Image re-ranking system based on closed frequent patterns

Winn Voravuthikunchai, Bruno Crémilleux, Frédéric Jurie

Noname manuscript No.
(will be inserted by the editor)

Image re-ranking based on closed frequent patterns

Winn Voravuthikunchai · Bruno Crémilleux · Frédéric Jurie

Received: date / Accepted: date

Abstract Text-based image retrieval is a popular and simple framework, which consists in using text annotations (*e.g.* image names, tags) to efficiently collect images relevant to a query word, from very large image collections. Even if the set of images retrieved using text annotations is noisy, it constitutes a reasonable initial set of images that can be considered as a bootstrap and improved further by analyzing image content. In this context, this paper introduces an approach for improving this initial set by *re-ranking* the so-obtained images, assuming that non-relevant images are scattered (*i.e.* they do not form clusters), unlike the relevant ones. More specifically, the approach consists in computing efficiently and on-the-fly *closed frequent patterns*, and in re-ranking images based on the number of patterns they contain. To do this, the paper introduces a simple but powerful new scoring function. Moreover, after the re-ranking process, we show how pattern mining techniques can also be applied for promoting diversity in the top-ranked images. The approach is validated on three different datasets for which state-of-the-art results are obtained.

Keywords Data mining, Frequent patterns, Image re-ranking, Image search

Winn Voravuthikunchai
GREYC - UMR6072 CNRS, University of Caen, ENSICAEN, France
Tel.: +33-231567434
Fax.: +33-231567330
E-mail: winn.voravuthikunchai@unicaen.fr

Bruno Crémilleux
GREYC - UMR6072 CNRS, University of Caen, ENSICAEN, France
Tel.: +33-231567435
Fax.: +33-231567330
E-mail: bruno.cremilleux@unicaen.fr

Frédéric Jurie
GREYC - UMR6072 CNRS, University of Caen, ENSICAEN, France
Tel.: +33-231567498
Fax.: +33-231567330
E-mail: frederic.jurie@unicaen.fr

1 Introduction

Web-image search has become a key feature of well-known search engines such as 'Google', 'Yahoo', 'Bing', *etc.* Similar to the way in which web-pages are retrieved, a user searches for images of interest by simply inputting a text query to the search engine. Within less than a second, millions of images related to the query are retrieved. Most of these search engines are primarily based on the use of text meta-data such as keywords, tags, and/or text descriptions nearby the images. Although retrieving images based on meta-data is extremely fast using inverted files, the retrievals are usually mixed with a significant amount of undesirable non-relevant images, due to the fact that keywords or tags do not always correspond to the visual content of the images. In Web-image search, users usually observe only the top few tens or hundreds of images. This implies that for this scenario, precision is more important than recall. In other words, having less images returned, if they are more relevant, is more interesting than having more images mixed with non-relevant ones.

The retrieval system can be improved by using a secondary re-ranking system in a cascade fashion. The key idea is to take benefit from the use of the visual information contained in the images, as shown by [17]. This new step re-orders the images by having the relevant images to be displayed up front of the non-relevant ones. Even if an additional computing step is performed, the

Image I_i	*Trans.* t_i	*rel.*
I_1	$\{a_1, a_2, a_3\}$	*yes*
I_2	$\{a_1, a_4, a_6\}$	*yes*
I_3	$\{a_1, a_7, a_9\}$	*no*
I_4	$\{a_2, a_3, a_6\}$	*yes*
I_5	$\{a_4, a_5, a_8\}$	*no*

(a) Initial ranking order.

Patterns $\mathcal{X}_j$
$\mathcal{X}_1 = \{a_1\}$
$\mathcal{X}_2 = \{a_4\}$
$\mathcal{X}_3 = \{a_6\}$
$\mathcal{X}_4 = \{a_2, a_3\}$

(b) Closed frequent patterns

Image I_i	$\mathcal{X}_j$ in t_i	$\#\mathcal{X}_j$ in t_i	rel.
I_2	$\mathcal{X}_1, \mathcal{X}_2, \mathcal{X}_3$	3	*yes*
I_1	$\mathcal{X}_1, \mathcal{X}_4$	2	*yes*
I_4	$\mathcal{X}_3, \mathcal{X}_4$	2	*yes*
I_3	$\mathcal{X}_1$	1	*no*
I_5	$\mathcal{X}_2$	1	*no*

(c) Images re-ranked.

Fig. 1 Toy example illustrating the re-ranking of images according to the count of frequent patterns ($minfr = 2$). Details are given in Section 3.1.

re-ranking is only applied to the top retrieved images from the first stage (*e.g.* few hundreds), and that allows a trade-off between effectiveness and efficiency. We give now the main insights of our approach.

Web-image re-ranking can be seen as a binary classification problem where the relevant images belong to the positive class. Although true labels are not provided, it is still possible to build class models based on the two following assumptions. (i) The initial text-based search provides a reasonable initial ranking, which is to say that a majority of the top-ranked images are relevant to the query, meaning that classifiers such as SVMs can be trained by using the top-ranked images as (noisy) positive images while the images that are ranked below or even the images from other datasets are treated as negative images (see *e.g.* [4]). (ii) The relevant images are visually similar to each other (at least within groups) while the non-relevant images tend to be not similar to any other images. Graph based re-ranking approaches exploit this second assumption, by modeling the connectivity among retrieved images [21].

Besides the challenge of building a class model from noisy labeled data, a challenging aspect of web image re-ranking is the efficiency of the approach. Indeed, the re-ranking process has to be done on-the-fly – since the queries from the users are not known in advance – hence limiting the type of algorithm that can be used. Although many approaches in the literature have shown excellent re-ranking results, most of them are computationally expensive and therefore unsuitable for real web image re-ranking applications.

In this context, this paper proposes a new approach for image re-ranking, building on recent advances in data mining [1,29]. Our approach is based on *frequent pattern mining*, a key technique in data mining. It allows the discovery of sets of image visual features shared by many images (more precisely, we use *closed frequent patterns*, we explain below the reasons for this choice). The key idea is that frequent patterns are more likely to occur in relevant images since such images have related content (this is illustrated on a toy example in Figure 1). Furthermore, once frequent patterns are extracted, the patterns found in each image infer the connectivity with other images in the database. More precisely, a new scoring function using the statistics of frequent patterns found in images is proposed. This scoring function allows to re-rank images without having to train any expensive classification models, contrarily to [21]. The encoding of images as pattern mining transactions is also critical. We used the adaptive thresholding process of [33], giving the same number of items (*i.e.* features) for each image. This prevents the risk that some non-relevant images have more frequent patterns only because they have more items than the relevant ones. In addition, we also propose efficient mining techniques either based on the extraction of closed patterns in the transposed database or on multiple random projections. Consequently, the proposed approach not only gives excellent results in terms of accuracy, but it is also very fast. Therefore, it is suitable for re-ranking applications. The paper also shows that the approach can be used for more general outlier detection tasks. The approach is validated through extensive experiments on three different datasets, and compared to state-of-the-art algorithms.

Recently, an interesting question has been brought up to the community: how to organize the returned images in a way that facilitates the users to browse through the results and find their images of most interest as quickly as possible. Indeed, re-ranking systems based on visual content tend to return a lot of duplicates or very similar images in the top ranks. If an image has high score, obviously all of other images that look identical or very similar to the image will have also high ranking scores. Actually, it will be more interesting for users to see a variety of images, (*i.e.* from different class instances, colors, aspect ratios, sizes, view points, *etc.*) rather than seeing tens of duplicate or very similar images. Promoting diversity is another aspect of image retrieval besides the precision and efficiency of the systems. The most straightforward idea is to group similar images into clusters, and display only an instance of each cluster. The users not only can quickly search for the type of image, but also can refine their search by clicking on the cluster of interest to see all the images inside the cluster. Traditional clustering algorithms have been applied to the top-ranked images [18,30]. Unfortunately, such algorithms are usually time consuming and can not be used on-the-fly. Moreover, they are not

scalable to large datasets. Another contribution of our paper is to propose to apply our duplicate detection approach [32], also based on pattern mining techniques, to group near duplicate images together. It has been shown, in [32], that the process of detecting groups of near duplicate images is very fast and scalable to large datasets. This allows to be computed on-the-fly and to be used as a post processing step after the re-ranking process with just a small amount of additional computational cost.

The paper is organized as follows: Section 2 discusses related works, while Section 3 describes our approach in detail, including how to encode images as data mining transactions, how to extract frequent patterns, how to re-rank images using frequent patterns, and how to group near duplicate images. Section 4 provides the experimental validation, and finally, Section 5 concludes the paper with an analysis and discussion of the presented results, as well as possible future work based in our findings.

2 Related Work

Image re-ranking has attracted a lot of attention during the last five years. The different approaches in the literature differ in the way they model the class of relevant images, using (i) clustering and cluster centers, (ii) topic models, (iii) classification based models, (iv) graph based models or (v) data mining models.

Clustering-based re-ranking algorithms exploit the property that relevant images form clusters, as they share some common visual properties [3,5,16]. Images are then ranked according to their distance to one of the cluster centers (non-relevant images are expected to be far from cluster centers since these images are supposed to be very diverse and scattered). However, in practice, the relevant images have large visual variability (*e.g.* side-views and front-views of bicycles look very different) so it is difficult to determine the number of relevant clusters for a given query. Moreover, how to compare the relevance between the images of different clusters is still an open issue (*e.g.* 'Is an image belonging to a small cluster but close to its center more relevant than another image which belongs to a bigger cluster but being farther from its center?').

Topic models have been used to deal with the diversity of relevant image content [6,7,9]. Brought by the field of natural language processing, these models assume that the content of a document is generated by the set of topics it contains. Within this framework, each image is basically mapped to a lower dimensional topic space representing the strength of each topic in the image. The images are then ranked according to the dominating topics in the entire retrieval set, which means that if an image contains many dominating topics, it is likely to be relevant to the query.

Classification-based approaches have also been used for re-ranking *e.g.* [4,11,25]. In this case, a classifier is trained using the initial top-ranked images as noisy positive training images. The trained classifier is used to give a new – hopefully more relevant – score to each image. The problem is that the selected pseudo-positive and pseudo-negative images may not be truly-positive and truly-negative, and can damage the classification model. Moreover, a new classifier has to be learned for each new query. Krapac *et al* [17] proposed a method based on query-relative visual word features to train a single generic classifier that can be used across different queries. It is possible to calculate which visual words are strongly associated with the query set, as the majority of the images are the relevant images; the visual words which occur often are the ones strongly associated to the query set. The statistics of the amount of strongly associated visual words can reflect the relevance of the image and can be used as generic features. Thollard and Quénot [28] proposed to combine an unsupervised re-ranking approach with a supervised re-ranking one. The unsupervised approach is based on the hypothesis that a relevant image is visually similar to some other relevant images, while a non-relevant image does not share similarity with any other images. The ranking score of the approach is based on the average distances between the K-nearest neighbors. Regarding the supervised re-ranking approach of [28], the idea is to train a single 'junk' classifier to filter out some non-relevant images that are typically found across all queries. These noisy images share some similar characteristics such as they commonly have very small size and are less textured (*e.g.* 'icons', 'banners'). The two re-ranking methods and the original ranking are complimentary and can improve the overall ranking.

Graph-based methods [13,15,20,35] have shown very good results in image re-ranking. A fully connected graph is constructed from a query set, graph in which images are represented by nodes that are connected by vertices when they are close enough. A regularization scheme is applied, hence enforcing the scores to be smooth on the graph while keeping the score consistent with the prior information, (*i.e.* the initial text-based ranking). Unfortunately, such a graph-based approach has very high computational complexity, due to the computation of the distance between all image pairs and the computation of the pseudo-inverse of the adjacency matrix.

Finally, frequent pattern mining has been used for removing outliers in [21]. Each image is described as

a transaction (or pattern). A pattern is made of items which are the visual words located on images' interest points. Frequent pattern mining is applied to find frequent combinations of visual attributes – constituting new image features – used by a one-class SVM to re-rank the images. Despite our approach also uses patterns, it is very different from [21]. One key difference lies in the way images are encoded as transactions. In [21], as the number of items per image varies a lot from an image to another, non-relevant images (which contain often more patterns as they are often richer in shape and texture) will contain more frequent patterns and will consequently have higher scores. In contrast, we adopted the encoding strategy of [33], for the aforementioned reasons. Another important difference with [21] lies in the way images are ranked: [21] trains a one-class SVM, which is slow when the dimensionality of the data is large, forcing themselves to use very poor image representation. In comparison, the simplicity and the efficiency of our scoring function allow to extract frequent patterns from very high dimension image features (*i.e.* 2,000 visual words and 21 spatial pyramids bins [19], resulting in 42,000 features) and use hundreds of thousands frequent patterns to rank images.

For promoting the diversity of the top-ranked images, traditional clustering techniques have been applied as a post processing stage on the top ranked images [18,30]. From their discriminative power, they have been shown to be effective to promote the diversity. However, visual clustering requires highly time-consuming computation which prevents its use as an online process. Another way for promoting the diversity is to simultaneously take diversity and relevance into account simultaneously [36]. The selection is then done by choosing not only the most relevant images to the query but also according to the diversity of the already selected images. Although the computational complexity is lower than clustering approaches, it can still be very slow when processing high number of retrievals. Instead of these techniques, we propose to use our efficient groups of duplicate detection approach which can group similar images together. It has been shown in [32] that the run time is less than 3 minutes to detect groups of duplicates in a database of 1 million images by using only a single core machine. Applying the approach as a post processing step to our re-ranking system will cost less than a second of extra computational time for retrieval sets in a scale of hundreds or thousands.

3 Re-scoring of retrieved images

As said in the introduction, the rational for using frequent patterns (*i.e.* frequent groups of visual features jointly occurring in images) for re-ranking is that (i) patterns that occur frequently are likely to come from relevant images as relevant images do share similarities (in contrast with non relevant images which are scattered) and (2) they can be computed on-the-fly very efficiently.

This section presents new scoring functions based on frequent patterns and explains how images can be represented as sets of binary items. This binarization step — required for mining patterns as data mining algorithms can only handle binary items — is critical as it provides the information from which the score will be computed. Finally, we explain how frequent pattern mining can be used for detecting groups of near duplicate images. We propose to use our groups of duplicates detection system [32] after the re-ranking step in order to promote diversity of the retrievals.

3.1 Scoring function

Let $\mathcal{A} = \{a_1, \ldots, a_k\}$ denotes the set of all possible items. In our case, items are visual words (*i.e.* quantized local features), and $\mathcal{A}$ is the visual vocabulary. A set of items $\mathcal{X} \subseteq \mathcal{A}$, is called a pattern. The length of a pattern $l(\mathcal{X}) = |\mathcal{X}|$ is the number of items in the pattern. The set of images is denoted by $\mathcal{I} = \{I_1, \ldots, I_{|\mathcal{I}|}\}$, where $|\mathcal{I}|$ represents the number of images. Each image is represented by a pattern and forms a database entry, so-called a *transaction*. The transaction of image I_i is denoted as t_i. The set of transactions $\mathcal{T}$ obtained from the retrieved images is called the *transaction database*, denoted as $\mathcal{T} = \{t_0, \ldots, t_{|\mathcal{T}|}\}$. A pattern $\mathcal{X}$ can be covered by (*i.e.* can be subset of) many transactions and the set of transactions covering $\mathcal{X}$ is called the *cover* of $\mathcal{X}$ with respect to the transaction database $\mathcal{T}$, denoted as $K_{\mathcal{T}}(\mathcal{X})$. More formally, $K_{\mathcal{T}}(\mathcal{X}) = \{k \in \{1, \ldots, n\} \text{ s.t. } \mathcal{X} \subseteq t_k\}$. The frequency measure provides the number of occurrences of a pattern $\mathcal{X}$ in the database. $\mathcal{X}$ is considered to be a frequent pattern if its frequency $fr(\mathcal{X}) = |K_{\mathcal{T}}(\mathcal{X})|$ is above a minimum frequency threshold $minfr$ (in other words, $minfr$ is the minimum number of images that must contain a pattern $\mathcal{X}$ in order to consider $\mathcal{X}$ as a frequent pattern).

Using these notations, we define the set of frequent patterns as :

$$\mathcal{F}(\mathcal{T}, minfr) = \{\mathcal{X} \subseteq \mathcal{A} \text{ s.t. } fr(\mathcal{X}) \geq minfr\} \quad (1)$$

Toy example. In order to illustrate the approach, let us discuss the toy example depicted in Figure 1. In this example, the images are supposed to be retrieved by a text based search engine and sorted by their initial rank (from 1 to 5). There are three relevant images, I_1,

I_2, and I_4 and two non-relevant images, I_3 and I_5 (see Figure 1(a)). In the toy example, image I_2 is described by the items a_1, a_4 and a_6. The frequency of the pattern $\mathcal{X} = \{a_2, a_3\}$ is 2 and $\mathcal{X}$ is covered by I_1 and I_4 (*i.e.* $K_{\mathcal{T}}(\{a_2, a_3\}) = \{1, 4\}$).

Scoring function. A first possible scoring function can be:

$$S(I_i) = |\mathcal{F}(\mathcal{T}, minfr) \subseteq t_i| \quad (2)$$

which is the number of frequent patterns included in the transaction representing the image I_i. Following our toy example, Figure 1(b) gives the closed frequent patterns which have been extracted with $minfr = 2$ (Section 3.2.2 gives the details of the mining step). Note that the frequent patterns are found mostly in the relevant images. At last, Figure 1(c) shows the re-ranking of the images according to the number of frequent closed patterns they contain.

Each pattern is characterized by its frequency and its length, both playing a role in determining the relevance of the pattern. We therefore suggest to improve the basic scoring function (*i.e.* eq. 2) by weighting individually each pattern, more important patterns receiving higher weights. Eq. 3 implements this weighting scheme.

$$S(I_i) = \sum_{\mathcal{X} \in \{\mathcal{F}(\mathcal{T}, F_{min}) \subseteq t_i\}} w(\mathcal{X}) \quad (3)$$

The following paragraphs give four different definitions of the weighting function $w(\mathcal{X})$.

1. Frequency: Frequent patterns appear in many images, likely to be the positive ones (remind that positive images are assumed to share patterns with other positive images). In contrast, rare patterns appear only in very few images, most of them being non-relevant images (as they do not share many patterns with other images). This property can be expressed by defining w as: $w(\mathcal{X}) = fr(\mathcal{X})$. In this case, patterns with higher frequencies will contribute more to the scoring function.
2. Length: The length of the patterns reflects the similarity between the images sharing them. Positive images tend share many visual features with other positive images. Therefore, long patterns are likely to come from them. On the other hand, non-relevant images tend not to share features with other images, and, therefore, the patterns extracted from them tend to be shorter. Based on this observation, a second proposal for w is: $w(\mathcal{X}) = l(\mathcal{X})$ where longer patterns contribute more to the scoring function.
3. Area: Since both frequency and length are important, we can consider them together. More formally, w can be defined as: $w(\mathcal{X}) = fr(\mathcal{X}) \times l(\mathcal{X})$.
4. Use of the original ranks: We made the assumption that the original ranking (the one obtained from the text-based search engine) is reasonably good. Since the original position of each image in this initial ranking is known, we can use this information in the weighting of the patterns. This is what we do in the following definition of w, by summing the inverse of the original ranks of the images containing $\mathcal{X}$. More formally, w is defined as $w(\mathcal{X}) = \sum_{k \in K_{\mathcal{T}}(\mathcal{X})} \frac{1}{k}$. As an example, if the pattern $\mathcal{X}$ is found in I_1, I_3, and I_5 which are at the images in the first, third, and fifth position in the original ranking, $w(\mathcal{X}) = \frac{1}{1} + \frac{1}{3} + \frac{1}{5}$. According to this weighting scheme, the frequent patterns found in the top images should contribute more to the final score.

The different weighting schemes have been experimented in Section 4.2.

3.2 Mining image transactions

3.2.1 Representing images by transactions

In order to extract frequent patterns from images, images have to be represented as sets of binary items. Starting from Bag-of-Words (BoW) histograms, which is considered as a good choice for representing images [12,26,34], the idea is to obtain binary items by thresholding the bins of the BoW histogram. Visual words whose frequencies are above the threshold are set to 'one' and are considered as the items of the image. More formally, in the BoW representation, each image I is described as a histogram of visual words $\mathbf{h} = (p(w_0|I), \dots p(w_d|I))$ where d is the size of the dictionary. The binary vector is described as $h_i^b = 1 \iff h_i \geq \tau$, where τ is the threshold, and i is the bin index of the histogram. Options for setting τ are described along with the explanation of the advantages and disadvantages of each method in the following paragraphs.

Fixed threshold. The simplest way is to set a fixed constant global threshold $\tau = C$ (Figure 2a). The disadvantage of this method is that the images which have flat distributions can be entirely set to 'zeros' or 'ones' (Figure 2a-left) depending on the threshold value. Besides having a single threshold for all features, an alternative is to set distinct thresholds for 'each' bin/feature. For example, the threshold for each features, can be set according to its mean value $\bar{f}_i$ or its median value $\tilde{f}_i$. However, the number of attributes after binarization (*i.e.* the number of ones) can vary a lot from an image to the another. This is problematic when extracting patterns since the images with more items will contain more patterns than the images having less items.

First q-quantile of the sorted histogram. Another method is to set the threshold to the first q-quantile of the sorted histogram $\mathbf{h}^s$ (Figure 2b). We denote $\mathbf{h}^s$ as a vector in which components correspond to the bins of $\mathbf{h}$ sorted in descending order. Then $\tau = h^s_{i^*}$, where i^* verifies $\sum_{i \leq i^*-1} h^s_i < \frac{1}{q} \wedge \sum_{i \leq i^*} h^s_i \geq \frac{1}{q}$. This binarization method seems to be a suitable choice since the total amount of bits being set to one comes from a fixed proportion of the total probability distribution of the original histogram. Unlike the fixed threshold method, this thresholding ensures that at least one bit is set to one (each image contains at least one item), and there will not be any case where all the bits are set to one (no image contain all items). This binarization method can avoid having too many or too less items in flat distributions (Figure 2b-left) as it would be if using a fixed threshold (Figure 2a-left), However, a problem remains when only a few visual words dominate the image. We give here an example of an image of a small plane flying in the sky. This image contains a large amount of a single non-gradient visual word. The amount of this visual word exceeds the first q-quantile which sets only the bit corresponding to this visual word to one. In this case, the visual words representing the plane are discarded. Having only one attribute, the combinations of attributes in this image are limited. Figure 2b-right illustrates this problem.

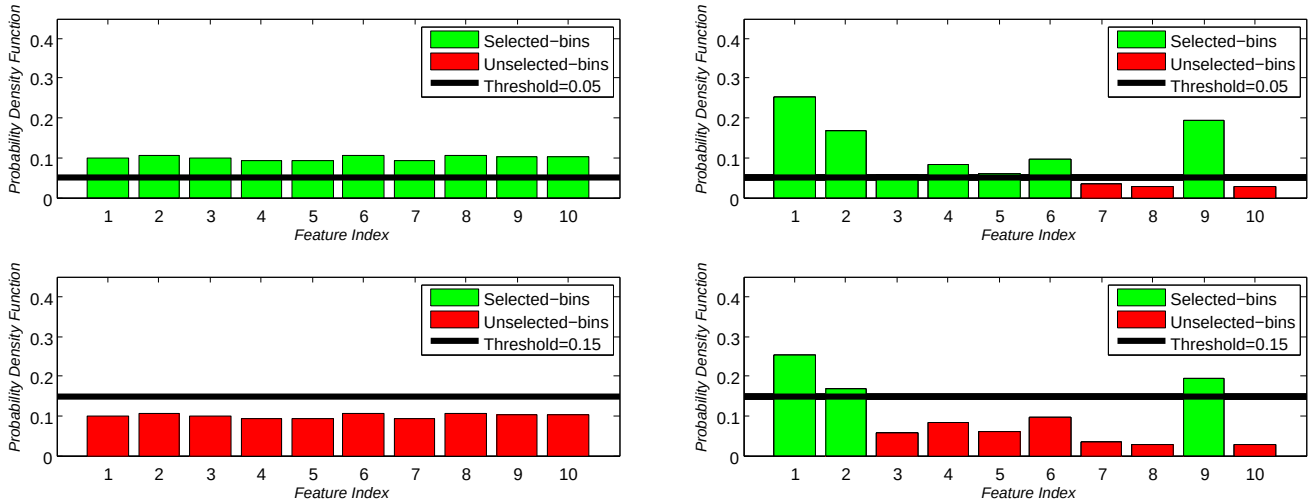
Top-K bins. This method sets τ to the top-k most frequent visual words, $\tau = h^s_k$ (Figure 2c). Selecting the visual words which have the highest counts is favorable since they are the most representative features in the image. Unlike the two previously described methods, this binarization ensures that all images will have a reasonable amount of attributes. This can avoid the problems of having too high or too low number of attributes as mentioned in the previous examples. The main advantage of this binarization is that images will have exactly the same number of items, which ensures that the number of frequent patterns found in each image truly reflects the connectivity with other images. Using other binarization methods, may allow non-relevant images to have more frequent patterns than the relevant ones, due to the possibility of having more items in the transactions. Furthermore, having a fixed number of attributes also makes the implementation more efficient. For example, it allows to adopt fixed array sizes which is much more efficient in terms of computation time and memory usage (the exact need of memory is known in advance).

3.2.2 Efficient frequent pattern mining

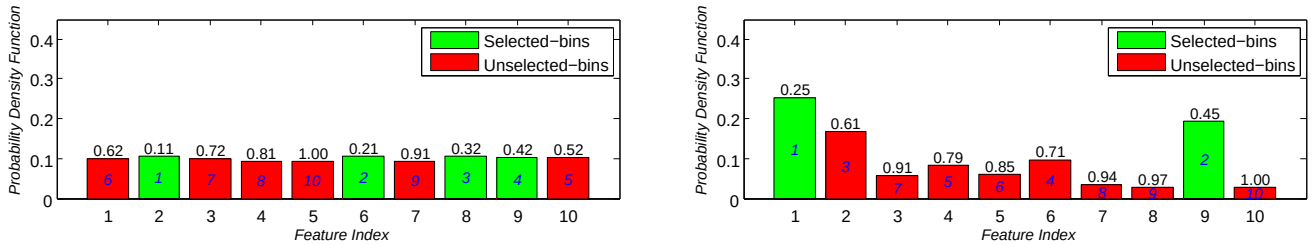
Once images are represented as sets of items, the next step is to extract *frequent patterns* [1] (see definition Section 3.1).

Pattern condensed representations As each subset of a frequent pattern is also a frequent pattern, the entire set of frequent patterns can be very large and can include redundant information (*i.e.* many frequent patterns are extracted from the same set of images). To reduce this redundancy, we consider two condensed-representations of frequent patterns: (i) *closed frequent patterns* [22], and (ii) *maximal frequent patterns* [2]. Regarding the definitions, $\mathcal{X}$ is a closed frequent pattern if $\mathcal{X}$ is frequent and $\nexists Y \supset \mathcal{X} \mid fr(\mathcal{X}) = fr(\mathcal{Y})$ where $\mathcal{Y}$ is any superset of $\mathcal{X}$. A closed pattern summarizes the frequency of a subset of patterns having the same frequency value. On the other hand, $\mathcal{X}$ is a maximal frequent pattern, if $\mathcal{X}$ is frequent and $\nexists Y \supset \mathcal{X} \mid fr(\mathcal{Y}) \geq minfr$ where $\mathcal{Y}$ is any superset of $\mathcal{X}$. Maximal patterns are the longest patterns (w.r.t. the items) satisfying the $minfr$ threshold. All frequent patterns can be derived from both condensed representations but the difference lies in the knowledge of the frequency values: the exact frequencies of all the frequent patterns can be derived from the frequent closed patterns but not from the maximal frequent patterns. Mining closed or maximal patterns also significantly enhances the computing effort [29]. Indeed, specific pruning techniques make the mining of closed and maximal frequent patterns much more efficient than the mining of the whole set of frequent patterns. Moreover, since the amount of closed frequent patterns is much less than the amount of frequent patterns, the computational cost of the re-ranking stage is reduced. In practice, we experimented with frequent patterns, closed frequent patterns, and maximal frequent patterns and concluded from these experiments that closed frequent patterns gave the best results both in terms of efficiency and performance. In practice, we use the LCM algorithm to extract closed and maximal frequent patterns [29]. LCM stands for Linear time Closed pattern Miner and is one of the fastest implementations to mine closed frequent patterns. Theoretically, the complexity of LCM is bounded by a linear function with respect to the number of closed frequent patterns, thus its acronym. LCM uses a prefix preserving closure extension to completely enumerate closed patterns. This allows counting the support of a pattern efficiently during the mining process.

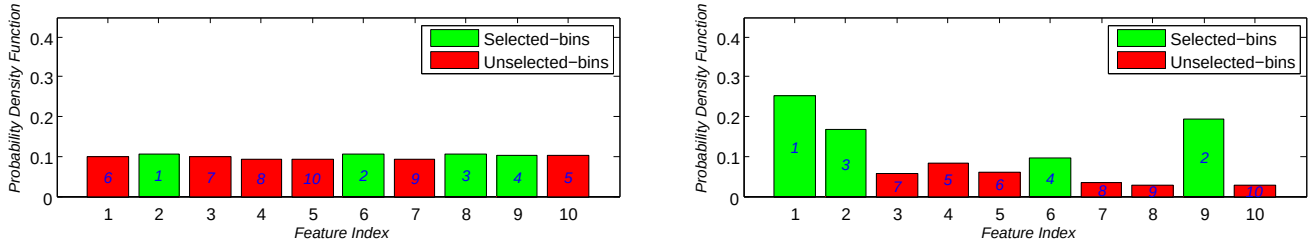
Improving mining efficiency. The mining complexity is linear with the number of images but can grow exponentially in the worst case with the number of items,



(a) Fixed threshold binarization.The bins having values above the threshold (black straight line) are selected as transaction items. Problems occurs in flat distributions where all bins are selected (top-left) or none of the bins is selected (bottom-left).



(b) First q-quantile of the sorted histogram binarization. The probability is accumulated starting from the bin with the highest probability to the bin with the lowest probability. (The blue-italic numbers in the bins show the accumulating order, the black numbers on top of the bins show the accumulate probability). The bins having probability up to the first q-quantile of the sorted values are selected as transaction items (in the figures q is set to be 2.5). The first q-quantile is the data value where the cumulative distribution function crosses 1/q. Only few bins are selected if there are high peaks in the distributions (right).



(c) top-K bins binarization. The bins which have values among the top-K values are selected as transaction items (the blue numbers in the bins show the ranking order, K is set to 4). This binarization gives equally a reasonable amount of items for all images (both left and right have 4 items).

Fig. 2 Toy example of the three types of binarization methods. All figures on the left represent the binarizations of a flat distribution histogram. All figures on the right represent the binarizations of a distribution histogram with high peaks. The green bars are the bins in which their feature indexes are used as transaction items. The red bars are the discarded bins. For example, the transaction encoded from the histogram in Figure (c)-left is {2,6,8,9}.

which can be unfortunately very large. We investigated two solutions to make the mining process more efficient: (i) mining frequent patterns from the transposed data, (ii) reducing the number of items per transaction by applying multiple random projections.

Matrix transposition. [24] proposed to use the *Galois connection* property to solve the problem of mining patterns from high number of items with respect to the number of transactions. The principle is to transpose the original data matrix and then to extract the closed patterns from the transposed data matrix. Thanks to the Galois connection, we can infer the results that would be extracted from the initial data matrix by associating the closed patterns from the transposed matrix

Image I_i	*Trans.*$\mathcal{T}_i$
I_1	$\{a_1, a_2, a_3\}$
I_2	$\{a_1, a_4, a_6\}$
I_3	$\{a_1, a_7, a_9\}$
I_4	$\{a_2, a_3, a_6\}$
I_5	$\{a_4, a_5, a_8\}$

(a) Original database.

Items a_j	*Trans.* $\mathcal{T}_j$
a_1	$\{I_1, I_2, I_3\}$
a_2	$\{I_1, I_4\}$
a_3	$\{I_1, I_4\}$
a_4	$\{I_2, I_5\}$
a_5	$\{I_5\}$
a_6	$\{I_2, I_4\}$
a_7	$\{I_3\}$
a_8	$\{I_5\}$

(b) Transposed database.

Fig. 3 Toy example of the original transaction database in (a) and the transposition of the transaction database in (b).

with the closed patterns from the initial matrix. Finally, the same set of closed patterns with their frequencies are extracted, but much more efficiently. We use this transposition trick when the query dataset has few images. In other words, instead of considering an image as a data mining transaction with binary items, each image item is now considered as a data mining transaction containing a few images. Figure 3 explains the transposition of the data. In the following description of the method, we still use the terms of the original database.

Multiple random projections. It has been shown that representing high-dimensional data by multiple projections leads to good approximations of the data *e.g.* [14]. We propose a binarization framework consisting in (i) projecting the high dimensional feature space into several low dimensional sub-spaces by applying P random projections, (ii) binarizing features using top-K binarization, (iii) extracting and counting frequent patterns found in each sub-space (more details are in [33]). Note that the combination of the first two-steps bears similarity with local sensitive hashing (LSH) [10]. The difference is that in LSH a fixed threshold is used instead of top-K. We have mentioned the advantages of top-K over the fixed threshold in Section 3.2.1. Moreover, in practice, the projection is done by randomly selecting p visual words from the original BoW. This can be seen as projecting the original d-dimensional data to a p-dimensional subspace where the projection matrix is obtained by randomly selecting p basis vectors among the d ones from the original space (more complex projections have been investigated, without improving the performance). As an alternative to multiple random projections, we also experimented with principal component analysis (PCA) to reduce the dimensionality of input histograms. However, we obtained much worse results.

3.3 Promoting diversity

As mentioned in the introduction, it is more interesting for users to get as diverse as possible positive images. However, our scoring function tends to promote image belonging to large clusters. We observe that there are many groups of duplicates or near duplicate images (*i.e.* with slight differences in scaling, cropping, or colors *etc.*) in each query set, especially those queries such as 'flags', 'logos' in which there are not so much class variations. Having the top ranked images all the same, indeed, makes the result less appealing. To solve this, we make use of our groups of duplicates detection system described in [32] to cluster duplicates or near similar images together. We have shown in [32] that the system is very fast. The computational time for detecting groups of duplicates in a retrieval set of typically about 500 images is in a range of only a few tens of milliseconds. To describe the detection in brief, we target on representing each image as a very compact set of binary items (*i.e.* ten binary items). Note that we can use a very compact representation, since we are only interested in finding duplicates. This is not the case for modeling classes of objects which needs to have a richer image representation in order to capture large intra-class variations. We extract closed frequent patterns, with $minfr = 2$ (two images form a group of duplicates). From the set of frequent patterns, we keep only the very long patterns *e.g.* $l(x) > 8$. As mentioned earlier, the length of the patterns reflects the similarity of the images sharing the patterns (cf. Figure 8). We can control the similarity of the images in the group using the length threshold. The set of images sharing a long pattern is considered as a group of duplicates. More detail of groups of duplicate detection is given in [32]. If different groups overlap, meaning that they have common images, we merge those groups together. After the merging stage, we represent each group by the image which has the highest re-ranking score among other images in the group.

3.4 Scalability

The primary motivation for the proposed approach is the ability to re-rank the top few retrieved images such as provided by a first round of retrieval. However, in some case, a second search could be required to discover good candidates possibly buried far behind in the initially retrieved list. It can be done by extending the search to a second round of retrieval, providing a potentially large set of images, hence raising the question of the scalability of the approach. The overall complexity of the approach depends on the number of images

(N) and the number of patterns (P) in the following way: extracting the patterns is polynomial in N, computing patterns' scores (in the worst case) is $O(P \times N)$ or $O(P \times \log N)$ using a multi-treaded divide and conquer strategy (*e.g.* using a GPU). P depends on N but is bounded by the number of possible items (*i.e.* the number of visual words in the case of the bag-of-words representations). Finally, ranking the images based on the new score can be done in $O(N \log N)$. The bottleneck is hence the final ranking, which however can benefits from approximate or parallelized algorithms [23].

4 Experiments

This section provides the experimental validation of the proposed approach. We first describe the datasets and then present and analyze the experimental results.

For extracting closed frequent patterns, we use the LCM mining tool [29], setting $minfr = 2$ (*i.e.* a pattern is frequent if it appears in at least two images).

The visual binary attributes are obtained by binarizing bag-of-words (BoW) representations of images. To compute BoWs, we use multi-scale SIFT as local descriptors, as computed by the VLFeat library [31] (12 scales from 3 up to 14 pixels with the default step size of 2 pixels). The visual words are pooled from 3-level SPM [19] grids (1×1, 2×2, 4×4). To binarize the BoW representation we use the adaptive thresholding of [33].

4.1 Datasets and evaluation protocol

The approach is validated on the three following datasets: the INRIA Web Queries dataset [17], the QUAERO's visual concepts image dataset [27], and finally the eBay Motorbike dataset [8]. The rationale for choosing those 3 datasets is based upon their frequent usage in the recent computer vision literature, hence allowing us to provide experimental comparisons with state-of-the-art approaches such as [17,20,21,28].

The **INRIA Web Queries dataset** consists of top-ranked images from text queries returned by a web search engine. In total, there are 71,478 images from 353 queries, having about 250 images per query. For each query, about 40% of the images are relevant to the query. The queries are very diverse, ranging from general object classes or scenery classes such as 'car', 'bird', 'mountain', *etc.* to specific names of objects, places, events, or persons such as 'Nike Logo', 'Eiffel tower', 'Cannes festival', 'Cameron Diaz',*etc.* For each query, the annotation giving the relevance to the query is provided. The evaluation protocol is as follows. For each query, the images are sorted according to their ranking score. The *Average Precision AP* is calculated per each query and the *mean* Average Precision mAP is reported.

The **QUAERO's visual concepts image dataset** is similar to the INRIA Web Queries dataset in the way the images are obtained. The diversity of the queries, the types of images, as well as the intra-class variations within each query, are comparable to the ones of the INRIA Web Queries dataset. The main difference is that the number of images per concept is larger with about 950 images per concept. Moreover, since the dataset was created after the INRIA Web Queries dataset, web-search engines had been improved, resulting in cleaner data. For each concept, about 55% of the images are relevant images. The number of concepts is also larger consisting of 519 concepts. In total, this dataset contains 187,029 images. We follow the same evaluation protocol as used in [28] by using 100 concepts as test data to report the mAP over the 100 test concepts.

The **eBay Motorbike dataset** contains 5,245 images of different types of motorbikes collected from eBay. The dataset was designed for studying the problem of removing outlier images from a large dataset, rather than re-ranking web images. Since the images are for advertising, the quality of the images are better than web images. The dataset is also much cleaner with 97% of relevant images (*i.e.* motorbike images) compare to the INRIA Web Queries and the QUAERO's visual concepts datasets which have about 40-60% of relevant images per query. The metric used to evaluate the performance is the *Equal Error Rate* (EER).

4.2 On setting the system

This section justifies the chosen options regarding the algorithm and studies the sensitivity of the parameters. For doing this, we used a sub-part of the INRIA Web Queries dataset, by randomly selecting 30 queries (out of the 353 queries).

Transposed VS Random projections. As mentioned in Section 3.2.2, we proposed two methods for making frequent pattern mining more efficient, either by mining the transposed database or by doing random projections. We compared the two alternatives, computed from the same BoW representation. We run grid-search to obtain the best hyper-parameters (*i.e.* K for the first method, and P, p, K for the second) which gives the highest score for each method on a validation set.

In terms of performance, both methods are comparable (the difference in mAP is of about 1%). However,

in terms of complexity, the random projection approach is better since it is linear according to the number of images. The computational time of pattern extractions process between the two methods are compared in Table 4. Note that for the random projections method, all sub-processes (*i.e.* transaction encoding and pattern extraction of each random projection) are independent to each other, and therefore, the computation for the sub-processes can be run in parallel. In Table 4, the number of CPUs used is equal to the number of random projections which is 50. Note that for the matrix transposition approach, the computational time for extracting frequent patterns from some queries (*i.e.* 'Map World' and 'Logo Chelsea') is very large. This is due to the number of items (*i.e.* images) of some transactions (*i.e.* features) can be very large. The reason is that these queries contain a lot of duplicates or near duplicates, and some features can appear in all of them. On the other hand, the computational time of the random projection approach is quite stable across different queries, since the number of items (*i.e.* features) per transaction (*i.e.* images) is fixed. To conclude this experiment, when multi-process resource is available and especially when the number of images is large, the random projection method is a better choice. The remaining experiments use this approach.

Hyper-parameters. In this set of experiments, the influence of the parameters as well as the justification for the chosen values are given. There are three parameters to be considered in our approach (see Section 3.2 for their definitions): (i) the value K in the adaptive thresholding process, (ii) the dimensionality of the projected features p, and (iii) the number of random projections P. The performance is expected to increase with P since more information of the original representation is used. We first set P to the arbitrary value of 20, and run grid-search to find K and p. As shown in Figure 4, to obtain good performance, K has to be set to about 10% of p. The performance is not sensitive to the dimensionality of the projected features p, and a large range of values can be used. Nevertheless, small values for p are more desirable since they lead to small values for K. Remind that the mining complexity grows exponentially with K. To see the influence of P, we next fix $K = 20$ and $p = 800$ and evaluate the results with different values of P. As shown by Figure 5, the performance increases with the number of projections and saturated around $P = 50$.

Types of patterns. In this set of experiments, we investigate how the performance is related to the type of pattern, chosen as being either (i) frequent patterns, (ii) closed frequent patterns or (iii) maximal frequent patterns (see previous section for their definitions). As

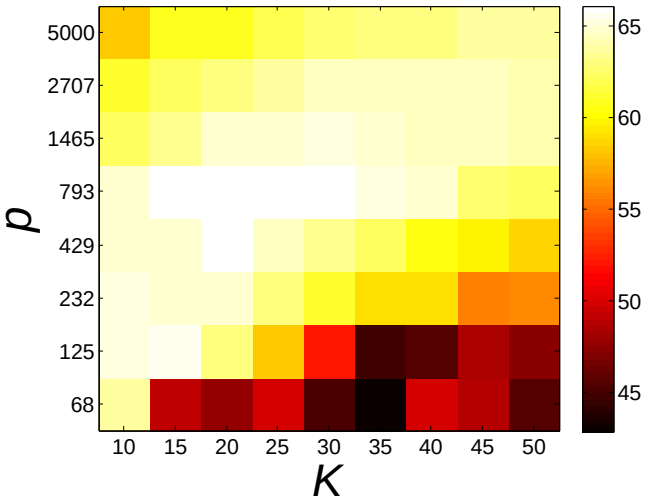


Fig. 4 mAP as a function of p and K.

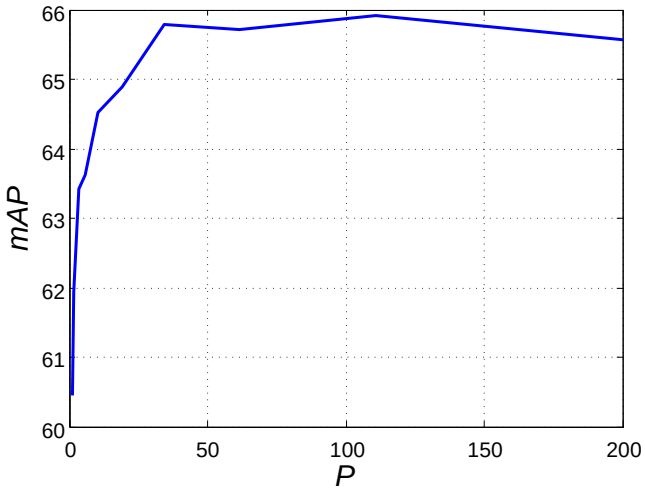


Fig. 5 mAP as a function of P.

Table 1 Mean Average Precision and number of patterns for each type of patterns.

Type	mAP	#Patterns
Frequent pattern	65.1	2M
Closed frequent pattern	65.7	250k
Maximal frequent pattern	54.5	87k

shown Table 1, frequent *closed* patterns performs as well as frequent patterns, despite there is about 5 times less patterns. This was expected as closed frequent patterns carry the same information as frequent patterns. We also observe that the number of *maximal* frequent patterns is 5 times lower than the number of closed frequent patterns, but the performance is significantly worse. We do not report here run-times, but they are approximately proportional to the number of patterns that are produced.

Table 2 Performance for different vocabulary sizes.

Vocab. size	100	1,000	2,000
mAP	62.9	65.7	**66.4**

Table 3 Comparison of different weighting schemes.

Weighting	*mAP*(%)
No weight	66.4
Frequency weighted	66.5
Length weighted	66.4
Area weighted	66.2
OriRank weighted	**68.8**

Vocabulary size. In this experiment, different vocabulary sizes are investigated. As shown by Table 2, the performance increases with the size of the vocabularies. However, increasing the size of the vocabulary also increases the encoding time so that we limited the size to 2,000 visual words.

Weighted scoring function. In this experiment, we investigated different weighting schemes used with Eq. 3. The results in Table 3, show that weighting the closed frequent patterns according to their frequency or to their length does not give any improvement. The explanation could be that longuer patterns are less frequent than the shorter patterns, and vice-versa. Giving more importance to long patterns, means that we explicitly also give more importance to the less frequent patterns which is not what we want. On the other hand, if we give more importance to frequent patterns, we explicitly also give more importance to shorter patterns. Weighting according to the area does not give any improvement either, as it is unlikely to find long closed frequent patterns with high frequency.

However, we observed that integrating the original ranking significantly improve the performance (+2.4%), see Table 3. We tried different functions besides the inverse of the ranking such as exponential functions but could not obtain better results. This weighting scheme is applied in the remaining experiments.

Best P random projections. As described in Section 3.2.2, we evaluated the performance given by the use of 5,000 random projections, and kept the best few projections which perform best. We made 3-fold cross-validation on the validation data and found that the performance saturates after adding the 20 best random projections. The improvement is of 2% *mAP* over using all the projections. Consequently, this feature selection not only makes the process faster but also improves performance. For the remaining experiments, we selected the best 20 (out of 5,000) random projections on the validation data and use them on the test data.

4.3 Computational time

Regardless of the time for computing the BoW representation – which can be done off-line and is the same for any re-ranking approach – we observed that extracting the patterns is linear according to the number of images. In addition, the computational time for scoring an image is linear with the number of patterns. The computational time for pattern extractions and scoring the images are given Table 4. As shown by Table 4, the approach is fast enough for being used on-the-fly. Moreover, the computation time is stable across different queries. On average, it takes about 0.15 seconds, which is comparable to the computational times reported by [28].

Table 4 Computational time (in sec.) for pattern extraction and image scoring), as well as number of patterns. Values are given both for the method using transposed data (Tr.) and the one using multiple random projections (Rp.). Computed on a validation set.

Query	Pat. Ext.(s)		Scoring(s)		#Pat.	
	Tr.	*Rp.*	*Tr.*	*Rp.*	*Tr.*	*Rp.*
Maradona	0.18	0.10	0.01	0.03	2k	7k
Giraffe	0.30	0.10	0.02	0.04	4k	8k
Times square	0.49	0.14	0.03	0.07	7k	14k
Grand canyon	11	0.12	0.05	0.06	10k	11k
Logo Chelsea	3.53	0.13	0.40	0.06	42k	11k
Map World	8.62	0.13	1.03	0.07	100k	12k
Mean 30 queries	*1.58*	*0.11*	0.17	0.04	17k	9k
Std. 30 queries	*6*	*0.02*	0.02	0.02	30k	3k

4.4 Performance evaluation and comparison with related state-of-the-art methods

In this section, we compare our re-ranking method with the state-of-the-art re-ranking methods available in the literature. In these experiments, the three hyper-parameters are set as: $P = 20$, $p = 800$, $K = 20$. The size of vocabularies is of 2,000. The original ranking is integrated excepted for the eBay Motorbike dataset for which the initial positions of the images are meaningless. The selection of the best random projections is activated. Note that we have cross-validated the settings on all datasets, and, interestingly, the choice of the hyper-parameters is very stable. Moreover, even the 20 best random projections apply well to all datasets. It explains why we eventually used the same settings for the 3 datasets.

INRIA Web Queries dataset. On this dataset, as shown by Table 5, our re-ranking approach improves the original search engine ranking by about 15% *mAP*

Table 5 Comparison to other re-ranking approaches on the INRIA Web Queries dataset.

Method	$mAP(\%)$
Original Search Engine	56.9
Query-ind.+Query-dep. [28]	65.5
LogReg (visual) [17]	64.9
SpecFilter+MRank [20]	**73.8**
Ours	72.2

Table 6 Comparison to other existing re-ranking approaches on the QUAERO's visual concepts image dataset.

Method	$mAP(\%)$
Original ranking	70.4
Query-ind.+Query-dep. [28]	72.7
Ours	**76.1**

Table 7 Comparison to other existing re-ranking approaches on the Ebay Motorbike dataset.

Method	$EER(\%)$
Implicit Shape Model [8]	71.0
FP+SVM [21]	72.6
Ours	**80.0**

and is better than any classifier-based approaches [17, 28]. Comparing with the graph ranking of [20], our approach has comparable performance in terms of mAP. However as explained in Section 2, our approach is real-time while the graph based ranking approach has a significantly higher complexity and cannot be used on-the-fly.

We can observe Figure 6 that after applying our re-ranking the top results are very clean compared to the original ranking, especially for queries in which images have small variations *e.g.* 'logos', 'maps', and 'flags' up to immediate variations *e.g.* 'landmarks', 'celebrities'. For queries in which images have a large range of diversity *e.g.* 'Generic objects', 'Animals', the top images are less clean. Nevertheless, our re-ranking approach can still improve the original ranking from the text-based search.

QUAERO's visual concepts image dataset. In this dataset, as shown by Table 6, our re-ranking approach improves the original search engine ranking by about 6% mAP, and is 3% mAP better than the best result reported on this dataset [28].

eBay Motorbike dataset. As shown by Table 7 and Figure 7, our approach is 7.4% EER better than the state-of-the-art approach of [21].

4.5 Promoting diversity

In this section, we use our groups of duplicate detection approach [32] to find groups of near duplicate images,

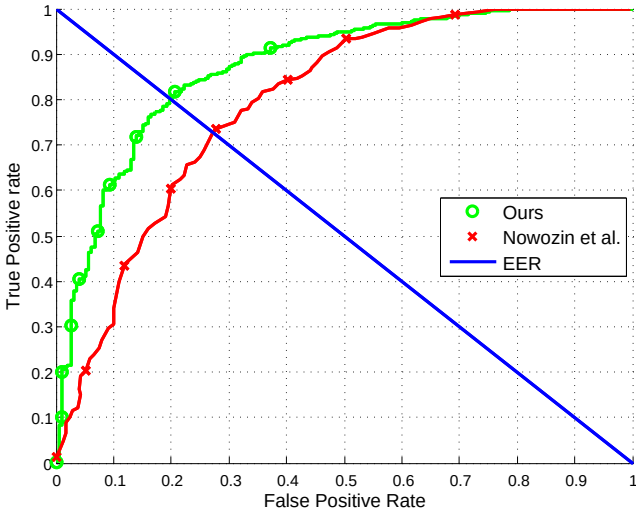


Fig. 7 ROC curves comparing our re-ranking system with [21] on the Ebay dataset.

hence increasing the diversity of the resulting images. We made an experiment to show that the similarity between images forming clusters increases according to the length of the patterns (Fig 8). Notice that the number of images in the clusters (the frequency) is inversely proportional to the length of the patterns.

For each group, we use the image with the highest re-ranking score as the instance to represent the group and display the results. In Fig. 6, we show some qualitative results illustrating the results after the re-ranking stage, and after the duplicate grouping. We observe that queries in which there are less diversity in the classes, such as logos (*e.g.* Nike logo), there are a large number of duplicates. After the grouping, we can see significant improvement in promoting the diversity. However, for classes with large intra-class variations such as animals (*e.g.* Monkey), we do not observe the difference between the top ranked images before and after the grouping stage. Nevertheless, the top-ranked images are already diversified.

5 Conclusions

In this paper, we propose a method to re-rank the images obtained from text-based image search engines. A new scoring function for re-ranking images in the context of text-based image retrieval is defined. Our method relies on the hypothesis that non-relevant images are more scattered than relevant ones, the proposed scoring function updates the original scores by measuring the amount of frequent patterns contained in images. Building on this assumption, our approach

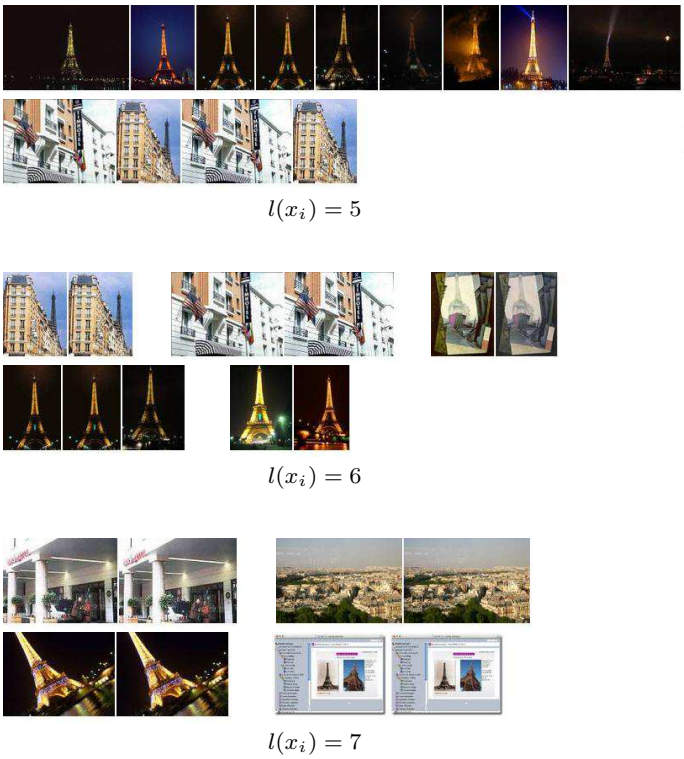


Fig. 8 Eiffel query: Sample of clusters (images stacked together) obtained at different length thresholds. The similarity between the images in each clusters is propotional to the length of the pattern. The images supporting patterns with $l(x_i) = 5$, are similar with variations in colors, shapes, aspect ratios, slight changes in background.The images supporting patterns with $l(x_i) = 7$, are identical to each other.

mines closed frequent patterns which are groups of image attributes shared by sets of images. Closed frequent patterns are likely to be discovered in the relevant images which have visually similar appearance, and not in the non-relevant images which do not share similarity to other images (negative images' appearances are very diverse). We showed that weighting closed frequent patterns based on the initial ranking position of the images supporting the patterns can improve the results of the non-weighted scheme. Regarding the efficiency of the approach, we show that we can reduce the mining complexity while maintaining the same quality of results by using multiple random projections. This allows to re-rank images even for very large sets of retrievals as the re-ranking computation scales linearly to the number of images. In addition of being fast enough for on-the-fly usage, the approach gives state-of-the-art results on three different challenging datasets. At the end, we show that we can apply our group of detection approach to group duplicates or very near duplicates together in order to promote diversity in the top ranked images.

Acknowledgements

This work was partially funded by the QUAERO project supported by OSEO, French State agency for innovation.

References

1. R. Agrawal and R. Srikant. Fast algorithms for mining association rules in large databases. In *VLDB*, pages 487–499, 1994.
2. R. J. Bayardo Jr. Efficiently mining long patterns from databases. In *ACM Sigmod Record*, volume 27. ACM, 1998.
3. N. Ben-Haim, B. Babenko, and S. Belongie. Improving web-based image search via content based clustering. In *CVPR Workshop*, 2006.
4. T. Berg and D. Forsyth. Animals on the web. In *CVPR*, 2006.
5. T. L. Berg and A. C. Berg. Finding iconic images. In *CVPR Workshop*, 2009.
6. R. Fergus, L. Fei-Fei, P. Perona, and A. Zisserman. Learning object categories from google's image search. In *ICCV*, 2005.
7. R. Fergus, P. Perona, and A. Zisserman. A visual category filter for google images. In *ECCV*, 2004.
8. M. Fritz and B. Schiele. Towards unsupervised discovery of visual categories. In *DAGM*, 2006.
9. M. Fritz and B. Schiele. Decomposition, discovery and detection of visual categories using topic models. In *CVPR*, 2008.
10. A. Gionis, P. Indyk, and R. Motwani. Similarity search in high dimensions via hashing. In *Proceedings of the 25th International Conference on Very Large Data Bases*, VLDB, 1999.
11. D. Grangier and S. Bengio. A discriminative kernel-based model to rank images from text queries. *PAMI*, 30:13711384, 2008.
12. K. Grauman and T. Darrell. The pyramid match kernel: Discriminative classification with sets of image features. In *ICCV*, 2005.
13. W. H. Hsu, L. S. Kennedy, and S.-F. Chang. Reranking methods for visual search. *Multimedia, IEEE*, 14:14–22, 2007.
14. H. Jégou, F. Perronnin, M. Douze, J. Sánchez, P. Pérez, and C. Schmid. Aggregating local image descriptors into compact codes. *IEEE PAMI*, 34(9):1704–1716, 2012.
15. Y. Jing and S. Baluja. Visualrank: Applying pagerank to large-scale image search. *PAMI*, 30:1877–1890, 2008.
16. L. Kennedy and M. Naaman. Generating diverse and representative image search results for landmarks. In *WWW*. ACM, 2008.
17. J. Krapac, M. Allan, J. Verbeek, and F. Jurie. Improving web-image search results using query-relative classifiers. In *CVPR*, 2010.
18. A. Ksibi, G. Feki, A. B. Ammar, and C. B. Amar. Effective diversification for ambiguous queries in social image retrieval. In *CAIP (2)*, volume 8048 of *Lecture Notes in Computer Science*, pages 571–578. Springer, 2013.
19. S. Lazebnik, C. Schmid, and J. Ponce. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *CVPR*, 2006.
20. W. Liu, Y. Jiang, J. Luo, and S. Chang. Noise resistant graph ranking for improved web image search. In *CVPR*, 2011.

21. S. Nowozin, K. Tsuda, T. Uno, T. Kudo, and G. BakIr. Weighted substructure mining for image analysis. In *CVPR*, 2007.
22. N. Pasquier, Y. Bastide, R. Taouil, and L. Lakhal. Discovering frequent closed itemsets for association rules. In *ICDT*. 1999.
23. S. Rajasekaran and J. H. Reif. Optimal and sublogarithmic time randomized parallel sorting algorithms. *SIAM Journal on Computing*, 18(3):594–607, 1989.
24. F. Rioult, J.-F. Boulicaut, B. Crémilleux, and J. Besson. Using transposition for pattern discovery from microarray data. In *8th ACM SIGMOD Workshop in DMKD*, 2003.
25. F. Schroff, A. Criminisi, and A. Zisserman. Harvesting image databases from the web. In *ICCV*, 2007.
26. J. Sivic and A. Zisserman. Video google: A text retrieval approach to object matching in videos. In *ICCV*, 2003.
27. Y. Su and F. Jurie. Visual word disambiguation by semantic contexts. In *ICCV*, 2011.
28. F. Thollard and G. Quénot. Content-based re-ranking of text-based image search results. In *ECIR*, 2013.
29. T. Uno, T. Asai, Y. Uchida, and H. Arimura. An efficient algorithm for enumerating closed patterns in transaction databases. In *DS*, 2004.
30. R. H. van Leuken, L. Garcia, X. Olivares, and R. van Zwol. Visual diversification of image search results. In *Proceedings of the 18th International Conference on World Wide Web*, WWW '09, pages 341–350, New York, NY, USA, 2009. ACM.
31. A. Vedaldi and B. Fulkerson. VLFeat: An open and portable library of computer vision algorithms. http://www.vlfeat.org/, 2008.
32. W. Voravuthikunchai, B. Crémilleux, and F. Jurie. Finding groups of duplicate images in very large dataset. In *BMVC*, pages 1–12, 2012.
33. W. Voravuthikunchai, B. Crémilleux, and F. Jurie. Histograms of pattern sets for image classification and object recognition. In *CVPR*, 2014.
34. C. Wallraven, B. Caputo, and A. B. A. Graf. Recognition with local features: the kernel recipe. In *ICCV*, 2003.
35. J. Wang, Y.-G. Jiang, and S.-F. Chang. Label diagnosis through self tuning for web image search. In *CVPR*, 2009.
36. K. Yang, M. Wang, X.-S. Hua, and H.-J. Zhang. Social image search with diverse relevance ranking. In S. Boll, Q. Tian, L. Z. 0001, Z. Zhang, and Y.-P. P. Chen, editors, *MMM*, volume 5916 of *Lecture Notes in Computer Science*, pages 174–184. Springer, 2010.

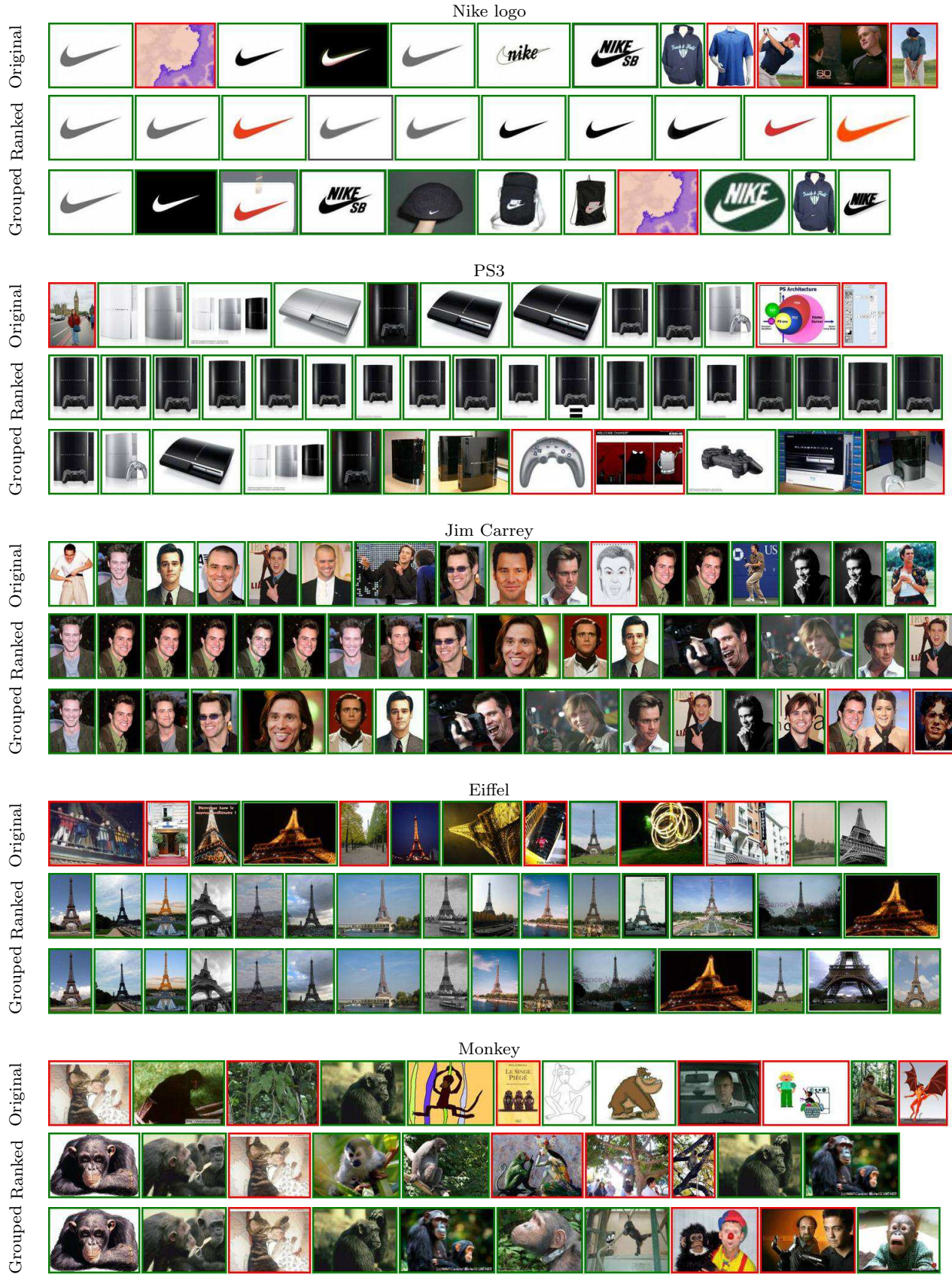


Fig. 6 Qualitative re-ranking results. We show the top-ranked results of five queries ranging in an order from small to high diversities categories (*i.e.* logos, specific objects, celebrities, specific landmarks, animals). The results after the re-ranking stage significantly improve the purity of the original ranking. However, the drawback of the re-ranking system is that it pulls all duplicate images up especially for queries with small diversities. After we group the duplicate images, we obtain better results in the aspect of promoting diversity.