation hashing, which is time consuming. Meanwhile, one permutation hashing is claimed to satisfy the binomial distribution in [12]. Thus, based on this property, we design a new algorithm named group based one permutation hashing(GOPH) to reduce the comparison time.

Basic idea After applying one permutation to documents, we first aggregate the generated fingerprints into n-groups. Then, from the first group to the last group, we progressively compute the similarity between the corresponding groups, and bring in the concept of small probability event as a filter to accelerate the comparison for the remain groups.

Assume $\mathcal{K}$ is the bin number of the comparison part, $\mathcal{X}$ is defined as the number of times that the fingerprints are equal in comparison part, denotes as

$$\mathcal{X} = \sum_{j=0}^{\mathcal{K}} 1\{min(\psi(\mathcal{D}_i)) = min(\psi(\mathcal{D}_i))\} \tag{9}$$

Assume T is the estimator of the number of match bin after $\mathcal{K}$ comparisons. Because there are only two situations: "match" or "unmatch" in each comparison, these two situations are opposite each other and independent from each other, and the comparison result is not related to the results of other comparisons. Obviously, $\mathcal{X}$ satisfies the binomial distribution, denotes as $\mathcal{X} \sim F(\mathrm{T}, \mathcal{K})$. Thus, The distribution function $\mathcal{F}(x)$ of the variable $\mathcal{X}$ is denoted as follow:

$$\mathcal{F}(x) = \begin{cases} \sum_{j=0}^{\mathcal{X}} \binom{\mathcal{K}}{j} T^j (1-T)^{\mathcal{K}-j} & , x < \mathcal{X} \\ \sum_{j=\mathcal{X}+1}^{\mathcal{K}} \binom{\mathcal{K}}{j} T^j (1-T)^{\mathcal{K}-j}, & x \geq \mathcal{X} \end{cases} \tag{10}$$

In the following, we first introduce the concept of small probability event, then introduce how the small probability event is used to avoid unnecessary comparison for the remain groups.

Definition 1 (Small Probability Event) Given an error tolerance ϵ, $\mathcal{F}(x)$ is the distribution function of variable $\mathcal{X}$, and $\mathcal{X} \sim F(T, \mathcal{K})$, an event is called a small probability event, if and only if $\mathcal{F}(x) \leq \epsilon$.

Assume $\mathcal{M}_a$ indicates the excepted average match bins for each groups, and $\mathcal{M}_{r_a}$ represents the minimum average of the remainder groups, if the final average match bins not less than $\mathcal{M}_a$ according to the current binomial distribution. Assume the event E indicates the probability that $\mathcal{M}_{r_a}$ can eventually reach $\mathcal{M}_a$, if the probability of event E is less than the error tolerance ϵ, we can see event E as a small probability event, and terminate the subsequent calculation after corresponding processing. More specifically, while comparing the value of $\mathcal{M}_a$ and $\mathcal{M}_{r_a}$, we have the following two cases:

i) If $\mathcal{M}_{r_a} < \mathcal{M}_a$, for the remain groups, we compute whether $\mathcal{F}(\mathcal{M}_{r_a})$ is smaller than ϵ. If $\mathcal{F}(\mathcal{M}_{r_a})$ is smaller than ϵ, we consider that it is almost impossible for these two documents to meet the similarity requirement, label them as dissimilarity documents, and terminal the algorithm; otherwise, we takes next group into further consideration.

ii) If $\mathcal{M}_{r_a} \geq \mathcal{M}_a$, for the remain groups, we still compare the value of $\mathcal{F}(\mathcal{M}_{r_a})$ and ϵ. If $\mathcal{F}(\mathcal{M}_{r_a})$ is smaller than ϵ, we consider that these two documents should be a similar document pair, add them to result set, and terminal the algorithm; otherwise, we takes next group into further consideration.

Algorithm Algorithm 1 illustrates the implementation details of the document pair comparison based on GOPH. In Line 1,we first initialize current group number n_c to 1, current match bins $\mathcal{M}_c$ to 0, and calculate $\mathcal{M}_a$ by input value n, k', $\mathcal{T}$. Then, for each group of document pair $(\mathcal{D}_i, \mathcal{D}_j)$, we gradually calculate their similarity. From Line 3 to Line 8, we update the $\mathcal{M}_c$ based on current group's information. After computing the value of $\mathcal{M}_{r_a}$ in Line 9, we compare its value with $\mathcal{M}_a$. From Line 10 to Line 17, if $\mathcal{M}_{r_a} < \mathcal{M}_a$, we further compare $F(\mathcal{M}_{r_a})$ with ϵ, if its value is larger than ϵ, we continue to compare the similarity of next group; Otherwise, we consider current document pair is a similar pair, and break the algorithm. From Line 18 to Line 24, If $\mathcal{M}_{r_a} \geq \mathcal{M}_a$, and $F(\mathcal{M}_{r_a})$ is larger than ϵ, we continue to compare the similarity of next group; Otherwise, we break the algorithm and consider current document pair is not similar. Following is an example of GOPH comparison algorithm.

Algorithm 1 GOPH comparison

Input:
$(\mathcal{D}_i, \mathcal{D}_j)$: the document pair; $\mathcal{T}$: user preferred similarity threshold, k': bin number of each group;
n: group number of the document; ϵ: largest error tolerance; A small probability e;
Output:
$\mathcal{O}$: Similar document pairs

1: l=1; $\mathcal{M}_c$=0; $\mathcal{M}_a$=$k'\mathcal{T}$;
2: **for** each group of document pair $(\mathcal{D}_i, \mathcal{D}_j)$ **do**
3: **for** each bins in l-th group **do**
4: **if** the bin has equal value **then**
5: $\mathcal{M}_c$++;
6: **end if**
7: **end for**
8: l++;
9: $\mathcal{M}_{r_a} = \frac{nk'\mathcal{T} - \mathcal{M}_c}{n - l}$;
10: **if** $\mathcal{M}_{r_a} < \mathcal{M}_a$ **then**
11: **if** $F(\mathcal{M}_{r_a}) > \epsilon$ **then**
12: $Continue$;
13: **else**
14: $\mathcal{O} \leftarrow (\mathcal{D}_i, \mathcal{D}_j)$;
15: $Break$;
16: **end if**
17: **end if**
18: **if** $\mathcal{M}_{r_a} \geq \mathcal{M}_a$ **then**
19: **if** $F(\mathcal{M}_{r_a}) > \epsilon$ **then**
20: $Continue$;
21: **else**
22: $Break$;
23: **end if**
24: **end if**
25: **end for**

Table 2 Probability distribution of $x \leq \mathcal{X}$ and $x > \mathcal{X}$ when k'=100, n=10, $\mathcal{T}$=0.6

$\mathcal{X}$	$F(x < \mathcal{X})$	$F(x \geq \mathcal{X})$	$\mathcal{X}$	$F(x < \mathcal{X})$	$F(x \geq \mathcal{X})$
5	3.27948E-33	1	55	0.131090453	0.868909547
10	1.25639E-26	1	60	0.456705514	0.543294486
15	2.31928E-21	1	65	0.820530647	0.179469353
20	5.56419E-17	1	70	0.975217177	0.024782823
25	2.71442E-13	1	75	0.998810999	0.001189001
30	3.46422E-10	1	80	0.999983588	1.64119E-05
35	1.35466E-07	0.999999865	85	0.999999949	5.0732E-08
40	1.80415E-05	0.999981959	90	1	2.33876E-11
45	0.000881808	0.999118192	95	1	7.01609E-16
50	0.016761687	0.983238313	100	1	6.53319E-23

Example 1 Given a document pair $(\mathcal{D}_i, \mathcal{D}_j)$, assume k'=100, n=10, $\mathcal{T}$=0.6, $\epsilon = 10^{-4}$ and $\mathcal{X} \sim F(\mathcal{T}, \mathcal{K})$. Then, the values of $F(\mathcal{X})$ for different $\mathcal{X}$ are shown in Table 2 and the value of $\mathcal{M}_a$ is 60. Assume there are 65 bins matching in the 1-st comparison, the value of $\mathcal{M}_{r_a}$ is 59.4. Because $\mathcal{M}_{r_a} < \mathcal{M}_a$, and $F(\mathcal{M}_{r_a})$ is larger than ϵ, we continue to compare next group. Assume only 5 bins matching in the 2-nd comparison, the value of $\mathcal{M}_{r_a}$ is 66.25. Because $\mathcal{M}_{r_a} \geq \mathcal{M}_a$, and $F(\mathcal{M}_{r_a})$ is larger than ϵ, we continue to compare next group. Assume in the 3-rd comparison, there are 10 bins matching, similarly, the value of $\mathcal{M}_{r_a}$ is 74.28. Because $\mathcal{M}_{r_a} \geq \mathcal{M}_a$, and $F(\mathcal{M}_{r_a})$ is larger than ϵ, we continue to compare next group. In the 4-th comparison, assume there are 10 bins matching, the value of $\mathcal{M}_{r_a}$ is 85. Obviously, $\mathcal{M}_{r_a} \geq \mathcal{M}_a$, we continue to compare the value of $F(x \geq 85)$ and ϵ. Because the probability of $F(x \geq 85)$ equals 5.0732E-08, which is smaller than ϵ, it means the probability that the document pair $(\mathcal{D}_1, \mathcal{D}_2)$ is a similar pair is a small probability event. Thus, we terminal the algorithm and consider current document pair is not similar pair.

3.4 GOPH for image near-duplicate detection

In this subsection, we describe how we extend the GOPH method originally developed for text near-duplicate detection to image near-duplicate detection. We describe it using visual words to replace visual words in the following sub-section.

Two images are near duplicate if the similarity **sim**$(\mathcal{I}_1, \mathcal{I}_2)$ is higher than a given similarity threshold $\mathcal{T}$. The goal is to retrieve all images in the database that are similar to a query image. The outline of the images are near duplicate detection algorithm is as follows: First a list of visual words are extracted from each image. A visual word is a single number having the property that two images $\mathcal{I}_1, \mathcal{I}_2$ have the same value of visual word with probability equal to their similarity **sim**$(\mathcal{I}_1, \mathcal{I}_2)$. To efficient compare the visual words, a one permutation ψ is used to evenly divide these visual words into k-bins and retrieval smallest possible representations as fingerprints. Then, we adopt GOPH algorithm to compute the similarity between these two fingerprint sets.

How does it work? Consider image $\mathcal{Y} = \arg\min \psi(\mathcal{I}_i \cup \mathcal{I}_j)$. Since ψ is an one permutation, each fingerprint of $\mathcal{I}_i \cup \mathcal{I}_j$ has the same probability of being the smallest possible fingerprint. Hence, $\mathcal{Y}$ can be constructed from $\mathcal{I}_i \cup \mathcal{I}_j$. If $\mathcal{Y}$ is an fingerprint of both $\mathcal{I}_i$ and $\mathcal{I}_j$, i.e. $\mathcal{Y} \subseteq \mathcal{I}_i \cap \mathcal{I}_j$, then $\min \psi(\mathcal{I}_i) = \psi(\mathcal{I}_j) = \psi(\mathcal{Y})$. Otherwise, if $\mathcal{Y} \subseteq \mathcal{I}_i \setminus \mathcal{I}_j$, then $\psi(\mathcal{Y}) < \psi(\mathcal{I}_j)$; if $\mathcal{Y} \subseteq \mathcal{I}_j \setminus \mathcal{I}_i$, then $\psi(\mathcal{Y}) < \psi(\mathcal{I}_i)$. Thus, for an one permutation ψ it follows

$$\mathbf{Pr}(min(\psi(\mathcal{I}_i)) = min(\psi(\mathcal{I}_i))) = \frac{\mathcal{I}_i \cap \mathcal{I}_j}{\mathcal{I}_i \cup \mathcal{I}_j} \tag{11}$$

To enhance the efficiency of comparison, the fingerprints are grouped into m-tuples. Similar to text comparison, from the first summary to the last summary, we gradually compute the similarity between the corresponding summaries, and estimate whether the remain summaries will meet the small probability event or not. If the remain summaries meets the small probability event, the algorithm terminals; Otherwise, we continue to calculate the fingerprints of the next summary.

4 Our strategy: hierarchical one permutation hashing

This section presents the Hierarchical One Permutation Hash (HOPH) for efficient multimodal near duplicate detection. Section 4.2 first explains the basic idea of HOPH. Then, Section 4.2 presents some theoretical analysis of HOPH, and Section 4.3 discuss the image implementation of HOPH.

4.1 Hierarchical one permutation hashing overview

Different with traditional OPH, which evenly divided the whole space $\mathcal{V}$ into k buckets, HOPH scheme is first grouping the original data entries into two groups, namely permutation group and division group, in each iteration. The space of each permutation group is evenly into k' parts, while the space of each division group is further divide into permutation group and division group sequentially, if the sub group size is greater than k'. More specifically, assume $a + b = 1$, HOPH divide the space into two groups with proportion $a : b$, namely (a:b)HOPH. Thus, after first iteration, the the whole space $\mathcal{V}$ is dividing into two groups $\mathcal{G}_1$ and $\mathcal{G}_2$ with size $\frac{a}{a+b}\mathcal{V}$, $\frac{b}{a+b}\mathcal{V}$ respectively. Then, we divide the space evenly into k' parts for the first group $\mathcal{G}_1$. For the second group $\mathcal{G}_2$, we check whether the number of data entries of its sub groups are larger than k' after division, if all the number is larger than k', then HOPH continue to divide the second group into two parts with $a : b$; Otherwise, HOPH terminal the division. Figure 2 shows the example of (1:1)HOPH.

				1				2				3				
	1				2				3		4		5		6	
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
$\psi(\mathcal{D}_1)$:	0	1	1	0	0	1	0	0	0	0	1	0	1	0	0	1
$\psi(\mathcal{D}_2)$:	0	1	1	0	0	0	1	0	0	0	1	0	1	0	1	0
$\psi(\mathcal{D}_3)$:	0	0	1	0	0	0	0	0	0	1	1	0	1	0	1	0

Fig. 2 Hierarchical one permutation hashing example

Similar to OPH, a random permutation ψ is generated firstly. For each document $\mathcal{D}_i$ a one permutation hashing min $\psi(\mathcal{D}_i)$ is recorded. Consider $\mathcal{D}_1, \mathcal{D}_2, \mathcal{D}_3 \subseteq \mathcal{V} = \{0, 1, ..., 16\}$. Assume $\mathcal{D}_1 = \{1, 2, 5, 10, 12, 15\}$, $\mathcal{D}_2 = \{1, 2, 6, 10, 12, 14\}$, $\mathcal{D}_3 = \{2, 9, 10, 12, 14\}$, and $k' = 2$ and $a : b = 1 : 1$. Similar to OPH, a one permutation ψ is generated firstly. Then, we divide the space with (1:1)HOPH into three groups. After that, we apply one permutation ψ on the three groups and evenly divide the space into k' buckets for each group, select the smallest nonzero in each bucket as samples. For example, [[1, 5], [⊗, 10], [12, 15]], [[1, 6], [⊗, 10], [12, 14]], and [[2, ⊗], [9, 10], [12, 14]]. We use ⊗ to denote an empty bin, which occur rarely while the number of nonzeros is large compared to k'.

In the example in Fig. 2 (which includes 3 documents), the sample selected from $\psi(\mathcal{D}_1)$ is [[1, 5], [⊗, 10], [12, 15]]. We re-index the elements of each bucket to use the smallest possible representations, because only elements with the same bin number need to be compared. For example, for $\psi(\mathcal{D}_1)$, after re-indexing, the sample [[1, 5], [⊗, 10], [12, 15]] becomes [[1, 1], [⊗, 0], [0, 1]]. Similarly, for $\psi(\mathcal{D}_2)$, the original sample [[1, 6], [⊗, 10], [12, 14]] becomes [[1, 2], [⊗, 0], [0, 0]], etc.

From the above example, the sets $\psi(\mathcal{D}_1)$ and $\psi(\mathcal{D}_2)$ have one identical smallest possible representations in each subgroup and the estimated similarity will be 0.625, while the exact similarity is 0.5. The sets $\psi(\mathcal{D}_1)$ and $\psi(\mathcal{D}_3)$ share one smallest possible representation and their similarity estimate is 0.375 (0.375 is exact).

Algorithm Algorithm 2 illustrates the implementation details of the (a:b)HOPH Comparison for the input document pair. For presentation simplicity, we assume there are l groups in each document, $r = \frac{a}{a+b}$, $\mathcal{P}_a$ represents the average matched probability should be achieve, $\mathcal{P}_r$ represents the excepted average matched probability for the groups haven't been compared, and an array $\mathcal{M}$ is used to store the number of matched bin for each group in Line 1. For each group of document pair $(\mathcal{D}_i, \mathcal{D}_j)$, we gradually calculate their similarity. From Line 3 to Line 8, we update the $\mathcal{M}_l$ based on the l-th group's information. Then, we compute the value of P_r in Line 11, we compare it with $\mathcal{P}_a$. From Line 12 to Line 19, if $P_r < P_a$, we further compare $F(P_r * k')$ with ϵ, if its value is larger than ϵ, we continue to compare the similarity of next group; Otherwise, we consider current document pair is a similar pair, and break the algorithm. From Line 20 to Line 26, If $P_r \geq P_a$, and $F(P_r * k')$ is larger than ϵ, we continue to compare the similarity of next group; Otherwise, we break the algorithm and consider current document pair is not similar. Following is an example of HOPH comparison algorithm.

Example 2 Given a document pair $(\mathcal{D}_i, \mathcal{D}_j)$, assume k'=100, n=10, $\mathcal{T}$=0.6, $\epsilon = 10^{-4}$ and $\mathcal{X} \sim F(\mathcal{T}, \mathcal{K})$. Then, the values of $F(\mathcal{X})$ for different $\mathcal{X}$ are shown in Table 2 and the value of $\mathcal{M}_a$ is 60. Assume there are only 40 bins matching in the 1-st comparison, then the value of P_r is 0.8. Because $P_r \geq P_a$, we further compare the value of $F(P_r * k')$ and ϵ. Because $P_r * k'$ is 80, and the probability of $F(x \geq 80)$ equals 1.64119E-05, which is smaller than ϵ. It indicates the event that the document pair $(\mathcal{D}_1, \mathcal{D}_2)$ is a similar pair is a small probability event. Thus, we terminal the algorithm and consider current pair is not similar pair.

In Algorithm 2, line 3-8 computes the number of the bins which have equal value, and the cost is $O(k')$. Line 11-25 searches the similar document pairs by probability Pr and $F(Pr * k')$, the cost is $O(1)$. Assume $F(Pr * k') = \epsilon$, the inverse function of F is denoted as F^{-1}, thus $Pr = \frac{F^{-1}(\epsilon)}{k'}$. According to the aforementioned theories, $Pr = \frac{P - \frac{\mathcal{M}_l}{k'} * r^l}{r^l} =$

$\frac{P}{r^l} - \frac{\mathcal{M}_l}{k'}$, we can compute $l = \log_r \frac{P*k'}{F^{-1}(\epsilon)+\mathcal{M}_l}$. Therefore, the total cost of Algorithm 2 is $O(k' * \log_r \frac{P*k'}{F^{-1}(\epsilon)+\mathcal{M}_l})$.

Algorithm 2 (a:b)HOPH comparison

Input:
$(\mathcal{D}_i, \mathcal{D}_j)$: the input document pair; $\mathcal{T}$: user preferred similarity threshold, k': bin number of each group; $\mathcal{M}$: array to store the number of matched bin ; ϵ: error tolerance;
Output:
$\mathcal{O}$: Similar document pairs

1: l=0; $\mathcal{M} \leftarrow \{0\}$; $\mathcal{P}$=$\mathcal{P}_a$=$\mathcal{T}$ $P_r = 0$; $r = \frac{a}{a+b}$;
2: **for** each l-th group of document pair $(\mathcal{D}_i, \mathcal{D}_j)$ **do**
3: **for** each bins in l-th group **do**
4: $\mathcal{M}_l$=0;
5: **if** the bin has equal value **then**
6: $\mathcal{M}_l$++;
7: **end if**
8: **end for**
9: l++;
10: $\mathcal{P}_r = \frac{\mathcal{P} - \frac{\mathcal{M}_l}{k'} r^l}{r^l}$;
11: $\mathcal{P} = \mathcal{P}_r$;
12: **if** $\mathcal{P}_r < \mathcal{P}_a$ **then**
13: **if** $F(\mathcal{P}_r * k') > \epsilon$ **then**
14: $Continue$;
15: **else**
16: $\mathcal{O} \leftarrow (\mathcal{D}_i, \mathcal{D}_j)$;
17: $Break$;
18: **end if**
19: **end if**
20: **if** $\mathcal{P}_r \geq \mathcal{P}_a$ **then**
21: **if** $F(P_r * k') > \epsilon$ **then**
22: $Continue$;
23: **else**
24: $Break$;
25: **end if**
26: **end if**
27: **end for**

4.2 Theoretical analysis of HOPH

In the following section, we will introduce some interesting theoretical analysis of HOPH, such as the number of match bin, the number of empty bin, the unbiased estimator and so on.

Assume $a + b = 1, r = \frac{a}{a+b}$, $(a : b)$HOPH first divides the whole space $\mathcal{V}$ into $l + 1$ group with length $r\mathcal{V}$, $r^2\mathcal{V} \cdots r^l\mathcal{V}$ and $r^{l-1}(1 - r)\mathcal{V}$ respectively, then divide the

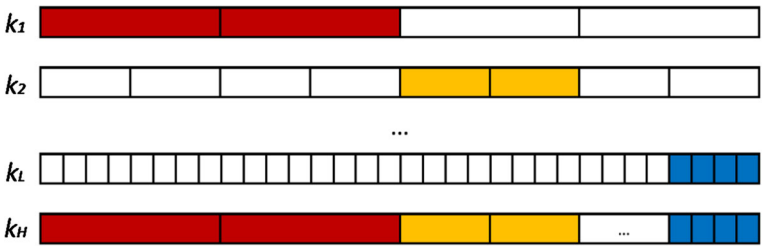


Fig. 3 Hierarchical one permutation hashing construction

corresponding space evenly into k' bins, as what is shown in Fig. 3. Based on the above assumption, $(a : b)$HOPH has the following properties:

Lemma 1 *The number of matched bin of* $(a : b)$*HOPH is*

$$\mathcal{N}_{mb_h} = (\sum_{j=1}^{l-1} r^{2j}\mathcal{N}_{mb_j} + r^{2l-1}\mathcal{N}_{mb_l})(l+1)$$

Proof As shown in Fig. 3, assume $(a : b)$HOPH consists of $l+1$ groups from l different one permutation hashes, which are evenly divided into $k_1, k_2, \cdots, k_l$ respectively.

$$\begin{aligned}
P(\mathcal{B}_{mb,i} = 1, i \in [1,\ k_h]) &= P(\mathcal{B}_{mb,i} = 1, i \in [1, rk_1]) \\
&+ P(\mathcal{B}_{mb,i} = 1, i \in [rk_1 + 1, rk_1 + r^2k_2]) + \cdots \\
&+ P(\mathcal{B}_{mb,i} = 1, i \in [\sum_{j=1}^{l-1} r^j k_j + 1, \sum_{j=1}^{l} r^j k_j]) \\
&+ P(\mathcal{B}_{mb,i} = 1, i \in [\sum_{j=1}^{l} r^j k_j] + 1, \sum_{j=1}^{l} r^j k_j + r^{l-1}(1-r)k_l]) \\
&= r\frac{\mathcal{N}_{mb_1}}{k_1} + r^2\frac{\mathcal{N}_{mb_2}}{k_2} + \cdots + r^l\frac{\mathcal{N}_{mb_l}}{k_l} + r^{l-1}(1-r)\frac{\mathcal{N}_{mb_l}}{k_l}
\end{aligned}$$

Since,

$$P(\mathcal{B}_{mb,i} = 1, i \in [1, k_h]) = \frac{\mathcal{N}_{mb_h}}{k_h}$$

Then,

$$\frac{\mathcal{N}_{mb_h}}{k_h} = r\frac{\mathcal{N}_{mb_1}}{k_1} + r^2\frac{\mathcal{N}_{mb_2}}{k_2} + \cdots + r^{l-1}\frac{\mathcal{N}_{mb_{l-1}}}{k_{l-1}} + r^{l-1}\frac{\mathcal{N}_{mb_l}}{k_l}$$

Since,

$$k_h = (l+1)k'; k_j = \frac{k'}{r^j}$$

We obtain,

$$\begin{aligned}
\mathcal{N}_{mb_h} &= (r^2\frac{\mathcal{N}_{mb_1}}{k'} + r^4\frac{\mathcal{N}_{mb_2}}{k'} + \cdots + r^{2(l-1)}\frac{\mathcal{N}_{mb_{l-1}}}{k'} + r^{2l-1}\frac{\mathcal{N}_{mb_l}}{k'})(l+1)k' \\
&= (r^2\mathcal{N}_{mb_1} + r^4\mathcal{N}_{mb_2} + \cdots + r^{2(l-1)}\mathcal{N}_{mb_{l-1}} + r^{2l-1}\mathcal{N}_{mb_l})(l+1) \\
&= (\sum_{j=1}^{l-1} r^{2j}\mathcal{N}_{mb_j} + r^{2l-1}\mathcal{N}_{mb_l})(l+1)
\end{aligned}$$

thus completing the proof. □

Lemma 2 *The number of empty bin of* $(a : b)$*HOPH is*

$$\mathcal{N}_{eb_h} = (\sum_{j=1}^{l-1} r^{2j} \mathcal{N}_{eb_j} + r^{2l-1} \mathcal{N}_{eb_l})(l+1)$$

Proof Similar to Lemma 1, we can obtain

$$P(\mathcal{B}_{eb,i} = 1, i \in [1, k_h]) = r\frac{\mathcal{N}_{eb_1}}{k_1} + r^2\frac{\mathcal{N}_{eb_2}}{k_2} + \cdots + r^l\frac{\mathcal{N}_{eb_l}}{k_l} + r^{l-1}(1-r)\frac{\mathcal{N}_{eb_l}}{k_l}$$

Since,

$$P(\mathcal{B}_{eb,i} = 1, i \in [1, k_h]) = \frac{\mathcal{N}_{eb_h}}{k_h}; k_h = (l+1)k'; k_j = \frac{k'}{r^j}$$

We obtain,

$$\begin{aligned}\mathcal{N}_{eb_h} &= (r^2\frac{\mathcal{N}_{eb_1}}{k'} + r^4\frac{\mathcal{N}_{eb_2}}{k'} + \cdots + r^{2(l-1)}\frac{\mathcal{N}_{eb_{l-1}}}{k'} + r^{2l-1}\frac{\mathcal{N}_{eb_l}}{k'})(l+1)k' \\ &= (\sum_{j=1}^{l-1} r^{2j}\mathcal{N}_{eb_j} + r^{2l-1}\mathcal{N}_{eb_l})(l+1)\end{aligned}$$

thus completing the proof. □

Lemma 3 *The unbiased estimator of* $(a : b)$*HOPH is*

$$\hat{\mathcal{R}}_{mb_h} = \sum_{j=1}^{l-1} r^j \frac{\mathcal{N}_{mb_j}}{k_j - \mathcal{N}_{eb_j}} + r^{l-1}\frac{\mathcal{N}_{mb_l}}{k_l - \mathcal{N}_{eb_l}}$$

Proof Similar to Lemma 1, we can obtain

$$\begin{aligned}\hat{\mathcal{R}}_{mb_h} &= r\hat{\mathcal{R}}_{mb_1} + r^2\hat{\mathcal{R}}_{mb_2} + \cdots + r^l\hat{\mathcal{R}}_{mb_l} + r^{l-1}(1-r)\hat{\mathcal{R}}_{mb_l} \\ &= \sum_{j=1}^{l-1} r^j\hat{\mathcal{R}}_{mb_j} + r^{l-1}\hat{\mathcal{R}}_{mb_l}\end{aligned}$$

Since,

$$\hat{\mathcal{R}}_{mb_j} = \frac{\mathcal{N}_{mb_j}}{k_j - \mathcal{N}_{eb_j}}$$

We obtain,

$$\hat{\mathcal{R}}_{mb_h} = \sum_{j=1}^{l-1} r^j \frac{\mathcal{N}_{mb_j}}{k_j - \mathcal{N}_{eb_j}} + r^{l-1}\frac{\mathcal{N}_{mb_l}}{k_l - \mathcal{N}_{eb_l}}$$

thus completing the proof. □

4.3 HOPH for image near-duplicate detection

In term of image near duplicate detection, the construction and comparison method of HOPH is similar to that of GOPH. Given a image collection, we first extract the visual word list from each image. In construction stage, we first generate a random permutation ψ, then apply HOPH scheme recursively divide the whole visual word space two groups in each iteration. For the front group, we apply permutation ψ to it, and evenly divide the space into k' buckets. For the latter group, we further divide the space into two groups again, if its

sub group size is greater than k'. In comparison stage, given two HOPH group, we gradually compute the similarity between the corresponding group from the first to the last, and estimate whether the remain part will meet the small probability event or not. If the remain part will trigger the small probability event, the algorithm terminals and outputs the result; Otherwise, the fingerprints of the next group will be calculated for further evaluation.

5 Performance evaluation

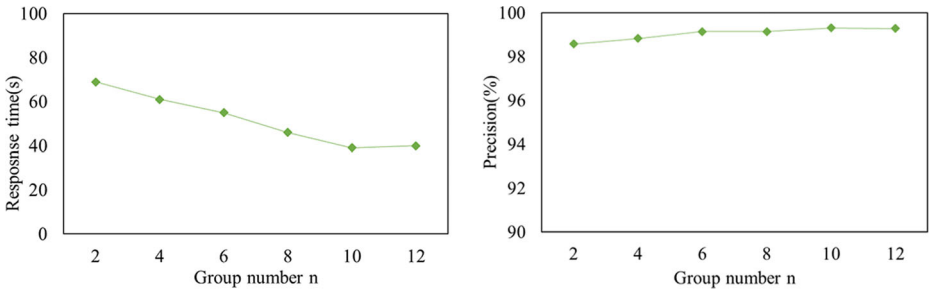
In this section, we present results of a comprehensive performance study to evaluate the efficiency and scalability of the proposed techniques in the paper. In our implementation, we evaluate the effectiveness of the following Hashing techniques.

- MinWise. Minwish hashing, which is a natural implementation of the method in [3].
- OPH. One permutation hashing, which is a natural implementation of the method in [12].
- GOPH. The group based one permutation hashing technique proposed in Section 3.3.
- (1:1)HOPH. The hierarchical one permutation hashing whose ratio of a to b equals 1:1.
- (1:2)HOPH. The hierarchical one permutation hashing whose ratio of a to b equals 1:2.
- (2:1)HOPH. The hierarchical one permutation hashing whose ratio of a to b equals 2:1.

Environment settings Experiments are run on a PC with Intel i7 6700HQ 2.60GHz CPU and 16G memory running Ubuntu 16.04 LTS. All algorithms in the experiments are implemented in Java.

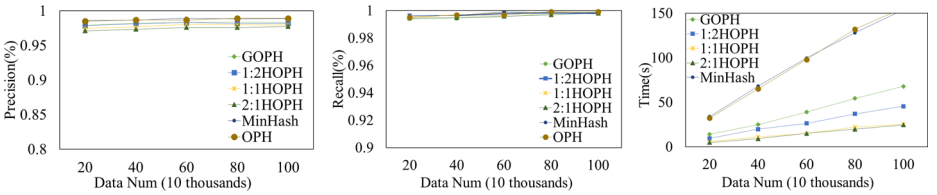
Workload A workload for this experiment consists of 100 input queries, and the precision, recall and response time are employed to evaluate the performance of the algorithms. By default, we set the error tolerance $e = 10^{-4}$, user preferred similarity threshold $T = 0.7$, data number $V = 60 * 10^4$.

Performance matric The objective evaluation of the proposed approach is carried out based on precision and recall. Precision measures the accuracy of the retrieval. It is the ratio of retrieved documents that are similar to the query.



(a) Response time (b) Precision

Fig. 4 Effect of the training group number on FS



(a) Precision (b) Recall (c) Response time

Fig. 5 Effect of different data number on FS

Precision is the ratio of retrieved images that are relevant to the query image.

$$Precision = \frac{number\ \ of\ \ similar\ \ documents\ \ retrieved}{Total\ \ number\ \ of\ \ documents\ \ retrieved}$$

Recall measures the robustness of the retrieval. It is defined as the ratio of relevant images in the database that are retrieved in response to a query.

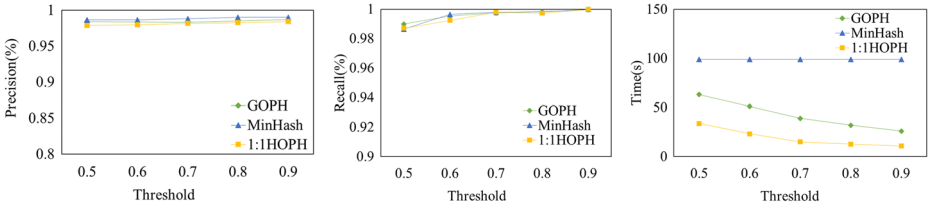
$$Recall = \frac{number\ \ of\ \ similar\ \ documents\ \ retrieved}{Total\ \ number\ \ of\ \ similar\ \ documents\ \ in\ \ dataset}$$

5.1 Evaluation on text dataset (FS)

Dataset Performance of various algorithms are evaluated on real dataset FundSet(FS). FS is obtained from the large-scale document database of NSFC in which each document is a NSFC proposal in PDF format. Taking some documents of funds proposal as the data source.

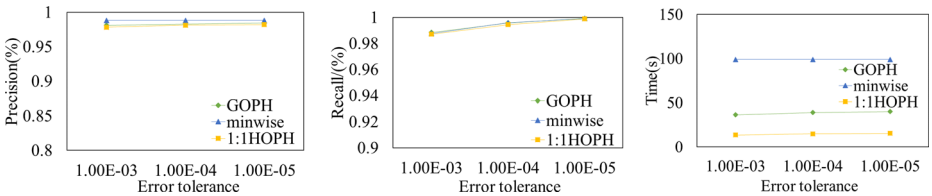
Evaluating training group number At first, we try to train the group number of text document n. Figure 4a shows that with n increasing, response time is reduced. But the downtrend slows down and at $n = 10$ the response time is minimum. It indicates that the performance cannot be boosted all along with the gradually increasing of n. On the other hand, as shown in Fig. 4b, no matter what value n is, the precision is very high, nearly 100%. Thus, we choose the $n = 10$ as the default value of GOPH.

Evaluation on different data number We investigate the response time, precision and Recall in Fig. 5 against the dataset FS, where other parameters are set to default values. Figure 5a depicts the accuracy of GOPH, 1:2HOPH, 1:1HOPH, 2:1HOPH, MinHash and OPH. Obviously, MinHash and OPH have the highest precision. When Data Num is larger



(a) Precision (b) Recall (c) Response time

Fig. 6 Effect of different threshold on FS

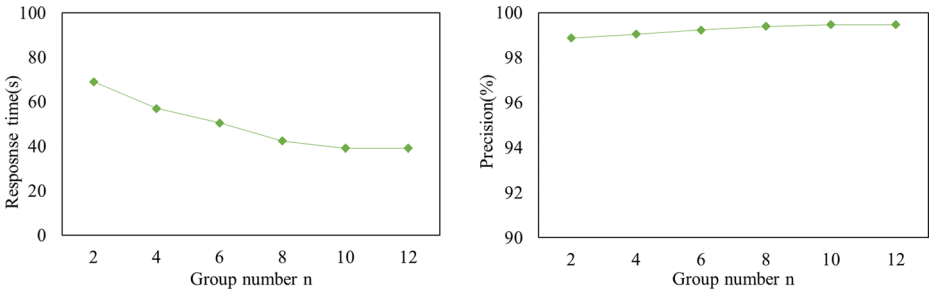


(a) Precision (b) Recall (c) Response time

Fig. 7 Effect of different small probability on FS

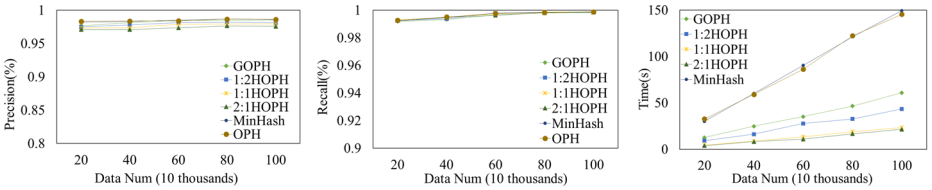
than $60 * 10^4$, the growth of precision slows down. Figure 5b demonstrates that the recall of them all are climbing with the increasing of Data Num, and there is little difference between them. Figure 5c shows that with the increasing of the Data Num, the response time of these 6 algorithms gradually rise. At Data Num $= 20 * 10^4$, all of the values are in the range between 150 and 250. But when Data Num increases to 100, we can see that the performance of 1:1HOPH and 2:1HOPH are much better than MinHash and OPH. Particularly, 2:1HOPH has the smallest response time among the algorithms, because the number of filter segments is the most in 2:1HOPH. On the other hand, 1:1HOPH is most suitable because of the equilibrium of precision, recall and speed. As above evaluation shown, in the aspect of efficiency 1:1HOPH is higher than 1:2HOPH. Besides, the response time of 1:1HOPH and 2:1HOPH are almost the same but the precision of 1:1HOPH is higher than the other. Meanwhile, as the accuracy of GOPH is close to OPH and its response time is two or three times faster than OPH, we select GOPH and 1:1HOPH to conduct the following comparison evaluation.

Evaluation on different threshold Fig. 6a depicts that with the rising of the threshold, the precision of MinHash, GOPH and 1:1HOPH slowly increases. All of them are larger than 98% when the threshold is larger 0.7. Apparently, the precision of MinHash is little higher than the precision of 1:1HOPH. Figure 6b illustrates that the three recalls stay the same tendency, they are near 100% when the threshold T is larger than 0.8. As shown in Fig. 6c, GOPH, MinHash and 1:1HOPH demonstrate superior performance in comparison with MinHash and the response time of GOPH and 1:1HOPH decline with the Threshold climbing. Obviously, compared with MinHash and GOPH, 1:1HOPH has the smallest response time.



(a) Response time (b) Precision

Fig. 8 Effect of the training group number on IS



(a) Precision (b) Recall (c) Response time

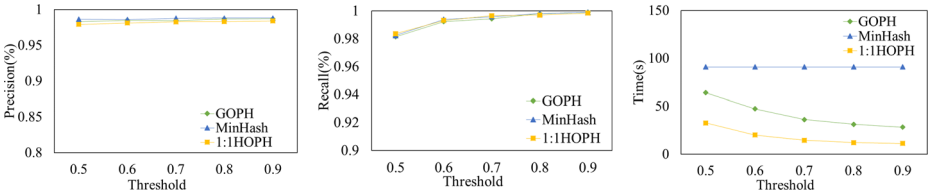
Fig. 9 Effect of different data number on IS

Evaluation on small probability Fig. 7a reports the precision of Minhash, GOPH and 1:1HOPH. Clearly, the precision of MinHash is invariable and the precision of GOPH and 1:1HOPH grow slowly. All of them are very high, MinHash is nearly 99% and two others are larger than 98%. Figure 7b demonstrates the recall of GOPH, MinHash and 1:1HOPH. It is easy to find there is little difference in recall. All of them gradually increase and are nearly 100% when the error tolerance equals 1.00E-05. In Fig. 7c, the performance of 1:1HOPH is nearly 5 times higher than MinHash. When we change the error tolerance to a smaller value, the climbing of the performance of GOPH and 1:1HOPH is not obvious.

5.2 Evaluation on image dataset (IS)

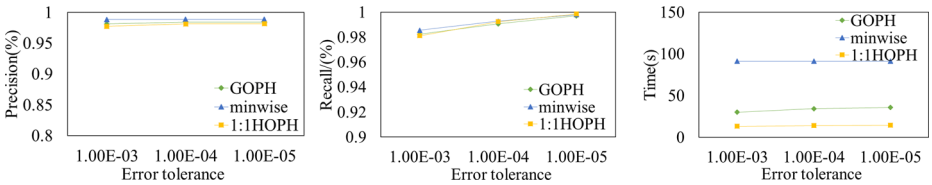
Dataset Our empirical studies aim to evaluate the performance of the filter against a subset of ImageNet datast. ImageNet is the largest image dataset for image processing and computer vision. It is organized according to the WordNet hierarchy (currently only the nouns), in which each node of the hierarchy is depicted by hundreds and thousands of images. This dataset includes: (1)14,197,122 images; (2)1,034,908 images with bounding box annotations; (3)1000 synsets with SIFT features; (4)1.2 million images with SIFT features. We generate a image dataset (IS) by selecting 1 million images from ImageNet. On IS, we evaluate the precision and efficiency of GOPH, 1:2HOPH, 1:1HOPH, 2:1HOPH, MinHash and OPH.

Evaluating training group number Firstly, we try to train the group number n. It is illustrated by Fig. 8a that with the raising of n, response time is going down gradually. But the downtrend slows down and at $n = 10$ the response time is minimum. The performance cannot be boosted all along with n gradually increasing. On the other hand, as shown in Fig. 8b, with the raising of Data Num, the precision increases step by step with fluctuations, and at $n = 10$, it is nearly 99.4%. Hence we choose the $n = 10$ as the default value of GOPH.



(a) Precision (b) Recall (c) Response time

Fig. 10 Effect of different threshold on IS



(a) Precision (b) Recall (c) Response time

Fig. 11 Effect of different small probability on IS

Evaluation on different data number We evaluate the precision, recall and response time of these 6 algorithms against IS. The precision is shown in Fig. 9a. apparently, all the precisions fluctuate in the range of 97.1% to 98.7% with the increasing of Data Num and the precision of MinHash and OPH are higher than others. It is obvious that in Fig. 9b there is litter difference in recall over these 6 algorithms and all of them are approximate 100% with the number of data increasing. Figure 9c demonstrates the trends of response time of MinHash and OPH are almost the same, much higher than the others. Particularly, 2:1HOPH and 1:1HOPH significantly outperform the other 4 algorithms in performance. It is clear that 1:1HOPH dominates to 2:1HOPH in the aspect of precision but the efficiency of the former is not lower than the the latter. On the other hand, the response time of 1:2HOPH is higher than 1:1HOPH. Furthermore, the efficiency of GOPH is much higher than OPH. Therefore, in the evaluation on different threshold and small PR, we compare the two algorithms mentioned-above and MinHash.

Evaluation on different threshold The precision of GOPH, MinHash and 1:1HOPH on difference threshold are shown in Fig. 10a. All of the precisions fluctuate in the interval of 97.9% to 98.7% The precision of MinHash is little higher than 1:1HOPH and GOPH. As shown in Fig. 10b, the recall of these algorithms stay the same tendency. They ascend gradually when the Threshold increases from 0.5 to 0.7. Figure 10c tells us that the response time of GOPH and 1:1HOPH decline step by step but the performance of MinHash remains unchanged. As expected, 1:1HOPH has the best performance among them. When the Threshold is smaller than 0.8, the response time of 1:1HOPH is less than 13s.

Evaluation on small probability In Fig. 11a, we evaluate the precision of GOPH, MinHash and 1:1HOPH. It is no doubt that the precision of MinHash stay a very high value, a little higher than the others which slowly raise with the error tolerance increasing from 1.00E-3 to 1.00E-4. After that they are almost invariable. On the other hand, as shown in Fig. 11b the recall of these algorithms go up moderately and at error tolerance is 1.00E-5 they are nearly 100%. We can see from Fig. 11c that, with the changing of error tolerance, the performance of MinHash remains the same but the others changed very smoothly. Like the situation on dataset FS, the performance of 1:1HOPH is much better than two others.

6 Conclusion

The problem of multimedia near duplicate detection is important due to the increasing amount of multimedia data collected in a wide spectrum of applications. In the paper, we propose introduce OPH to reduce the costly preprocessing time. Based on OPH, we propose GOPH to accelerate the comparison speed. Then, we design a novel hashing method namely

HOPH to further improve the performance. Both GOPH and HOPH can easily extend to image near duplicate detection. Finally, our comprehensive experiments convincingly demonstrate the efficiency of our techniques.

Acknowledgments This work was supported in part by the National Natural Science Foundation of China (61379110, 61472450, 61702560), the Key Research Program of Hunan Province(2016JC2018), project (2016JC2011, 2018JJ3691) of Science and Technology Plan of Hunan Province, and Fundamental Research Funds for Central Universities of Central South University (2018zzts588).

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, WWW 2007, Banff, Alberta, Canada, May 8-12, 2007, pp 131–140
2. Broder AZ, Glassman SC, Manasse MS, Zweig G (1997) Syntactic clustering of the web. Comput Netw 29(8-13):1157–1166
3. Broder AZ, Charikar M, Frieze AM, Mitzenmacher M (2000) Min-wise independent permutations. J Comput Syst Sci 60(3):630–659
4. Chum O, Philbin J, Isard M, Zisserman A (2007) Scalable near identical image and shot detection. In: Proceedings of the 6th ACM International Conference on Image and Video Retrieval, CIVR 2007, Amsterdam, The Netherlands, July 9-11, 2007, pp 549–556
5. Chum O, Philbin J, Zisserman A (2008) Near duplicate image detection: min-hash and tf-idf weighting. In: Proceedings of the British Machine Vision Conference 2008, Leeds, September 2008, pp 1–10
6. Hassanian-esfahani R, Kargar MJ (2018) Sectional minhash for near-duplicate detection. Expert Syst Appl 99:203–212
7. Henzinger MR (2006) Finding near-duplicate web pages: a large-scale evaluation of algorithms. In: SIGIR 2006: Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, Seattle, Washington, USA, August 6-11, 2006, pp 284–291
8. Hoad TC, Zobel J (2003) Methods for identifying versioned and plagiarized documents. JASIST 54(3):203–215
9. Indyk P, Motwani R (1998) Approximate nearest neighbors: Towards removing the curse of dimensionality. In: Proceedings of the Thirtieth Annual ACM Symposium on the Theory of Computing, Dallas, Texas, USA, May 23-26, 1998, pp 604–613
10. Jain P, Kulis B, Grauman K (2008) Fast image search for learned metrics. In: 2008 IEEE Computer society conference on computer vision and pattern recognition (CVPR 2008), 24-26 June 2008, Anchorage, Alaska, USA
11. Li P, Shrivastava A, Moore JL, König AC (2011) Hashing algorithms for large-scale learning. In: Advances in Neural Information Processing Systems 24: 25th Annual Conference on Neural Information Processing Systems 2011. Proceedings of a meeting held 12-14 December 2011, Granada, Spain, pp 2672–2680
12. Li P, Owen A, Zhang CH (2012) One permutation hashing for efficient search and learning. Mathematics
13. Li P, Wang M, Cheng J, Xu C, Lu H (2013) Spectral hashing with semantically consistent graph for image indexing. IEEE Trans Multimed 15(1):141–152
14. Lowe DG (1999) Object recognition from local scale-invariant features. In: ICCV, pp 1150–1157
15. Lowe DG (2004) Distinctive image features from scale-invariant keypoints. Int J Comput Vis 60(2):91–110
16. Nistér D, Stewénius H (2006) Scalable recognition with a vocabulary tree. In: 2006 IEEE Computer society conference on computer vision and pattern recognition (CVPR 2006), 17-22 June 2006, New York, NY, USA, pp 2161–2168
17. Pagh R, Stöckel M, Woodruff DP (2014) Is min-wise hashing optimal for summarizing set intersection? In: Proceedings of the 33rd ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, PODS'14, Snowbird, UT, USA, June 22-27, 2014, pp 109–120

18. Philbin J, Chum O, Isard M, Sivic J, Zisserman A (2007) Object retrieval with large vocabularies and fast spatial matching. In: 2007 IEEE Computer society conference on computer vision and pattern recognition (CVPR 2007), 18-23 June 2007, Minneapolis, Minnesota, USA
19. Qu Y, Song S, Yang J, Li J (2013) Spatial min-hash for similar image search. In: International conference on internet multimedia computing and service, ICIMCS '13, huangshan, China - August 17 - 19, 2013, pp 287–290
20. Shao J, Wu F, Ouyang C, Zhang X (2012) Sparse spectral hashing. Pattern Recogn Lett 33(3):271–277
21. Sivic J, Zisserman A (2003) Video google: a text retrieval approach to object matching in videos. In: 9th IEEE international conference on computer vision (ICCV 2003), 14-17 October 2003, Nice, France, pp 1470–1477
22. Torralba A, Fergus R, Weiss Y (2008) Small codes and large image databases for recognition. In: 2008 IEEE Computer society conference on computer vision and pattern recognition (CVPR 2008), 24-26 June 2008, Anchorage, Alaska, USA
23. Vsemirnov M (2004) Automorphisms of projective spaces and min-wise independent sets of permutations. SIAM J Discrete Math 18(3):592–607
24. Wang Y, Wu L (2018) Beyond low-rank representations: Orthogonal clustering basis reconstruction with optimized graph structure for multi-view spectral clustering. Neural Netw 103:1–8
25. Wang Y, Lin X, Zhang Q (2013) Towards metric fusion on multi-view data: a cross-view based graph random walk approach. In: 22nd ACM international conference on information and knowledge management, CIKM'13, San Francisco, CA, USA, October 27 - November 1, 2013, pp 805–810
26. Wang Y, Lin X, Wu L, Zhang W, Zhang Q (2014) Exploiting correlation consensus: towards subspace clustering for multi-modal data. In: Proceedings of the ACM International Conference on Multimedia, MM '14, Orlando, FL, USA, November 03 - 07, 2014, pp 981–984
27. Wang Y, Lin X, Zhang Q, Wu L (2014) Shifting hypergraphs by probabilistic voting. In: Advances in knowledge discovery and data mining - 18th pacific-asia conference, PAKDD 2014, Tainan, Taiwan, May 13-16, 2014. proceedings, Part II, pp 234–246
28. Wang Y, Lin X, Wu L, Zhang W (2015) Effective multi-query expansions: Robust landmark retrieval. In: Proceedings of the 23rd Annual ACM Conference on Multimedia Conference, MM '15, Brisbane, Australia, October 26 - 30, 2015, pp 79–88
29. Wang Y, Lin X, Wu L, Zhang W, Zhang Q (2015) LBMCH: Learning bridging mapping for cross-modal hashing. In: Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval, Santiago, Chile, August 9-13, 2015, pp 999–1002
30. Wang Y, Lin X, Wu L, Zhang W, Zhang Q, Huang X (2015) Robust subspace clustering for multi-view data by exploiting correlation consensus. IEEE Trans Image Process 24(11):3939–3949
31. Wang Y, Zhang W, Wu L, Lin X, Fang M, Pan S (2016) Iterative views agreement: an iterative low-rank based structured optimization method to multi-view spectral clustering. In: Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, IJCAI 2016, New York, NY, USA, 9-15 July 2016, pp 2153–2159
32. Wang Y, Lin X, Wu L, Zhang W (2017) Effective multi-query expansions: Collaborative deep networks for robust landmark retrieval. IEEE Trans. Image Processing 26(3):1393–1404
33. Wang Y, Zhang W, Wu L, Lin X, Zhao X (2017) Unsupervised metric fusion over multiview data by graph random walk-based cross-view diffusion. IEEE Trans Neural Netw Learning Syst 28(1):57–70
34. Wang Y, Wu L, Lin X, Gao J (2018) Multiview spectral clustering via structured low-rank matrix factorization. IEEE Trans Neural Networks and Learning Systems
35. Wu L, Wang Y (2017) Robust hashing for multi-view data: Jointly learning low-rank kernelized similarity consensus and hash functions. Image Vision Comput. 57:58–66
36. Wu L, Wang Y, Gao J, Li X (2018) Deep adaptive feature embedding with local sample distributions for person re-identification. Pattern Recogn 73:275–288
37. Wu L, Wang Y, Li X, Gao J (2018) Deep attention-based spatially recursive networks for fine-grained visual recognition. IEEE Trans Cybernetics
38. Wu L, Wang Y, Ge Z, Hu Q, Li X (2018) Structured deep hashing with convolutional neural networks for fast person re-identification. Comput Vis Image Underst 167:63–73
39. Wu L, Wang Y, Li X, Gao J (2018) What-and-where to match: deep spatially multiplicative integration networks for person re-identification. Pattern Recogn 76:727–738
40. Yang H, Callan JP (2006) Near-duplicate detection by instance-level constrained clustering. In: SIGIR 2006: Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, Seattle, Washington, USA, August 6-11, 2006, pp 421–428
41. Zhang D, Chang S (2004) Detecting image near-duplicate by stochastic attributed relational graph matching with learning. In: Proceedings of the 12th ACM International Conference on Multimedia, New York, NY, USA, October 10-16, 2004, pp 877–884

42. Zhang S, Yang M, Wang X, Lin Y, Tian Q (2015) Semantic-aware co-indexing for image retrieval. IEEE Trans Pattern Anal Mach Intell 37(12):2573–2587
43. Zhou W, Li H, Wang M, Lu Y, Tian Q (2012) Binary SIFT: towards efficient feature matching verification for image search. In: The 4th international conference on internet multimedia computing and service, ICIMCS '12, Wuhan, China, September 9-11, 2012, pp 1–6
44. Zhou D, Li X, Zhang Y (2016) A novel cnn-based match kernel for image retrieval. In: 2016 IEEE International conference on image processing, ICIP 2016, Phoenix, AZ, USA, September 25-28, 2016, pp 2445–2449
45. Zobel J, Moffat A (2006) Inverted files for text search engines. ACM Comput Surv 38(2):6

Chengyuan Zhang was born in Hunan Province, China. He received PhD degree in computer science from the University of New South Wales. Currently, he is a lecturer in School of Information Science and Engineering of Central South University, China. His main research interests include information retrieval, query processing on spatial data and multimedia data.

Yunwu Lin was born in Hunan Province, China, on April 1, 1995. He received the B.E. degree from CSUFT, China, in 2017. He is currently a Postgraduate in the School of Information Science and Engineering at Central South University. His research interests include multimedia systems and information retrieval.

Lei Zhu was born in Changsha, China, on June 7, 1988. He received the M.Sc. degree from Central South University, China, in 2014. He is currently a Ph.D. candidate in computer science and technology of Central South University. His research interests are in the field of multimedia systems and retrieval, spatio-temporal database and social media.

XinPan Yuan received the Ph.D. degree in Information Science and Engineering, Central South University in 2012, He is now a lecturer in School of Computer, Hunan University of Technology, Zhuzhou, China. His current research interests include Natural language processing and Information retrieval.

Jun Long is a Professor of School of Information Science and Engineering of Central South University, China. He received the M.Sc. and Ph.D. Degrees from Central South University, China, 2003 and 2011, respectively, both in computer science. His major research interests include network management, QoS guarantees and web service.

Fang Huang was born in Changsha, China. She received the PhD degree in traffic information engineering and control from Central South University, China, in 2007. Currently, she is a professor in School of Information Science and Engineering of Central South University, China. Her main research interests include social network mining and analysis, data mining and knowledge discovery, and big data analysis.