Learning Query-dependent Prefilters for Scalable Image Retrieval

Lorenzo Torresani
Dartmouth College
Hanover, NH, USA
lorenzo@cs.dartmouth.edu

Martin Szummer
Microsoft Research
Cambridge, UK
szummer@microsoft.com

Andrew Fitzgibbon
Microsoft Research
Cambridge, UK
awf@microsoft.com

Abstract

We describe an algorithm for similar-image search which is designed to be efficient for extremely large collections of images. For each query, a small response set *is selected by a fast* prefilter*, after which a more accurate ranker may be applied to each image in the response set. We consider a class of prefilters comprising disjunctions of conjunctions ("*`OR`*s of* `AND`*s") of Boolean features.* `AND` *filters can be implemented efficiently using* skipped inverted files*, a key component of web-scale text search engines. These structures permit search in time proportional to the response set size. The prefilters are learned from training examples, and refined at query time to produce an approximately bounded response set.*

We cast prefiltering as an optimization problem: for each test query, select the `OR`*-of-*`AND` *filter which maximizes training-set recall for an adjustable bound on response set size. This may be efficiently implemented by selecting from a large pool of candidate conjunctions of Boolean features using a linear program relaxation. Tests on object class recognition show that this relatively simple filter is nevertheless powerful enough to capture some semantic information.*

1. Introduction

The application domain of this paper is in content-based image retrieval. In operation, a query image is presented to the system; images from a (possibly very large) database are retrieved and ranked according to similarity to the query; and presented to the user in rank order. We are interested in retrieval using high-level notions of similarity, in particular *object class* similarity, rather than *object instance* similarity. Ideally, we want the similarity to work not only on a specific set of classes, thus we consider a general same class/different class setting.

To make such queries efficient against large databases, it is common to use some form of fast *prefiltering* on the database before applying a more expensive analysis or ranking step. In order to define terminology, let us briefly outline the structure of a typical image search system:

A dictionary of *visual words* is defined, which maps image regions to integers in the range $1..W$. Values of W in the literature range from 10^3 to 10^6. Image i is represented by several (possibly overlapping) regions $r \in R_i$, whose centers and sizes (parameters θ_r) are defined either by evenly sampling, randomly sampling, or running an interest-point operator. Each region is mapped to a word w_r using the dictionary. Thus each image is associated to a multiset of words $\mathcal{W}_i = \{w_r \mid r \in R_i\}$. Denote by $n_i = \#\mathcal{W}_i$ the number of words in $\mathcal{W}_i$. The words are aggregated in the *word histogram* $\mathbf{h}_i \in \mathbb{N}^W$, where $\mathbf{h}_i := [h_{i1}, ..., h_{iW}] = [h_{iw}]_{w=1}^W$ and $h_{iw} := \#\{w_r \mid w_r = w, r \in R_i\}$. The parameters θ_r are typically also remembered and used for ranking after prefiltering, but shall be ignored in this paper, where each image is assumed to be represented by its histogram alone.

At query time, a new image q is presented, and the task is to retrieve images similar to q from the database. This is divided into two stages. The first stage is the *prefilter*: selecting images using a fast index structure such that the selected images have histograms likely to be similar to $\mathbf{h}_q$. We call this set of images the *response set*.

In a second stage, *ranking*, the images in the response set are processed in greater detail, for example by applying more sophisticated ranking schemes to the returned histograms, or returning to the image for more comprehensive analysis. This stage will generally require computation linear in the size of the response set, and it is clear that if this computation is expensive per image, then limiting the size of the response set is the key to a tractable implementation.

We design with scalability to web-size collections of billions of images in mind. Thus, the prefiltering stage must be fast (its computational cost should ideally be strongly sublinear in the database size); and it should have *bounded response* (it should return at most a certain proportion of the database). The algorithm we shall describe has cost proportional to the size of the response set, so the bounded response property also controls the computational cost. The primary design variable throughout the paper is the bound

"photograph"	"boolean"	"chihuahua"	response
1	0	0	46,300,000
0	1	0	10,200,000
1	1	0	161,000
0	0	1	7,870,000
1	1	1	881

Table 1. **Efficiency of `AND` queries (text search example).** Number of documents returned by live.com text search for `AND`s of terms, each of which is common. Although each term is common, skip pointers in the index allow the conjunctions to be recovered very efficiently: essentially the first 1000 for each can be generated in constant time. By formulating visual queries as `OR`s of `AND`s, we can gain the same efficiency in similar-image search.

on response ratio, τ. For a database of 1 billion images, one might set $\tau = .01\%$, and then process the 10^5 returned images for each query using a fast linear ranker such as *tf-idf* [17].

Our approach uses co-occurrences of visual words for retrieval. In contrast to previous work, however, we observe that the co-occurrences may be defined at query time, not at index time, providing that the database is implemented using *skipped inverted files* [15, 12]. For each word, the inverted file lists the images in which the word appears (possibly with a certain count). The images that contain a cooccurrence of several words can be found by merging the individual lists. The merging is very efficient since skip pointers have been inserted into each inverted list, giving fast access to the location of an image in the list. The key is that even for words which are individually very common, the `AND` of several such words will often be rare, and retrieval time is proportional to the number of results, not to the frequency of occurrence of the individual words. Table 1 shows a text-based example: search for the words "photograph" and "boolean" returns 46M and 10M responses respectively. Searching for documents in which they co-occur, however, returns just 161K documents, with an estimated time to return those documents of under 1 second. Adding another term, "chihuahua", further reduces the response set size, despite each term being common on its own. The upshot is that we can use `AND` queries in order to reduce response set size. In order to retrieve examples similar to the query image, we shall define `OR`s of `AND` queries to maximize recall while constraining the response set to be small.

In the following sections, we discuss previous approaches, and then define the problem of learning a "same-different" classifier for each new query, subject to the bounded response constraint. The main technical contributions of our work are:

1. To cast the prefiltering problem as an explicit search for filters which retrieve small response sets with high recall.
2. Learning the prefilters in a hypothesis class matched to what search engines support efficiently, namely `OR` of `AND` queries.
3. Learning the prefilters at query time, so building an efficient prefilter for each individual query image.

2. Related work

Our key measure in reviewing existing work is the *response ratio*–the average proportion of the database retrieved in the response set—which we denote by ρ.

Most existing schemes reduce ρ by making the dictionary size W large, so that individual words are less frequent, reducing the chance of a random overlap with an irrelevant image. Either the original quantization is generated using a large W, or a new vocabulary of metawords ("phrases" [21], "configurations" [20, 19], "min-hashes" [1, 2], "triplets" [23]) is defined. We shall assume that any such translation has already been performed, so that W becomes the size of the metaword dictionary. In each case, prefiltering is performed by taking the list of n_q (meta)words in the query image, and running a search on the database to find images containing any of the words. This is an n_q-way `OR` query, and, implemented using inverted files, will take time proportional to the sum of response sizes of the individual words. For large W, the response ratio for each word will be low on average[1], but the response set size over all words may still be very large.

The simplest form of this strategy is to quantize local descriptors to $W = 10^6$, and then retrieve all images with at least one word in common with the query. However, as Chum *et al*. report [1] in one application, about 43% of the database will be returned for a typical query ($\rho = 43\%$), and we have found similar numbers in other scenarios. For lower vocabulary sizes (e.g. $W < 100,000$), the response is typically 100% of the database. This is despite the very large vocabulary, which, as figure 1 indicates, is too finely quantized for object class recognition. On the other hand, a useful property of this approach is that with little extra cost, *tf-idf* ranking can be performed on the response set [17, 18], so it is useful as an intermediate phase between prefiltering and more complex ranking.

A second approach is to mine a training set for frequently occurring word tuples, for example using "frequent itemset mining" [21], or by spatial selection of word tuples [20, 10, 23], or combinations of the two [22]. Such approaches have two main difficulties. First, if the new metawords (i.e. the tuples) are indeed very frequent, the response ratio will be too high for prefiltering. This is normally corrected by eliminating "too common" metawords from consideration, at least at the prefiltering stage. An alternative is to look for discriminative metawords [19], however this has so far been applied only for class recognition of predefined classes, not to general "same/different" classification. A second

[1]Note that the shape of the word frequency histogram will not in general follow Zipf's law. For example, it will be flat if clustering was used to form the vocabulary, but may be closer to exponential if word cooccurrences are used to form metawords.

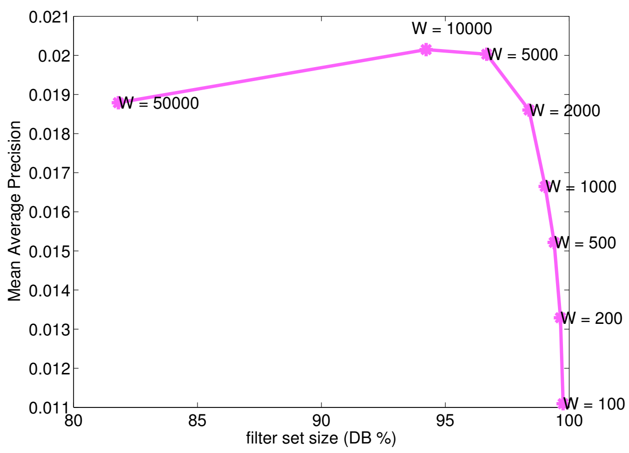

Figure 1. **Performance of *tf-idf* ranker on an object categorization benchmark.** Precision versus response set size obtained by varying the vocabulary size in a *tf-idf* ranker. The prefilter, based on single-word overlap, retrieves more than 80% of the database even at a vocabulary size of 50,000, at which point precision has begun to drop.

difficulty is that many more metawords may occur in a given image, increasing the chance of a single item overlap, and hence response set size.

An elegant extension of this concept is the min-hash system of Chum *et al.* [1, 2]. There, similarity between a pair of images i, j is defined by the *weighted histogram intersection*

$$\text{sim}(\mathbf{d}; i, j) := \frac{\mathbf{d}^\top \min(\mathbf{h}_i, \mathbf{h}_j)}{\mathbf{d}^\top \max(\mathbf{h}_i, \mathbf{h}_j)}. \tag{1}$$

for a predefined weight vector $\mathbf{d}$, and where $\min$ and $\max$ operate elementwise on vectors. The prefilter cleverly derived from this measure computes k hash values (or "sketch hashes") on the histograms of each image, and selects database images which match the query on at least one hash. The hash values may be thought of as being drawn from a very large vocabulary (of size W^2 or W^3), so the filter may again be treated as a single-word-overlap system. This technique produces the smallest response sets so far reported: for near-duplicate detection ($k = 64$), the response is $\rho = 0.005\%$ of the dataset, and for object instance detection ($k = 64$ to $k = 2680$), the ratio is between $\rho = 0.7\%$ and $\rho = 4\%$. As we show in this paper, this measure also provides advantages even as a measure of object *class* similarity, but with lower precision than our proposal.

A different approach is taken by Grauman *et al.* [5], where the histograms are not binned, but are represented by M locality-sensitive hash projections, where M is approximately the square root of the database size. In [5], M sorted lists of projection values are maintained, so that an approximate minimum of $|\mathbf{h}_q - \mathbf{h}_i|$ over all i can be found. Jain *et al.* [7] extend this representation to compute a learned distance metric $(\mathbf{h}_q - \mathbf{h}_i)^\top \mathbf{A}(\mathbf{h}_q - \mathbf{h}_i)$, which can be optimized with the same efficiency, and thus their problem statement is very close to ours. However, the memory requirement grows as $O(N\sqrt{N})$, as $\sqrt{N}$ sorted lists are required for the hash search. In contrast, the number of words stored per image is independent of database size, particularly for small vocabularies such as those used by min-hash.

3. Boolean "same or different" classifiers

The essence of the prefilter is that it decides which images in the database are the "same" as the query. For object instance recognition and near-duplicate detection, the definition of same versus different can be hand-engineered a priori based on histogram overlap and geometric consistency [1, 18]. For object class similarity, however, a training set of "same" and "different" training pairs is required. There is a considerable literature on distance metric learning, with [4] providing useful pointers into the literature. As our primary goal is efficiency, the "same/different" classifiers we consider must be constrained to be efficiently implementable on very large datasets. As illustrated above, and described in more detail in §3.3, classifiers which are `OR`s of `AND`s of indexed words are efficient, so we consider prefilters of this form.

The fundamental building block of our queries will be a *decision stump* of the form "word w occurs more than t times". We also include "fewer than" stumps, which are treated exactly analogously, so the description below covers only the "more than" case. We denote a generic stump by $C_s(\mathbf{h})$, whose parameters are a word index w_s and a threshold t_s. More formally C_s is a binary function of a histogram $\mathbf{h} = [h_1, ..., h_W]$ defined by

$$C_s(\mathbf{h}) = C(w_s, t_s; \mathbf{h}) := (h_{w_s} > t_s). \tag{2}$$

A stump classifies images i and j as "similar" if it is true for both, i.e.

$$C_s(i, j) := C_s(\mathbf{h}_i) \wedge C_s(\mathbf{h}_j). \tag{3}$$

We now define a "*phrase*" using a set of stumps $\mathcal{S}$, which applies to a histogram $\mathbf{h}$ by taking the `AND` of the stumps:

$$P_{\mathcal{S}}(\mathbf{h}) := \bigwedge_{s \in \mathcal{S}} C_s(\mathbf{h}_i). \tag{4}$$

A phrase marks two images as similar if it is present in both:

$$P_{\mathcal{S}}(i, j) := P_{\mathcal{S}}(\mathbf{h}_i) \wedge P_{\mathcal{S}}(\mathbf{h}_j). \tag{5}$$

Finally, a complete *classifier* is defined as an `OR` of `AND`s of stumps. The classifier is specified by a set Σ of sets of stumps as follows

$$Q_\Sigma(\mathbf{h}_i, \mathbf{h}_j) := \bigvee_{\mathcal{S} \in \Sigma} P_{\mathcal{S}}(\mathbf{h}_i, \mathbf{h}_j) \tag{6}$$

$$= \bigvee_{\mathcal{S} \in \Sigma} \bigwedge_{s \in \mathcal{S}} C_s(\mathbf{h}_i, \mathbf{h}_j). \tag{7}$$

This is an instance of at least two forms of classifier: decision forests [16] and set covering machines [13], and could be trained using standard methods. However such methods do not ensure bounded response, which is a key requirement of our prefilters. Let us assume, nevertheless, that such a classifier has been trained (our training procedure is described in §3.2) yielding a phrase pool Σ, and see how it can be used to bootstrap a bounded-response per-query classifier.

3.1. Tuning and bounding a classifier at query time

The key idea is that although the global classifier using the whole phrase pool Σ does not have bounded response, the phrases which define it are constructed so that each includes at most τ_C training examples, where τ_C is a small fraction of the response bound τ. By selecting a subset of the phrases in Σ on a per-query basis, we can control the size of the response set (as estimated on the training set) for that query.

We need a large training set of images to accurately measure the response ratio. In our work the set of images for which labels are provided is a very small subset of the entire training collection.

The training set comprises N images for which we compute visual word histograms $\{\mathbf{h}_1, ..., \mathbf{h}_N\}$. The more images we have, the more accurately we can estimate the response ratio. For a very small subset of these images, we additionally need labels indicating which pairs of images are similar. We represent the similarity labels as a set of M positive pair indices $\mathcal{M} = \{(i_m, j_m)\}_{m=1}^{M}$, where $(i,j) \in \mathcal{M}$ iff images i and j are similar. For a new query image q, let the *active set* of phrases be the subset A_q present in the query image, namely $A_q = \{\mathcal{S} \in \Sigma \mid P_{\mathcal{S}}(\mathbf{h}_q)\}$, which will be denoted $\{P_1, ..., P_K\}$ in the following. Then the evaluation of $Q_\Sigma(\mathbf{h}_q, \mathbf{h}_i)$ for any image i reduces to

$$Q_\Sigma(\mathbf{h}_q, \mathbf{h}_i) = \bigvee_{k=1}^{K} P_k(\mathbf{h}_i). \tag{8}$$

The goal is to choose a subset B of A_q such that the restricted classifier defined by B has good performance on the training set, subject to a maximum response of τ on the training set. If the training set statistics are similar to the database statistics, this low response will apply once the query is issued against the database, and the response ratio will be close to the target. A subset B is represented by a binary indicator vector $\mathbf{b} \in \{0,1\}^K$. The training set can be represented as an $N \times K$ binary matrix $\mathbf{F}$ where the presence of a phrase k in image i is indicated by $F_{ik} = P_k(\mathbf{h}_i)$. The positive pairs are denoted by an $M \times K$ matrix $\mathbf{T}$, with $T_{mk} = P_k(\mathbf{h}_{i_m}) \wedge P_k(\mathbf{h}_{j_m})$, i.e. T_{mk} is 1 when phrase k occurs in both images of pair m. The objective is then to choose $\mathbf{b}$ to maximize the number of true positives among the pairs:

$$\sum_{m=1}^{M} \mathbb{H}\Big[\big(\sum_{k=1}^{K} T_{mk} \cdot b_k\big) \geq 1\Big], \tag{9}$$

subject to the response constraint

$$\sum_{i=1}^{N} \sum_{k=1}^{K} F_{ik} \cdot b_k < \tau. \tag{10}$$

In the above, $\mathbb{H}$ is 1 when the inequality is satisfied and 0 otherwise; a pair is retrieved when at least one of the phrases in the pool Σ is selected by $\mathbf{b}$, since we are taking an `OR` the phrases. Note that the response constraint restricts the total *postings list length* of the response, i.e. the sum of the per-phrase response set sizes, so is a stricter constraint than the original, as discussed further below.

The above constraints may be expressed as a 0-1 integer program by introducing slack variables in the form of binary vector $\xi \in \{0,1\}^M$, yielding the problem

$$\min_{\mathbf{b},\xi} \; \mathbf{1}^\top \xi \qquad \text{subject to} \qquad \mathbf{T}\mathbf{b} \geq \mathbf{1} - \xi \tag{11}$$
$$\mathbf{1}^\top \mathbf{F}\mathbf{b} < \tau.$$

We evaluated several approaches to solving this program: various SAT solvers, an LP relaxation of the objective, and a greedy method analogous to the fitting algorithm for the Set Covering Machine [13]. The greedy algorithm yields the best results in practice, and its compute time is typically a second for a training set with $O(100K)$ images, so is our current method of choice.

Given $\mathbf{b}$, and hence the phrase subset $B \subset A_q \subset \Sigma$, the final query-specific filter applied to the database is

$$\bigvee_{\mathcal{S} \in B} P_{\mathcal{S}}(\mathbf{h}). \tag{12}$$

This filter is expressed as an `OR` of `AND`s. The response set defined by this classifier is retrieved and passed to the ranker.

3.2. Generating the phrase pool Σ

Selection of the phrase pool Σ is performed in an offline training process, analogous to bounding the response of a classifier at query time. We have used the same training sets for both, although offline sets could be larger since processing time is not constrained. Some preliminary observations are that each phrase $\mathcal{S}$ defines an axis-aligned box in histogram space, and that Σ defines a union of such boxes, which we refer to as the *positive region*.

The goal is to choose a set of phrases Σ which maximizes the number of true positives in the positive region:

$$TP = \sum_{(i,j) \in \mathcal{M}} Q_\Sigma(\mathbf{h}_i, \mathbf{h}_j). \tag{13}$$

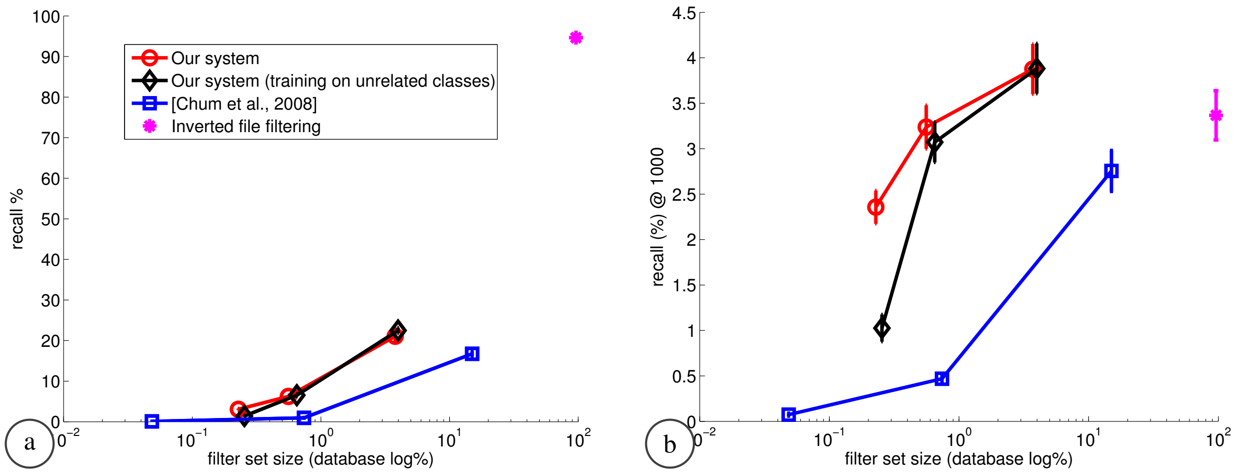

Figure 2. **Recall as a function of response ratio**. (a) Proportion of positives in the response set. (b) Proportion of positives at 1000 after *tf-idf* ranking. Our prefilter provides 5 times as many positives to the ranker than the only competitive system.

subject to an upper bound on the sizes of the individual boxes

$$\sum_i P_{\mathcal{S}}(\mathbf{h}_i) < \tau_C \quad \forall \mathcal{S} \in \Sigma. \tag{14}$$

Such an objective does not directly prevent overfitting, because placing the smallest possible boxes around each positive pair would maximize the objective. This is avoided because of an implicit regularization in the training algorithm we now describe. We experimented with several algorithms for phrase selection, and describe one—a variant of "bump hunting" [3]—which has proved effective in practice.

The algorithm takes one parameter—the phrase length L, and returns a set of phrases Σ, with the following properties: all positive pairs are recalled ($\forall_{(i,j)\in\mathcal{M}} Q_\Sigma(\mathbf{h}_i, \mathbf{h}_j) = 1$); each phrase satisfies (14); and the sizes of the boxes are in some sense "large".

Starting with $\Sigma = \oslash$, each positive pair $(i, j) \in \mathcal{M}$ is considered in turn, and those for which $Q_\Sigma(\mathbf{h}_i, \mathbf{h}_j)$ is not already true will generate a phrase to add to Σ.

For such an example (i, j), the phrase is built from a series of stumps in L iterations, adding one stump per iteration. Let $P^{r-1}(\mathbf{h})$ be the partial phrase at iteration r, to which will be added a stump. For each word in the dictionary, a candidate stump of the form $h_w > t$ is chosen as the smallest t (i.e. largest box) for which the current cumulative response ratio $\frac{1}{N}\sum_{i=1}^N P^{r-1}(\mathbf{h}_i) \wedge (h_{iw} > t)$ is below $\tau_C^{r/L}$. Thus the response constraint is progressively tightened until the final iteration, when it becomes equal to τ_C. From the $\approx W$ candidate stumps thus defined, the one which yields largest recall when added to $P^{r-1}(\mathbf{h})$ is selected.

3.3. Performance considerations

Our vocabulary sizes are very small ($W = 100$ for one test), so some of these words occur very frequently in the index. However the response bound ensures that these will be selected only as part of rare conjunctions, so skipped inverted files will still be efficient.

The stump filters may be implemented using inverted files by generating metawords for each used stump filter. Our examples use about 20K stumps (100 words with 255 possible threshold values), so this is the effective vocabulary size. Some of these metawords will be very common, for example "bin $w > 1$" for any w, but again will be selected only as part of conjunctions that are rare.

If more stumps were needed, or the thresholds were to be chosen at runtime, one would generate metawords of the form "$2^p r <$ bin $w \le 2^{p+1} r$" for appropriate values of r and p. Then a query of the form "bin $w > t$" can be executed in time $O(\log_2 t)$ for any t.

4. Experiments

Caltech256. We tested our system on the Caltech256 [6] dataset, which contains labeled images of 256 object categories. The system is evaluated on the task of retrieving images of the same category as the query image. We split the Caltech256 dataset into three distinct sets of images: query, training, and database images. The queries were drawn from a subset of 20 arbitrarily chosen categories, by randomly selecting 25 images from each of these categories. The training set was formed by randomly choosing 15 images from each of the 256 categories, thus yielding $M = \frac{15\cdot 14}{2} \times 256$ positive pairs. The images not belonging to the query or training set were used as database images to measure the retrieval performance of the system. Bag-of-words features were obtained by computing SIFT descriptors [11] at interest points detected with the Hessian-Affine detector [14]. SIFT descriptors were quantized into visual words with the vocab-

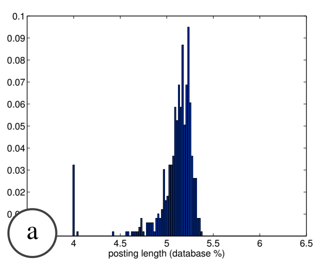

Figure 3. **Distribution of postings list length.** Test set histograms of postings list length: (a) ours for $\tau = 5\%$; (b) min-hash (using settings yielding average response set size roughly equal to 5%). The PLL is the cost of actually running the prefilter, the cost of running *tf-idf* on the output, and is the quantity that is bounded *on the training set* in (10). The fact that it is also close to the bound on the test set is felicitous.

ularies described in [9], which were computed by running the k-means clustering method on a large dataset of Flickr images. The *tf-idf* weights for each dictionary were learned on the Caltech256 training images. Figure 1 shows the performance obtained with a pure *tf-idf* ranker. The plot shows mean average precision (truncated at 1000) versus average size of the response set. Note that for tf-idf ranking the best precision is obtained using a dictionary of 10K words, which yields on average a response set containing over 95% of the database images. Increasing the dictionary size further reduces the response set but, as shown in this plot, hurts precision.

Flickr distractors. In order to test the scalability of our approach, we added to the database more than 800K Flickr photos taken from the collection described in [8]. In our evaluation we assumed that these images do not contain occurrences of the query object category and so they served purely as distractors. Another set of 200K Flickr photos from the same collection was added to the training set for the purpose of measuring ρ, the response ratio, during the phrase pool generation and the query-dependent refinement of the classifier.

Prefiltering evaluation. We evaluated our system under various choices of parameter values. The best results were obtained using a vocabulary of size $W = 100$ and phrases of length $L = 4$. We generated the pool of phrases by repeating the phrase generation procedure for values of τ_C varying in the set $\tau \times \{0.01, 0.03, 0.05, 0.07, 0.09\}$, for a given choice of τ. We report results for $\tau \in \{0.001, 0.005, 0.05\}$, corresponding to postings length bounds (10) ranging from 0.1% to 5% of the database. Phrases were selected at query time using the greedy algorithm of [13]. We compared our prefiltering algorithm with Chum's min-hash method. We tried several tunings of the parameters in their system (e.g. the dictionary size, the number of sketches, and the sketch sizes) and present the best obtained results here. We also evaluated the simple filtering produced by retrieving all images sharing at least one visual word with the query by means of inverted files (using the setting $W = 10^6$). Figure 2 summarizes the prefiltering performances of the methods. The plot in Figure 2(a) shows recall, i.e. the proportion of retrieved images belonging to the category of the query, as a function of the response set size (plotted on a log scale). The size of the response set is relevant for scalability, one of the primary concerns of this paper, and the recall within the response set places an upper bound on the recall of any subsequent ranker. Note that for the same response set size, our algorithm retrieves 5 times as many true positives as the min-hash method, while the mere sharing of one or more individual words performs very little filtering in practice. Figure 2(b) shows recall at 1000, obtained after ranking the images in the response set by tf-idf score (using $W = 10^6$, as this is the best setting for tf-idf ranking). This simulates the situation where a computationally expensive ranker will be run on the top-1000 candidate list obtained by fast tf-idf scoring. Our algorithm yields a recall @1000 higher than that of one-word-sharing via inverted files, and with a *much smaller* response set. Error bars are also show in these plots.

In order to evaluate the ability of our classifier to generalize to **never seen object classes**, we removed from our training set all images of the 20 query categories and then retrained our system. As shown on figure 2 (black diamond), performance was only slightly worse than that obtained by training on all classes. This suggests that our approach learns a same-different classifier which is fairly category-independent and thus usable in general settings where the test categories may be different from those present in the training set.

Figure 3 shows the distribution of the postings list length (PLL) obtained when setting $\tau = 5\%$. It is instructive to look at the PLL since it corresponds to the actual cost of running the prefilter and also bounds the cost of applying the tf-idf ranker on the resulting response set. Furthermore, the PLL is the quantity that we constrain on the training set in (10). Figure 3(a) indicates that even the *test set* postings list length is close to the target bound. The min-hash approach instead produces postings lists that vary widely in length (Figure 3(b)), thus yielding widely differing query execution time in practice.

Figure 4 is a box plot of the recall obtained on the held-out 20 query categories using $\tau = 0.5\%$. The red central mark of each box indicates the median recall on that category, the edges of the box are the 25th and 75th percentiles, and the whiskers extend to the most extreme points considered not to be outliers, while the outliers are plotted individually.

Table 2 summarizes the phrase selection statistics. From a pool of about 60K phrases, about 400 are typically present in

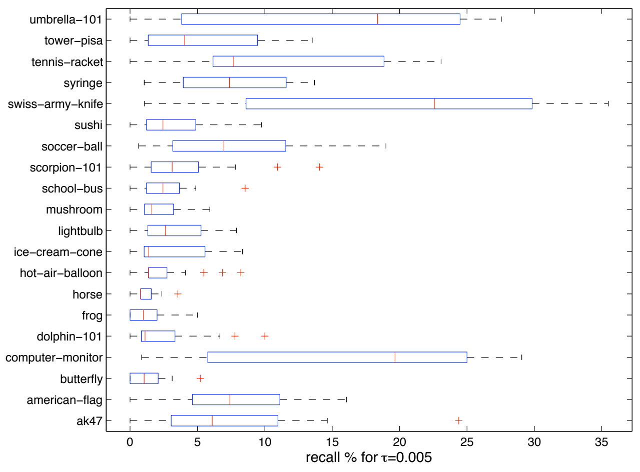

Figure 4. **Box plot of recall per individual query category.**

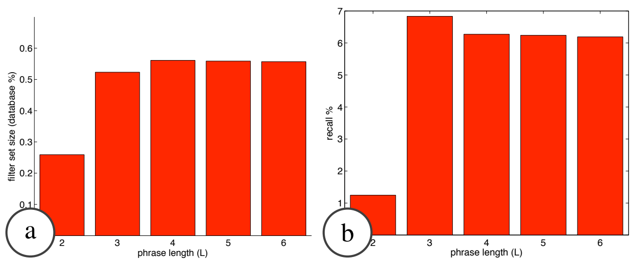

Figure 5. **Performance of our method for $\tau = 0.5\%$ as a function of phrase length:** (a) average response set size; (b) corresponding recall.

τ	$\#\Sigma$	Avg active $\#A_q$	Avg selected $\mathbf{1}^\top \mathbf{b}$
0.05	58533	470	9.3
0.005	57035	343	3.0
0.001	58533	344	2.3

Table 2. **Phrase generation and selection.** Typical queries contain 2–9 ORs, many fewer than the 64–256 of min-hash.

a given query, and the query-dependent selection chooses 2 to 9 of them to query the index.

Figure 5 shows the performance obtained by varying the length L of the learned phrases. Our algorithm is relatively insensitive to the choice of this parameter value.

Figure 6 presents some examples of query images and corresponding true positives retrieved by our algorithm. Note that in most examples, each selected phrase retrieves different relevant images. This suggests that our query-dependent prefilter tends to choose complementary phrases. Most of the images in these response sets are rather different in appearance from the queries and therefore could not be retrieved using a simple near-duplicate detection algorithm.

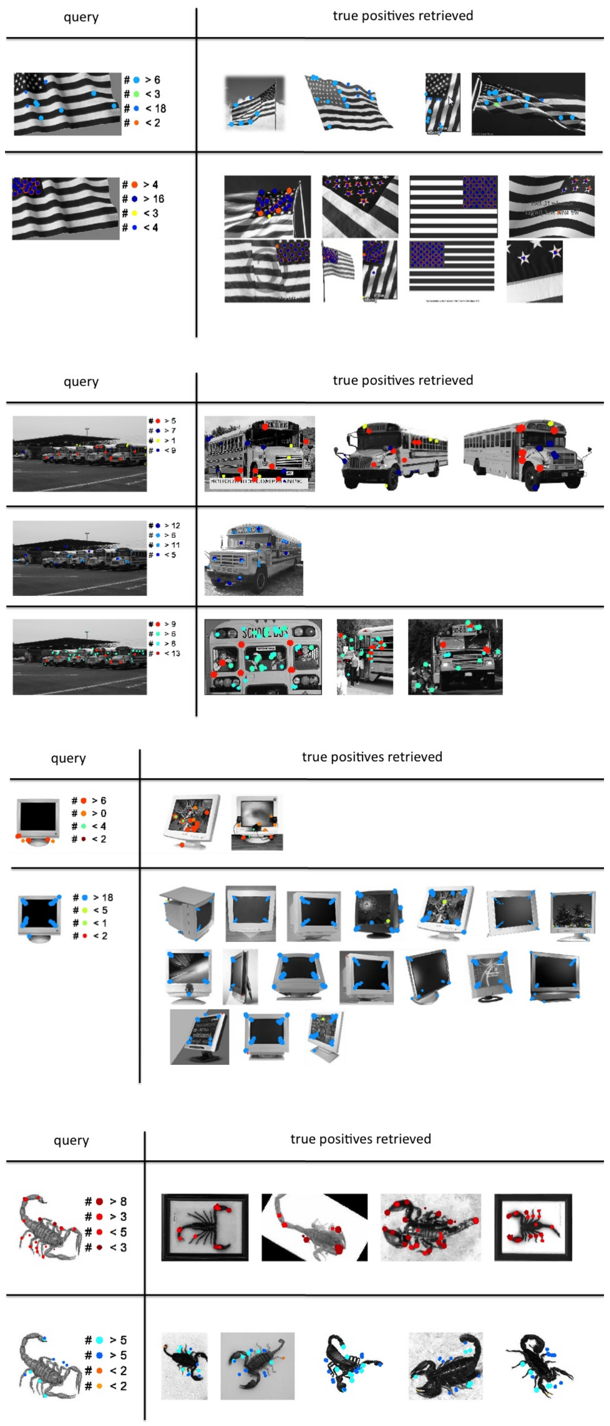

Figure 6. **Visualization of prefiltering results for $\tau = 0.5\%$.** The first column in each example shows repeated copies of the query image, one for each visual phrase selected at query time. The decision stumps used in each phrase are shown next to the query, with color-coded visual words. The true positive images retrieved by each phrase are shown in the second column.

5. Discussion

We have shown that constructing classifiers at query time allows prefiltering of a small number of candidates from a large corpus with reasonable recall, several times higher than that of the nearest competition. While previous approaches have looked for commonly occurring patterns, we look for rare ones with disjoint coverage of the query space.

Our system is designed to scale to billions of images. Here we reported results only on a million-image set. However, the indexing efficiencies should extend to a billion images—at the time of writing we are testing a 100 million-image index—but the outstanding question is whether the distractor statistics will become so flat that we are swamped by false positives. However, the relatively small degradation in performance we have observed from 30K to 1M images gives some hope that this effect will not be too pronounced.

The design separates the contents of the image index (which can be enormous and could take months to update with new visual words or their combinations) from the classifier that uses it (which can be learned dynamically at query time to form flexible new combinations of words from the index). Systems based on indexing only metawords cannot evolve incrementally and are too limited in adapting to new queries, classes or training data.

Currently the classifier is symmetric in the query and database images. However, we have much more CPU time available to analyze the query image, so it would be interesting to see if an *asymmetric* filter permits more accurate classification.

For future work, it would also be interesting to explore more general phrases. Our phrases were separable in that eq. 5 decomposed into two independent tests of query and database image. A straightforward extension would allow phrases to test whether the both images contained the same count of a visual word, within some tolerance.

6. Acknowledgements

This work was carried out while LT was at Microsoft Research Cambridge. We are grateful to Michael Isard, Ollie Williams, and Nick Craswell for useful discussion. Thanks to Herve Jegou for sharing data.

References

[1] O. Chum, J. Philbin, M. Isard, and A. Zisserman. Scalable near identical image and shot detection. In *Conf. Image and Video Retrieval (CIVR)*, pages 549–556, NY, 2007. ACM.

[2] O. Chum, J. Philbin, and A. Zisserman. Near duplicate image detection: min-hash and tf-idf weighting. In *British Machine Vision Conf.*, 2008.

[3] J. H. Friedman and N. I. Fisher. Bump hunting in high-dimensional data. *Statistics and Computing*, 9(2):123–143, Apr. 1999.

[4] A. Frome, Y. Singer, and J. Malik. Image retrieval and classification using local distance functions. In *Advances in Neural Information Processing Systems (NIPS)*, volume 19, pages 417–424, 2007.

[5] K. Grauman and T. Darrell. The pyramid match kernel: Efficient learning with sets of features. *Jrnl. of Machine Learning Research (JMLR)*, 8:725–760, Apr. 2007.

[6] G. Griffin, A. Holub, and P. Perona. Caltech-256 object category dataset. Technical Report 7694, California Institute of Technology, 2007.

[7] P. Jain, B. Kulis, and K. Grauman. Fast image search for learned metrics. In *Proc. Comp. Vision Pattern Recogn. (CVPR)*, pages 1–8, June 2008.

[8] H. Jegou, M. Douze, and C. Schmid. Hamming embedding and weak geometric consistency for large scale image search. In *European Conf. Computer Vision*, Oct. 2008.

[9] H. Jegou, H. Harzallah, and C. Schmid. A contextual dissimilarity measure for accurate and efficient image search. In *Proc. Comp. Vision Pattern Recogn. (CVPR)*, 2007.

[10] S. Lazebnik, C. Schmid, and J. Ponce. A discriminative framework for texture and object recognition using local image features. In J. Ponce, M. Hebert, C. Schmid, and A. Zisserman, editors, *Toward Category-Level Object Recognition*, pages 423–442. Springer, 2006.

[11] D. Lowe. Distinctive image features from scale-invariant keypoints. *Intl. Jrnl. of Computer Vision*, 60(2):91–110, 2004.

[12] C. D. Manning, P. Raghavan, and H. Schütze. *Introduction to Information Retrieval*. Cambridge University Press, 2008.

[13] M. Marchand and J. Shawe-Taylor. The set covering machine. *Journal of Machine Learning Research*, 3:723–746, 2002.

[14] K. Mikolajczyk and C. Schmid. Scale and affine invariant interest point detectors. *Intl. Jrnl. of Computer Vision*, 60(1):63–86, 2004.

[15] A. Moffat and J. Zobel. Self-indexing inverted files for fast text retrieval. *ACM Trans. Inf. Syst.*, 14(4):349–379, 1996.

[16] F. Moosmann, E. Nowak, and F. Jurie. Randomized clustering forests for image classification. *IEEE Trans. Pattern Analysis and Mach. Intell. (PAMI)*, 30(9):1632–1646, Sept. 2008.

[17] D. Nistér and H. Stewénius. Scalable recognition with a vocabulary tree. In *Proc. CVPR*, pages 2161–2168, 2006.

[18] J. Philbin, O. Chum, M. Isard, J. Sivic, and A. Zisserman. Object retrieval with large vocabularies and fast spatial matching. In *Proc. Comp. Vision Pattern Recogn. (CVPR)*, 2007.

[19] T. Quack, V. Ferrari, B. Leibe, and L. Van Gool. Efficient mining of frequent and distinctive feature configurations. In *Intl. Conf. Computer Vision*, pages 1–8, Oct. 2007.

[20] J. Sivic and A. Zisserman. Video data mining using configurations of viewpoint invariant regions. In *Proc. Comp. Vision Pattern Recogn. (CVPR)*, pages 488–495, 2004.

[21] J. Yuan, Y. Wu, and M. Yang. Discovery of collocation patterns: from visual words to visual phrases. In *Proc. Comp. Vision Pattern Recogn. (CVPR)*, pages 1–8, June 2007.

[22] Y.-T. Zheng, M. Zhao, S.-Y. Neo, T.-S. Chua, and Q. Tian. Visual synset: Towards a higher-level visual representation. In *Proc. Comp. Vision Pattern Recogn. (CVPR)*, 2008.

[23] C. L. Zitnick, J. Sun, R. Szeliski, and S. Winder. Object instance recognition using triplets of feature symbols. Technical Report MSR-TR-2007-53, Microsoft Research, 2007.