Sim-Min-Hash: An efficient matching technique for linking large image collections

Wan-Lei Zhao, Hervé Jégou, Guillaume Gravier

Sim-Min-Hash: An efficient matching technique for linking large image collections

Wan-Lei Zhao, INRIA, Rennes
Hervé Jégou, INRIA, Rennes
Guillaume Gravier, CNRS/IRISA, Rennes

Abstract

One of the most successful method to link all similar images within a large collection is min-Hash, which is a way to significantly speed-up the comparison of images when the underlying image representation is bag-of-words. However, the quantization step of min-Hash introduces important information loss. In this paper, we propose a generalization of min-Hash, called Sim-min-Hash, to compare sets of real-valued vectors. We demonstrate the effectiveness of our approach when combined with the Hamming embedding similarity. Experiments on large-scale popular benchmarks demonstrate that Sim-min-Hash is more accurate and faster than min-Hash for similar image search. Linking a collection of one million images described by 2 billion local descriptors is done in 7 minutes on a single core machine.

Keywords: Image linking, image search, min-Hash

1. Introduction

In the last decade, we have witnessed the explosion of multimedia contents in Internet. Professional and user-generated multimedia data proliferate over TV channels and Internet, in particular on social networks. There exist a significant proportion of the images (near-)duplicate or visually related content. A recent survey in [16] estimates that 22% of images have near-duplicates in a collection of two billion web images, and 8% are near-duplicate to more than ten images.

This raises the scientific challenge of linking all images within a large collection, to facilitate copyright enforcement, smart navigation, and more generally the analysis and organization of the multimedia databases by adding hyperlinks among images/videos [6, 10]. In addition, a graph connecting relevant visual content is useful to improve the quality of traditional content-based image search methods. For instance, the graph is a pre-requisite for search techniques based on reciprocal nearest neighbors [8, 12] or database augmentation techniques [1].

This paper considers this cross-matching problem for large image collections. The solutions [4, 14, 15, 3] that have been proposed to overcome the underlying quadratic complexity can be grouped into two categories, depending on whether the features they adopt are global or local. The global approaches are the most efficient. For instance, the method presented in [15] maps global feature vectors to binary hash codes. Images are grouped as near-duplicates once their binary codes are identical or similar. As other approaches based on global descriptors, this approach is efficient but only able to connect very similar images.

The methods based on local features are the most precise. Thanks to the introduction of BoVW based retrieval framework, one can find the results associated with a query image in a fraction of second [9, 7, 12]. However, in the cross-matching problem, all the images of one collection have to be submitted in turn. A very simple local approach [14] circumvents the problem by using a very small number of descriptors per image. This reduces the constant factor significantly, but results in poor linking quality, as shown in our experiments. One of the most successful approach to this problem is the min-Hash technique, which was first introduced in image retrieval for fast matching [4, 5]. It incorporates geometrical verification in the construction process to remove outliers [3] and links all images that are spatially related. In essence, this method computes a similarity measure which aims at approximating the Jaccard similarity between bag-of-words. As a result, it provides an excellent trade-off between linking efficiency and quality. However, as now shown in many papers (*e.g.*, [7]), the bag of words introduces large amount of quantization errors and has been outperformed in content-based image retrieval.

This paper introduces an extension of min-Hash, called Sim-min-Hash (SMH), to compare sets of real-valued vectors. It drastically improves the comparison metric between images while being very efficient for the retrieval and linking problems. This is achieved by adding to the original sketches extra information, in particular in the form of binary codes. The underlying motivation is to exploit the similarity measurements between vectors, in the spirit of Ham-

ming Embedding technique [7] in the bag-of-word context. It is then applied in the context of min-Hash for the purpose of image link discovery. As a result, our method drastically decreases the number of false collisions between hash codes and improve the precision. In addition, unlike traditional min-Hash, it is possible to keep all the matches in the main memory for a million-sized image collection, as will be demonstrated by our experiments in Section 4.

2. Background: min-Hash

This section briefly introduces the basics of min-Hash [2], as first introduced in the general context of efficiently estimating the Jaccard measure between sets. We then explain how it was used for the purpose of image comparison.

Let's first consider the space $\mathcal{O}$ of possible objects. For instance, this space could be the set of English words. The space $\mathcal{O}$ is assumed discrete but not necessarily finite. The similarity between two objects o_1 and o_2 is binary: They are identical or different, which is denoted by the Kronecker operator: $\delta(\mathsf{o}_1, \mathsf{o}_2) = 1$ *iff* $\mathsf{o}_1 = \mathsf{o}_2$, otherwise 0.

A document $\mathcal{X} \subset \mathcal{O}$ is a set of objects. Various similarity measures exist to assess the resemblance of two documents $\mathcal{X}$ and $\mathcal{Y}$. We consider the Jaccard similarity coefficient, which normalizes the number of objects by the number of distinct objects in the two sets:

$$J(\mathcal{X}, \mathcal{Y}) = \frac{|\mathcal{X} \cap \mathcal{Y}|}{|\mathcal{X} \cup \mathcal{Y}|}. \quad (1)$$

This similarity equals 0 when there is no object in common, and to 1 *iff* the sets are identical. In practice, this similarity measure is costly. The objective of min-Hash is to estimate the Jaccard coefficient in a probabilistic manner to speed-up the comparison. For this reason, it is seen as a kind of Locality-Sensitive-Hashing technique for sets.

Min-Hash proceeds as follows. First, we define a set of M hash functions $\pi_j,\ j = 1 \dots \mathsf{M}$. Each function $\pi_j : \mathcal{O} \to \mathbb{N}$ maps any possible object to an integer: $\mathsf{o} \mapsto \pi_j(\mathsf{o})$. Then, for any set $\mathcal{X} \subset \mathcal{O}$, we compute a set of M distinct hash keys $\sigma_j(\mathcal{X})$ as

$$\sigma_j(\mathcal{X}) = \min_{\mathsf{o} \in \mathcal{X}} \pi_j(\mathsf{o}). \quad (2)$$

The concatenation of these keys gives a vector of integers $\boldsymbol{\sigma}(\mathcal{X}) = [\sigma_1(\mathcal{X}), \dots, \sigma_{\mathsf{M}}(\mathcal{X})]$, which is the min-Hash representation of the set $\mathcal{X}$.

The key property exploited by min-Hash is the following. For any key $\sigma_j(.)$ and two documents $\mathcal{X}$ and $\mathcal{Y}$, we have

$$\mathbb{P}\left(\sigma_j(\mathcal{X}) = \sigma_j(\mathcal{Y})\right) = J(\mathcal{X}, \mathcal{Y}). \quad (3)$$

Since the expectation of $\delta(\sigma_j(\mathcal{X}), \sigma_j(\mathcal{Y}))$ equals the probability $\mathbb{P}\left(\sigma_j(\mathcal{X}) = \sigma_j(\mathcal{Y})\right)$, and assuming that the hash functions are independently drawn, min-Hash estimates the Jaccard coefficient by a similarity $\mathsf{s}_{\mathsf{mh}}(.,.)$ as

$$J(\mathcal{X}, \mathcal{Y}) \approx \mathsf{s}_{\mathsf{mh}}(\mathcal{X}, \mathcal{Y}) = \frac{1}{\mathsf{M}} \sum_{j=1}^{\mathsf{M}} \delta(\sigma_j(\mathcal{X}), \sigma_j(\mathcal{Y})), \quad (4)$$

where the variance of the approximation linearly depends on M^{-1}. One key ($\mathsf{M} = 1$) is not sufficient because the result is binary. A reasonable approximation for the estimated similarity s_{mh} is typically obtained with a few hundreds keys, depending on the statistical property of subsets of $\mathcal{O}$.

Sketches. The comparison can be sped-up by compounding several hash keys into one [4], which is called *sketch*. This reduces the false positive rate since it is equivalent to requiring the co-occurrence of multiple objects. For instance, by compounding two keys into one, min-Hash identifies matching pairs of objects. Note that the use of sketches also reduces the probability of observing the true matches. A good compromise is to combine two hash keys into one, *i.e.*, to focus on the pairs of objects [3].

Image comparison with min-Hash. min-Hash is particularly adapted to the case of binary BoVW [13], where a visual vocabulary comprising k visual words is used to produce a binary vector that represents the image. The i^{th} component equals 1 if at least one SIFT descriptor is assigned to the i^{th} visual word, otherwise 0.

Min-Hash is straightforwardly applied by considering all visual words as objects. The functions σ_j are defined as independent random permutations of the set $[1 \dots \mathsf{k}]$. We consider specifically the case of sketches where a code is produced from a pair of hash keys as

$$\sigma'_j(\mathcal{X}) = (\sigma_{2j}(\mathcal{X}) - 1) \times \mathsf{k} + \sigma_{2j-1}(\mathcal{X}), \quad (5)$$

in which case the sketch $\sigma'_j(\mathcal{X}) \in [1 ... \mathsf{k}^2]$. An image is then described by the min-Hash vector re-defined as $\boldsymbol{\sigma} = [\sigma'_1(\mathcal{X}), \dots, \sigma'_{\mathsf{M}'}(\mathcal{X})]$, where $\mathsf{M}' = \mathsf{M}/2$ denotes the number of sketches. Two vectors are compared with Eqn. 4. This is efficiently done by storing the images, based on their sketches, in look-up tables or inverted files. For details, the reader may refer to [3].

The computational advantage of min-Hash over binary BoVW is that the intersection is carried out via pairwise comparison of sketch vectors. This is cheaper than performing the bag-of-words comparison for two reasons. Firstly, the number of sketches M' is smaller than the number of non-zero components. Secondly, considering sketches made from pairs of features (*i.e.*, co-occurrences), the space of possible values is significantly larger than the number of visual words. Therefore with min-Hash, comparing vectors which are significantly more sparse than binary BoVW, is a key factor for efficient comparison [9].

3. Sim-Min-Hash

This section first introduces our Sim-min-Hash framework for matching set of real-valued vectors. It then focuses on the particular case of using Hamming Embedding as a similarity measurement, before a discussion on implementation details.

Sim-min-Hash: The principle. We are interested in using a similarity between vectors that is no longer binary, unlike for the objects considered in regular min-Hash. Therefore, we consider a similarity measure $m : \mathcal{O} \times \mathcal{O} \rightarrow \mathbb{R}+$ assigning a real value $m(\mathsf{x}, \mathsf{y})$ to any pair of objects. We are specifically interested in similarity measures of the form

$$\begin{cases} m(\mathsf{x},\mathsf{y}) = 0 & \text{if } q(\mathsf{x}) \neq q(\mathsf{y}) \\ m(\mathsf{x},\mathsf{y}) = \Omega(\mathsf{x},\mathsf{y}) & \text{if } q(\mathsf{x}) = q(\mathsf{y}) \end{cases}, \tag{6}$$

where $q(.)$ is a hashing function, for instance a quantizer assigning a visual word to a vector. Such similarities are interesting for computational purpose, as only vectors assigned to the same quantization index are compared. Note that the constant function $\Omega(.,.) = 1$ amounts to performing a binary comparison between objects. This shows that the regular min-Hash is a particular case of our method.
The Sim-min-Hash method proceeds as follows.

1. The functions π_j are defined by applying a random permutation to the quantization indexes.
2. For each hash function π_j, we compute the hash key as in Eqn. 2. However, we also keep track of the object (*i.e.*, the vector) for which the minimum is obtained, which is denoted by $\mathsf{c}_j(\mathcal{X})$. Note that we assume that this object is unique, *i.e.*, only one vector of $\mathcal{X}$ is assigned to the same index. Since it is not the case for local descriptors, we select one based on a rule. In our case, we simply keep the one that appears first when enumerating the set $\mathcal{X}$.

The similarity from Eqn. 4 is then modified to integrate the measurement between vectors, as

$$\mathsf{s}_{\mathsf{smh}}(\mathcal{X},\mathcal{Y}) = \frac{1}{\mathsf{M}} \sum_{j=1}^{\mathsf{M}} \Omega(\mathsf{c}_j(\mathcal{X}), \mathsf{c}_j(\mathcal{Y})). \tag{7}$$

The sketch extension mentioned in Section 2 is also naturally integrated in this framework, as explained below.

SMH with Hamming Embedding (HE). HE [7] is an effective extension of BoVW to better compare images based on their local features. Compared to BoVW, a binary signature $b(\mathsf{x})$ is computed per feature x, which is then represented by a tuple $(q(\mathsf{x}), b(\mathsf{x}))$. The advantage of using a binary code is that the feature is compact in memory: The binary code length is typically 64 bits (8 bytes). HE formally computes the similarity between two images as

$$s_{\text{he}}(\mathcal{X},\mathcal{Y}) = \frac{1}{\alpha_{\mathcal{X}}\alpha_{\mathcal{Y}}} \sum_{\mathsf{x}\in\mathcal{X}} \sum_{\mathsf{y}\in\mathcal{Y}} w\left(h\left(b(\mathsf{x}), b(\mathsf{y})\right)\right) \times \delta(q(\mathsf{x}), q(\mathsf{y})), \tag{8}$$

where $\alpha_{\mathcal{X}}$ and $\alpha_{\mathcal{Y}}$ are normalization factors and $h(.,.)$ measures Hamming distance between $b(\mathsf{x})$ and $b(\mathsf{y})$. The scalar function $w(.)$ gives a weight to the Hamming distance between the bit-vectors $b(\mathsf{x})$ and $b(\mathsf{y})$. The lower is the distance, the stronger is the weight, which equals 0 above a certain threshold $\mathsf{h_t}$. We adopt the entropy weighting scheme [7].

The Sim-min-Hash framework naturally applies to HE: Eqn. 7 is simply used with

$$\Omega(\mathsf{c}_j(\mathcal{X}), \mathsf{c}_j(\mathcal{Y})) = w\left(h\left(b(\mathsf{c}_j(\mathcal{X})), b(\mathsf{c}_j(\mathcal{Y}))\right)\right). \tag{9}$$

In the case of sketches obtained from multiple hash keys, $\Omega(\mathsf{c}_j(\mathcal{X}), \mathsf{c}_j(\mathcal{Y})) = 0$ if (a) one of the quantization indices is different or if (b) one of the Hamming distances is above the threshold $\mathsf{h_t}$. If none of these conditions is satisfied, the weight is obtained by summing the different weights (in Eqn. 9) associated with the Hamming distances. In the rest of the paper, a sketch is obtained from two hash keys.

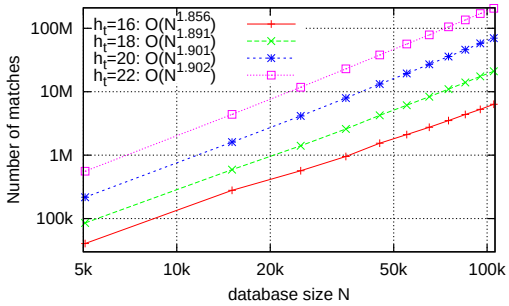

Figure 1. Number of collected matches sketches for varying image set size and of threshold $\mathsf{h_t}$ ($\mathsf{M} = 768$). The complexity is empirically estimated for all $\mathsf{h_t}$.

Implementation details. Similar to HE, the Sim-min-Hash representation takes a set of tuples (key, two binary codes) as input. The main difference is that the set of tuples is of fixed size, *i.e.*, M'. An inverted file is constructed for each of the M' sketches. The lists are accessed with the keys (which are obtained with Eqn. 5). Each element in the list contains an image id and two binary codes. Note that there is exactly one such entry per image in each inverted file. The search is conducted similarly as in HE. The difference is that SMH queries M' sketches separately in M' inverted files.

In the cross-matching scenario, the matched sketches are kept in a long buffer to improve the memory access efficiency (better use of cache with sequential access). This is possible if the number of matches does not explode with

Table 1. Retrieval Performance on University of Kentucky (4×Recall@top4) and Oxford5k (mAP), Oxford105k (mAP) Datasets. k=32k.

Method	UKB	Oxford5k	Oxford105k	avg. time*
BoVW	2.97	33.8	20.3	72 ms
HE [7]	**3.59**	**66.1**	**62.0**	46 ms
mHash [3]	2.37	19.1	13.8	26 ms
SMH	2.70	32.5	22.0	1.2 ms
Miner [14]	1.74	1.1	1.1	1.2 ms

*: the average time cost is measured on Oxford105k.

the size of dataset, which is possible with our technique even for million-sized collection. For the regular min-Hash, BoVW or HE, the total number of matches becomes rapidly prohibitive as the database grows. This strategy improves the speed by more than one order of magnitude compared to updating an array cumulating image scores.

The threshold h_t is one of the two parameters (along with M) that affects the quality and efficiency of the HE Sim-min-Hash. Its impact is analyzed in Fig. 1, where we plot the total number of matches collected when performing the full cross-matching of a large collection, namely the Oxford105k benchmark [11]. The figure shows that the number of matches grows quadratically for larger thresholds. This has an undesirable impact on both the memory and the efficiency. Lower thresholds reduce the burden of processing the matches and suppress the fast growth of memory consumption. In the rest of the paper, we adopt $h_t = 18$ as a compromise between quality and memory/speed complexity.

4. Experiments

This section compares Sim-min-Hash with HE with the regular min-Hash [3]. We also compare with the conventional BoVW and HE [7] approaches, and with the recent *Miner* technique [14], which focuses on image linking in large collections. The Miner uses fewer features (10 per image) and a smaller vocabulary size (k=1024), on purpose. For all other algorithms, we used the same input SIFT descriptors and visual vocabulary (k=32k) learned on an independent dataset. The standard choices are adopted for BoVW and HE [7], except that we use the RootSIFT [1] variant. $M' = 768$ sketches are used for both min-Hash and SMH. We focus on the quality of the initial links provided by these algorithms, which is the most costly operation. Note that techniques such as spatial verification or query expansion are complementary and shown effective with min-Hash [3].

Datasets and evaluation protocol. We have used public benchmarks, namely the Oxford5k-105k [11] (up to 105k images) and UKB benchmarks [9] (10k images). The evaluation is carried out considering two scenarios:

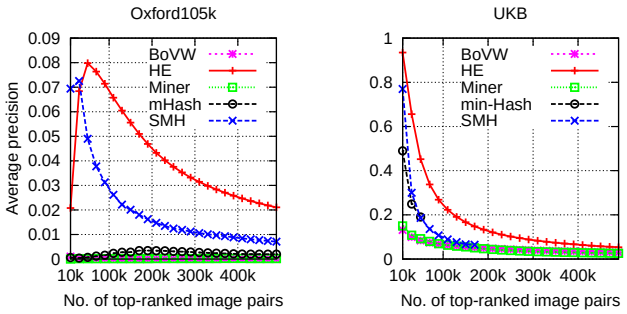

Figure 2. Link discovery: average precision against the number of selected links.

1. A regular image search task;
2. The discovery task to detect visually related images.

For image search, we adopt the standard evaluation protocol associated with each benchmark: mean average precision (mAP) is employed for Oxford datasets and the average number of images ranked in first 4 positions is used for UKB. For discovery, the systems must attribute a score to all possible links in the collection. The pairwise connections between images from image groups are treated as the ground-truth. The precision curves are drawn by checking the percentage of true-positive connections at each truncating point. Note that the precision on Oxford105k is low, partly because many true-positive pairs are out of the ground-truth coverage [3]. All the experiments have been pulled out on single core (2.8GHz).

Image retrieval scenario. Table 1 shows that the performance of SMH is consistently better than that of conventional min-Hash. It is also much faster by limiting the number of random memory accesses. Note however that SMH uses more memory (+16 bytes/sketch) to store the binary vectors. The SMH's performance is close to BoVW for this task, but it is 60 times faster. As expected, HE performs the best, but is much slower than SMH and requires more memory because the number of features is larger than the number of sketches. The Miner technique is not adapted to this retrieval task: It is very efficient but gives poor results.

Link discovery scenario. Figure 2 reveals a consistent behavior of the algorithms over two datasets. Both Miner and BoVW achieve poor performances because the absolute scores are not reliable. In contrast, the Jaccard measure estimated by min-Hash is more consistent across queries. This explains the better behavior of min-Hash over BoVW for discovery compared to the search task, since search only considers the relative scores for a given query.

The best methods are clearly those using a non-binary similarity between local descriptors, namely HE and our SMH method. In this discovery task, SMH significantly outperforms BOW and min-Hash. SMH is especially in-

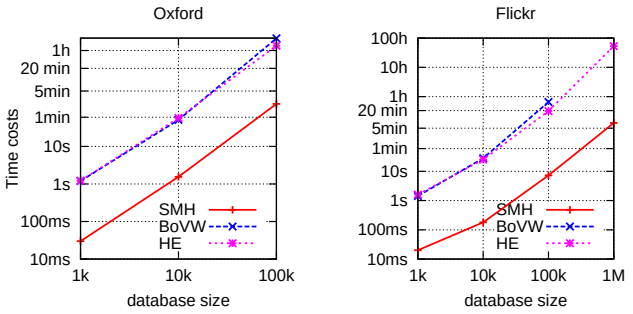

Figure 3. Timings for linking an image collection.

teresting for this task because it is the most discriminative approach in the set of efficient approaches (mHash and Miner).

Complexity and timings. All the algorithms considered here are *theoretically* quadratic in the number of images. Figure 3 reports the cost, for varying subset sizes and excluding the description cost, of finding all the links from the Oxford105k dataset and from a larger dataset comprising one million Flickr images and described by a total of approximately 2 billion descriptors. The timings have been measured on a single core of the same machine. The code has a similar degree of optimization for all techniques.

As one can see, SMH is between one and two orders of magnitude faster than BoVW and HE. With the same query time, it processes a set at least one order of magnitude larger than these techniques.

5. Conclusion

Sim-min-Hash is an efficient and effective way to compare sets of vectors, which improves the original min-Hash technique for comparing set of vectors. It is well adapted to linking images in large collections, thanks to its higher efficiency over regular search techniques: Linking the images of a million-sized image collection is fulfilled in about 7 minutes on a single processor core.

Acknowledgements

This work was done as part of the Quaero project, funded by Oseo, French agency for innovation.

References

[1] R. Arandjelović and A. Zisserman. Three things everyone should know to improve object retrieval. In *CVPR*, 2012.

[2] A. Z. Broder. On the resemblance and containment of documents. In *Proc. Compression and Complexity of Sequences*, 1997.

[3] O. Chum and J. Matas. Large-scale discovery of spatially related images. *PAMI*, Feb. 2010.

[4] O. Chum, J. Philbin, M. Isard, and A. Zisserman. Scalable near identical image and shot detection. In *CIVR*, 2007.

[5] O. Chum, J. Philbin, and A. Zisserman. Near duplicate image detection: min-hash and tf-idf weighting. In *BMVC*, 2008.

[6] K. Heath, N. Gelfand, M. Ovsjanikov, M. Aanjaneya, and L. J. Guibas. Image webs: Computing and exploiting connectivity in image collections. In *CVPR*, 2010.

[7] H. Jégou, M. Douze, C. Schmid. Improving bag-of-features for large scale image search. *IJCV*, May 2010.

[8] H. Jégou, C. Schmid, H. Harzallah, and J. Verbeek. Accurate image search using the contextual dissimilarity measure. *PAMI*, Jan. 2010.

[9] D. Nistér and H. Stewénius. Scalable recognition with a vocabulary tree. In *CVPR*, 2006.

[10] L. Pang, W. Zhang, H.-K. Tan, and C.-W. Ngo. Video hyperlinking: libraries and tools for threading and visualizing large video collection. In *ACMMM*, 2012.

[11] J. Philbin, O. Chum, M. Isard, J. Sivic, and A. Zisserman. Object retrieval with large vocabularies and fast spatial matching. In *CVPR*, 2007.

[12] D. Qin, S. Gammeter, L. Bossard, T. Quack, and L. van Gool. Hello neighbor: Accurate object retrieval with k-reciprocal nearest neighbors. In *CVPR*, 2011.

[13] J. Sivic and A. Zisserman. Video Google: A text retrieval approach to object matching in videos. In *ICCV*, 2003.

[14] W. Voravuthikunchai, B. Crémilleux, and F. Jurie. Findging groups of duplicate images in very large datasets. In *BMVC*, 2012.

[15] B. Wang, Z. Li, M. Li, and W.-Y. Ma. Large-scale duplicate detection for web image search. In *ICME*, 2006.

[16] X.-J. Wang, L. Zhang, M. Liu, and W.-Y. Ma. ARISTA: image search to annotation on billions of web photos. In *CVPR*, 2010.