INRIA-LEARs participation to ImageCLEF 2009

Matthijs Douze, Matthieu Guillaumin, Thomas Mensink, Cordelia Schmid, Jakob Verbeek

INRIA-LEARs participation to ImageCLEF 2009

Matthijs Douze, Matthieu Guillaumin, Thomas Mensink, Cordelia Schmid, Jakob Verbeek
LEAR Team, INRIA Rhône-Alpes, 655 Avenue de l'Europe, 38330 Montbonnot, France
firstname.lastname@inria.fr

Abstract

We participated in the Photo Annotation and Photo Retrieval tasks of ImageCLEF 2009.

For the Photo Annotation task we compared TagProp, SVMs, and logistic discriminant (LD) models. TagProp is a nearest-neighbor based system that learns a distance measure between images to define the neighbors. In the second system a separate SVM is trained for each annotation word. The third system treats mutually exclusive terms more naturally by assigning a probabilities to the mutually exclusive terms that sum up to one. The experiments show that (i) both TagProp and SVMs benefit from a distance combination learned with TagProp, (ii) the TagProp system, which has very few trainable parameters, performs somewhat worse than SVM in terms of EEC and AUC but better than the SVM runs in terms of the hierarchical image annotation score (HS), and (iii) LD is best in terms of HS and close to the SVM run in terms of EEC and AUC.

In our experiments for the Photo Retrieval task we compare a system using only visual search, with systems that include a simple form of text matching, and/or duplicate removal to increase the diversity in the search results. For the visual search we use our image matching system that is efficient and yields state-of-the-art image retrieval results. From the evaluation of the results we find that the adding some form of text matching is crucial for retrieval, and that (unexpectedly) the duplicate removal step did not improve results.

Categories and Subject Descriptors

H.3 [**Information Storage and Retrieval**]: H.3.1 Content Analysis and Indexing; H.3.3 Information Search and Retrieval; H.3.4 Systems and Software; H.3.7 Digital Libraries; I.5 [**Pattern Recognition**]: I.5.1 Models; I.5.2 Design Methodology; I.5.4 Applications

General Terms

Measurement, Performance, Experimentation

Keywords

Image Categorization, Nearest Neighbors, Similarity Measures, Feature Selection

1 Introduction

In our participation to ImageCLEF we submitted runs in the Photo Annotation and Photo Retrieval tasks. Considering the best submitted run of each team, our systems achieved a second-best result for both tasks out of the 19 participants for each task. As the systems used for both tasks differ significantly we discuss them separately below.

2 Photo Annotation Task

In this section we present the system we used for the Photo Annotation task, and the results that were obtained. In Section 2.1, we first present our tag prediction model, and in Section 2.2 we describe the set of image features we used. We present our experimental results in Section 2.3, and our conclusions in Section 2.4.

2.1 A Discriminatively Trained Nearest Neighbor Model

Our TagProp method, short for Tag Propagation, is a new nearest neighbor type model that predicts tags by taking a weighted combination of the tag absence/presence among neighbors. For a more detailed presentation and additional experimental results see [2]. The main features of TagProp are the following. First, the weights for neighbors are based on their distance, and set automatically by maximizing the likelihood of annotations in a set of training images. Second, our model allows the integration of metric learning. This enables us to optimize a linear combination of several distance measures to define the neighbor weights for the tag prediction task. Third, TagProp includes word-specific logistic discriminant models. These models use the weighted nearest neighbor tag predictions as inputs and are able, using just two parameters per word, to boost or suppress the tag presence probabilities for words that are very frequent or very rare. This results in a significant increase in the number of words that are recalled, *i.e.* assigned to at least one test image. Our tag prediction model is conceptually simple, yet has been shown to outperform current state-of-the-art image annotation methods on standard data sets [2].

Weighted Nearest Neighbor Tag Prediction. Our goal is to predict the relevance of annotation tags for images. We assume that some visual similarity or distance measures between images are given, abstracting away from their precise definition. To model image annotations, we use Bernoulli models for each keyword. The dependencies between keywords in the training data are not explicitly modeled, but are implicitly exploited in our model.

We use $y_{iw} \in \{-1, +1\}$ to denote the absence/presence of keyword w for image i, hence encoding the image annotations. The tag presence prediction $p(y_{iw} = +1)$ for image i is a weighted sum over the training images, indexed by j:

$$p(y_{iw} = +1) = \sum_j \pi_{ij} p(y_{iw} = +1|j), \tag{1}$$

$$p(y_{iw} = +1|j) = \begin{cases} 1 - \epsilon & \text{for } y_{jw} = +1, \\ \epsilon & \text{otherwise,} \end{cases} \tag{2}$$

where π_{ij} denotes the weight of image j for predicting the tags of image i. We require that $\pi_{ij} \geq 0$, and $\sum_j \pi_{ij} = 1$. We use ϵ to avoid zero prediction probabilities, and in practice we set $\epsilon = 10^{-5}$. To estimate the parameters that control the weights π_{ij} we maximize the log-likelihood of the predictions of training annotations. Taking care to set the weight of training images to themselves to zero, *i.e.* $\pi_{ii} = 0$, our objective is to maximize

$$\mathcal{L} = \sum_{i,w} c_{iw} \ln p(y_{iw}), \tag{3}$$

where c_{iw} is a cost that takes into account the imbalance between keyword presence and absence. Indeed, in practice, we often have many more tag absences than presences. Depending on the source of the annotations of the training images, absences can be much noisier than presences. This is the case when most tags in annotations are relevant, but the annotation do not include all relevant tags. This happens in user annotations taken from photo sharing sites such as Flickr, as users will not annotate each image with all relevant tags, but rather just put a hand-full of mostly relevant tags. To balance tag absences and presences, we set $c_{iw} = 1/n^+$ if $y_{iw} = +1$, where n^+ is the total number of positive labels, and likewise $c_{iw} = 1/n^-$ when $y_{iw} = -1$.

The weights π_{ij} of training images j used when predicting tags for image i can be defined on their rank, or distance. Using the rank, we always assign a weight γ_k to the k-th nearest neighbour. Thus $\pi_{ij} = \gamma_k$ if j is the k-th nearest neighbor of i. Using the distances, we assign a weight relative to the similarity between the images. Which has the advantage that weights depend smoothly on the similarity, which is crucial if the distance is to be adjusted during training. Using distances, the weight of a training image j for an image i is defined as

$$\pi_{ij} = \frac{\exp\left(-d_{\mathbf{w}}(i,j)\right)}{\sum_{j'} \exp\left(-d_{\mathbf{w}}(i,j')\right)}, \tag{4}$$

where $d_{\mathbf{w}}(i,j) = \mathbf{w}^\top \mathbf{d}_{ij}$ with $\mathbf{d}_{ij}$ a vector of base distances between image i and j, and $\mathbf{w}$ contains the positive coefficients of the linear distance combination. Note that the number of parameters equals the number of base distances that are combined. When we use a single distance, referred to as the SD variant, $\mathbf{w}$ is a scalar that controls the decay of the weights with distance, and it is the only parameter of the model. When multiple distances are used, the variant is referred to as ML, for "metric learning".

Word-specific Logistic Discriminant Models. Weighted nearest neighbor approaches tend to have relatively low recall scores, which is easily understood as follows. In order to receive a high probability for the presence of a tag, it needs to be present among most neighbors with a significant weight. This, however, is unlikely to be the case for rare tags. Even if we are lucky enough to have a few neighbors annotated with the rare tag, we tend to predict the presence with a low probability.

To overcome this, we introduce word-specific logistic discriminant models that can boost the probability for rare tags and decrease it for very frequent ones. The logistic model uses weighted neighbor predictions by defining

$$p(y_{iw} = +1) = \sigma(\alpha_w x_{iw} + \beta_w), \tag{5}$$

$$x_{iw} = \sum_j \pi_{ij} y_{jw}, \tag{6}$$

where $\sigma(z) = (1 + \exp(-z))^{-1}$ and x_{iw} is the weighted average of annotations for tag w among the neighbors of i, which is equivalent to Eq. (1) up to an affine transformation. The word-specific models add two parameters to estimate for each word.

Training the Model. To reduce the computational cost of training the model, we do not compute all pairwise π_{ij}. Rather, for each i we compute them only over a large set, and assume the remaining π_{ij} to be zero. For each i, we select K neighbors such that we maximise $k^* = \min\{k_d\}$, where k_d is the largest neighbor rank for which neighbors 1 to k of base distance d are included among the selected neighbors. In this way we are likely to include all images with large π_{ij} regardless of the distance combination $\mathbf{w}$ that is learnt. Therefore, after determining these neighborhoods, our algorithm scales linearly with the number of training images.

The log-likelihood of the word-specific logistic discriminant with fixed π_{ij}, is concave in $\{\alpha_w, \beta_w\}$, and can be trained per keyword. To optimize the log-likelihood with respect to $\mathbf{w}$ we can again use the projected gradient method. In practice we estimate the parameters $\mathbf{w}$ and $\{\alpha_w, \beta_w\}$ in an alternating fashion. We observe rapid convergence, typically after alternating the maximization three times.

2.2 Image Features

We extract different types of features commonly used for image search and categorisation. We use two types of global image descriptors: Gist features [10], and color histograms with 16 bins in each color channel for RGB, LAB, HSV representations. Local features include SIFT [7] as well as a robust hue descriptor [14], both extracted densely on a multi-scale grid or on Harris-Laplacian interest points. Each local feature descriptor is quantized using k-means on samples from the training set. Images are then represented as a 'bag-of-words' histogram. All descriptors but Gist are L1-normalised and also computed in a spatial arrangement [6]. We compute the histograms over three horizontal regions of the image, and

concatenate them to form a new global descriptor, albeit one that encodes some information of the spatial layout of the image. To limit color histogram sizes, here, we reduced the quantization to 12 bins in each channel.

This results in 15 distinct descriptors, namely one Gist descriptor, 6 color histograms and 8 bag-of-features (2 detectors x 2 descriptors x 2 layouts). To compute the distances from the descriptors we follow previous work and use L2 as the base metric for Gist, L1 for global color histograms, and χ^2 for the others.

2.3 Experimental Results

For the details of the Photo Annotation task we refer to [9]. Below, before presenting the experimental results, we first briefly describe the systems that were used to generate the five runs that we submitted. We also give an overview of systems we used and the runtime of different components.

Submitted Runs. We submitted five runs: two TagProp runs, two SVM runs, and a run using a logistic discriminant model.

- **TagProp-SD.** For this run we used the TagProp model with a single distance, and with the word-specific logistic model. As distance we used a simple, equally weighted, sum of all 15 base distances (which we first all normalized to have a maximum value of 1). This model has a single parameter that controls the decay of the weights, and 2 parameters for each keyword (2×53), in total 107 parameters.

- **TagProp-ML.** For this run we used the TagProp model with metric learning, i.e. a (positive) linear combination of the 15 base distances is learned. In this model 15 parameters are learned to combine the distances, and 2 parameters for each keyword; 121 parameters in total.

- **SVM-SD.** For this run we have trained an SVM for each separate keyword. The kernel is a standard RBF kernel of the form $k(x, y) = \exp(-d(x, y)/\lambda)$, where d is the 'flat' distance combination also used in TagProp-SD, and λ is set to the average of all pairwise distances between training images. To obtain probabilities for the presence of each keyword, we learn a separate sigmoidal transformation for each keyword on the SVM output values. This model involves for each keyword 3.000 parameters for the SVM weight vector, plus one parameter for the bias term, and two for the sigmoid transformation. Thus in total $53 \times 3003 = 159.159$ parameters.

- **SVM-ML.** For this run we have first learned a linear distance combination using TagProp-ML, and then use this combined distance for the SVM. Again we use an RBF kernel, where distances are normalized by the average pairwise distance among all paris of training images. The number of parameters is the same as that of the SVM-SD system, plus the 15 parameters for the learned distance.

- **LD-ML.** This run uses the same kernel as SVM-ML. However, rather than learning an SVM per annotation term, we learned a multi-class logistic discriminant model. Mutually exclusive terms were grouped so that the classifier gives probabilities over these terms that sum to one. For other terms this run is very similar to SVM-ML, except that here the logistic loss is minimized rather than the hinge loss. The number of parameters is the same as for the SVM-ML run.

We note that due to lack of time we did not use cross-validation to optimize the regularization parameter of the SVM and LD models. Instead, we did not use regularization, and simply minimized the empirical loss. We expect a modest but significant increase in performance when optimizing over the regularization parameter. For details on SVM and LD we refer to machine learning textbooks, such as [1].

While all of our runs produce probabilities for the tag presence, only the last run produces multinomial probabilities for mutually exclusive image labels (e.g. for 'Spring', 'Summer', 'Fall', 'Winter', and 'No Season').

We did not apply any post-processing to enforce the requirements imposed by the hierarchical scoring method, which requires that for mutually exclusive image labels, exactly one of them has a score > 0.5. We expect that using some simple post-processing to enforce these requirements the image annotation scores, but not the EEC and AUC measures, could be improved.

Method	Run identifier	EEC	AUC	HS+A	HS-A
TagProp-SD	LEAR-44-2-1245581860906	0.304383	0.756649	0.757356	0.731446
TagProp-ML	LEAR-44-2-1245581967963	0.273385	0.793920	0.771921	0.747238
SVM-SD	LEAR-44-2-1245581505805	0.258642	0.813300	0.748797	0.722422
SVM-ML	LEAR-44-2-1245582451309	**0.249469**	**0.823105**	0.756161	0.730415
LD-ML	LEAR-44-2-1245582143586	0.256169	0.804713	**0.791800**	0.**769664**
Best in evaluation	various	0.234476	0.838699	0.829490	0.810325
Median in evaluation	various	0.395069	0.636666	0.693359	0.658824

Table 1: Performance of our five runs in terms of EEC (lower is better), AUC (higher is better), and the Hierarchical Scoring method with and without user agreement (higher is better). The best result among our five runs for each evaluation measure are printed in bold. The best and median result among all participants are also included.

Evaluation of Results. In Table 1 we present the evaluation scores that we obtained with our different runs. We also included the best and median result among all submitted runs.

First of all, we note that when we consider the best run of each of the 19 participating teams, our result is the second best in terms of EEC or AUC (and fifth in terms of the hierarchical annotation measures HS+A and HS-A). All of our submitted runs are clearly above the level of the median result.

Considering the results of our two TagProp runs, we see that there is a clear advantage of including metric learning to learn an appropriate combination of the 15 distance measures. This is in accordance with earlier experimental results [2]. Similarly, the SVM-ML run that uses the distance combination learned with TagProp also performs better than the SVM-SD run that uses the default distance combination. From cross-validation experiments we observed that the same holds for the LD classifier.

Interestingly, the SVM runs are better than the TagProp runs in terms of EEC and AUC, but worse in terms of the hierarchical scoring methods (HS+A and HS-A). The reason for this different evaluation result is not clear. It is possible that it is an effect of the class balancing in TagProp.

The LD-ML run results in probabilities that sum to one for mutually exclusive terms. Even though we did not force one of the probabilities to be > 0.5 we see that this runs scores significantly better than the others in terms of the HS+A and HS-A. However, in terms of EEC and AUC this run performs slightly worse than the SVM-ML run.

System Overview and Computational Cost. All systems were implemented using mixed C, C++ and Matlab code, and were run on a Standalone PC with four Q6800 processors at 2.93GHz and 8Gb of RAM.

The image annotation systems we implemented can be summarized as follows. In the "training" stage we process the 3.000 annotated training images to find the parameters of the model, and in the "testing" stage we use the model to predict the relevance of annotation terms for the 18.000 test images. Both stages proceed in a similar manner:

1. Compute global and local image features for each image.
2. **Training only:** Compute a k-means clustering for each type of local features.
3. Map the local features to the k-means centers to produce a 'bag of words' histogram for each image.
4. For each image compute distances to training images.
5. **TagProp only:** Determine nearest training images of each image.
6. **SVM+LD only:** For each image evaluate kernel function to all training images.
7. **Training only:** Find parameters of prediction model.
8. **Testing only:** Use prediction model to compute relevance of annotation terms.

Method	Training	Testing
TagProp-SD	7.9 s.	1.2 s.
TagProp-ML	20.4 s.	7.5 s.
SVM-SD	81.6 s.	63.4 s.
SVM-ML	138.6 s.	109.6s.
LD-ML	2h5m	74.5 s.

Table 2: Time needed to train the annotation models from 3.000 annotated images, and time needed to apply them to the 18.000 test iamges. The feature extraction stage is not included.

The feature extraction and quantization stage is common to all our submitted runs. Using our implementation the run times were as follows:

- Feature extraction: 5h16m for all 18.000 test images, for one test image: 1.05s.
- K-means local feature quantization (training images only): 5h08m
- Applying k-means quantization: 1h30m for all 18.000 test images, for one test image: 0.30s.
- Computing distances and neighborhoods: 1h42m + 5m, for one test image: 0.39s + 0.02s.

The time needed to train the different models, and to apply them to the 18.000 test images are given in Table 2. Including the feature extraction stage the total to annotate one test image is approximately 1.77s. which is almost entirely spent on feature extraction.

2.4 Conclusion

The performance of the TagProp runs are somewhat behind that of the SVM runs in terms of EEC and AUC, while the situation is reversed in terms of HS+A and HS-A. Among our runs the LD-ML run is best in terms of HS+A and HS-A, and not far behind the SVM-ML run in terms of EEC and AUC. All models benefit from using the distance combination learned with TagProp rather than the default distance combination. From these results we think it is an interesting option to integrate the metric learning within the LD learning framework, this is related to the multiple kernel learning framework [15].

In future work we plan to perform similar comparisons also on data sets where the annotations are very noisy, as they are often used in the image annotation literature (e.g. user-provided tags of Flickr images, as opposed to carefully assigned image labels as in this evaluation or the PASCAL VOC evaluation). Such an evaluation is interesting as it will show how tolerant the models are to noise in the image labels.

3 Photo Retrieval Task

For the details of the photo retrieval task we refer to [11].

To develop the mixed text/image retrieval tools, no validation set with a ground truth was available. We optimized our techniques by manually looking through the query results of the relatively small test set. We submitted distinct runs for promising techniques among which we could not tell the best one.

Our algorithm is structured as a series of filters that process a stream of potential results. The stream is output by an initial large-scale indexing system (a *source*) that performs queries in the 500 000-entries data set. The components for this chain may process images or text. They are described in the following.

3.1 Text processing components

Due to our limited expertise on text processing, we use simple word-based parsing techniques. We do not use any semantic analysis of the text. This means, in particular, that we do not exploit the cluster

descriptions. This word-based analysis is relatively effective for the Belgavox data set, because captions are made up to be used with word queries.

For queries, we used only the topic titles, *not the cluster titles*. Indeed, most of the cluster titles are combinations of the topic title, plus a *sink* that captures all other instances of the topic title (eg. the topic title "koekelberg" is associated with topic titles "fernand koekelberg", "dewael koekelberg" and the sink "koekelberg -fernand -dewael"). This means that we use the same information in Part 1 as in Part 2 of the queries.

For captions, we remove the most common form of meta-data, that usually includes the date and location of the picture and some terms of use. We detect it as a capitalized string followed with " : ". We then remove all punctuation and capital letters, split the string into words, and remove stop-words.

The text components are:

Text Source: the words occurring in the image captions are indexed in an inverted file. Results are ordered by the number of occurrences of the query topic (or the minimum of occurrences if the topic title contains several words). Ex-aequo captions are ordered randomly.

Text Filter: an input image is removed if it has no caption or a query word is missing from the caption.

Text Duplicate Filter: input captions are compared with the captions of images that have already been returned as final results. If the distance is below a threshold t_{txt}, the captions are considered too similar and the image is removed. The distance we use is a letter edit distance (implemented as a discrete time warping), normalized by the length of the longest of the two captions. Thus, this is an approximate duplicate filter.

3.2 Image processing components

3.2.1 Pre-processing

All the images are pre-processed to remove white margins and calibration patterns. Without this, images could be considered similar only because of their patterns.

These patterns are easy to recognize because they are always at a border of the image and there are only a few kinds of patterns: a CMYK version and a RGB version.

3.2.2 Image matching

Our baseline system builds upon the BOF image querying method [13] and recent extensions [4, 5, 12]. In the following we briefly describe the steps used here.

Local descriptors and assignment. We extract image regions with the Hessian-affine detector [8] and compute SIFT descriptors [7] for these regions. To obtain a bag-of-features representation for an image, we assign each descriptor to the closest visual word (Euclidean distance) from a visual vocabulary. The visual vocabulary, of size k, is obtained by k-means clustering performed on an independent data set of Flickr images. Such a nearest-neighbor quantizer, which assigns an index $q(x)$ to a descriptor x, implicitly divides the feature space into *cells*, i.e., the regions of a Voronoi diagram corresponding to the space partitioning.

Hamming Embedding (HE). HE provides a more precise representation of the descriptors than only the quantized index [4], i.e., it adds a compact binary representation. This representation subdivides each cell associated with a given visual word into regions. Associating a binary signature $s(x)$ with a descriptor x refines the descriptor matching, as two descriptors x and y match if they are assigned to the same visual word, i.e., if $q(x) = q(y)$, *and* if the Hamming distance $h(s(x), s(y))$ between their binary signatures is lower or equal than a threshold h_t. We set the signature length to 64 bit.

Weightings. The histogram of visual word occurrences is weighted using the TF-IDF weighting scheme of [13] and subsequently normalized with the L2 norm. The TF-IDF weighting factor of a matching score is

$$w_{\mathrm{tfidf}}(x) = \log\left(\frac{N}{N_{q(x)}}\right)^2 \tag{7}$$

where N is the total number of descriptors in the database and N_w is the number of descriptors in the data set assigned to visual word w. This weighting reduces the influence of visual words that occur often in the whole database.

In [4], the Hamming distance results in a binary decision, i.e., two descriptors match or not. However, the distance reflects the closeness of descriptors and should be taken into account. Since we have higher confidence in smaller distances, we weight them with a higher score.

The weight associated with a Hamming distance between binary signatures $s(x)$ and $s(y)$ is obtained with a Gaussian function [5]:

$$w_{\mathrm{hd}}(x, y) = \exp\left(-\frac{h(s(x), s(y))^2}{\sigma^2}\right). \tag{8}$$

where $\sigma = 16$.

We also apply two kinds of weightings that take into account the observed frequency of the descriptors. This reduces the influence of repeating patterns in images (intra-image burstiness) and re-occurring patterns in the whole database (inter-image bustiness) [5]. In contrast with TF-IDF weighting, these weightings take into account descriptor distances and can be applied when all the query descriptors have been matched.

The point matching scores, weighted with $w_{\mathrm{tfidf}}(x)$, w_{hd} and the burstiness corrections, are summed up to produce an image matching score s.

Differences with a "same-scene" recognition method. We found that the matching performed by the default method is too strict: only images of the exact same scene are retrieved. In this case, we are more interested in image category recognition. Therefore we relaxed the image matching by:

- using a coarse visual word quantization ($k = 1000$ visual words instead of 20 000 or 200 000);
- using a permissive Hamming Threshold ($h_t = 32$ instead of 24). This lets through 1/2 of the point matches, instead of 6 %;
- not performing the Weak Geometry Check. This allows large re-combinations of the scene geometry.

3.2.3 Indexing

In order to compute the image matching score efficiently, the set of descriptors of the image data set is stored in a structure similar to the inverted file used in text retrieval, and used in the image search system of [13]. This structure is composed of k lists of descriptor entries, each corresponding to a visual word.

For a given visual word, the list contains an entry per descriptor that was assigned to this visual word. The entry contains the index of the image where the point was detected and the binary signature associated with the descriptor.

Compared to an exhaustive scan of the data set descriptors, this greatly reduces the complexity, because only the descriptors assigned to the same visual word as the query descriptor are processed.

3.2.4 Components

We developed the following image components:

image source: the data set images are indexed with an inverted file on their visual words. The query images are the example images of a topic.

image duplicate filter: each time a result image is output, it is added to an inverted file. The filter only lets through images that are different enough from the ones in the inverted file (image matching score $s_{\mathrm{im}} < t_{\mathrm{im}}$). This is an approximate duplicate filter.

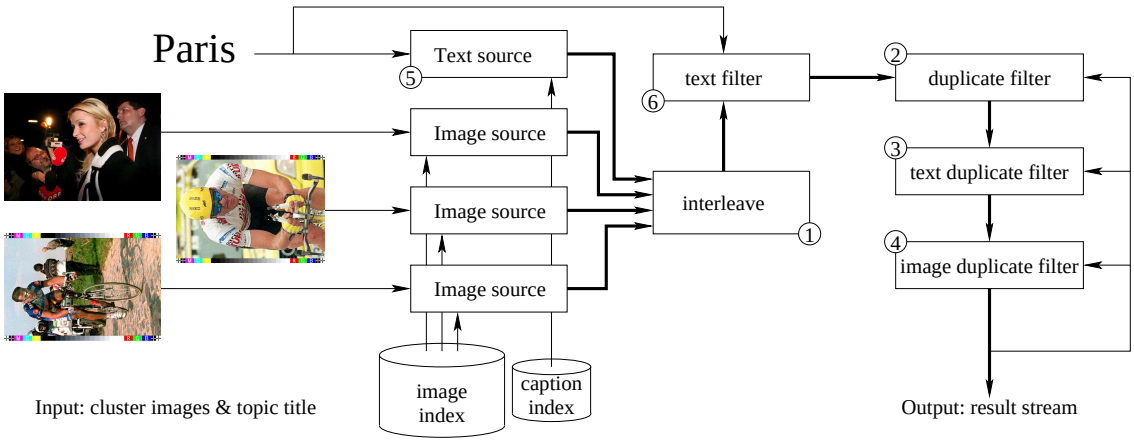

Figure 1: Structure of the retrieval system for Run 2. Thick arrows represent image streams.

We also developed a facial similarity component based on [3]. However, for most images, this technique is unreliable compared to the precise information given by the names in the captions.

3.3 Experiments

Here we describe our retrieval system and comment on its results.

3.3.1 Structure

Figure 1 shows the structure of our retrieval system for run 2. In addition to the components described above, we use an *interleave* component ① that interleaves the images from several input streams, and a *duplicate filter* ② that filters out images that have been returned already.

Interleaving the results from the various sources improves the diversity in the first results of the system. The duplicate filters are ordered from fastest to slowest: first the exact duplicate filter ②, then the two approximate ones, ③ and ④. The approximate duplicate filters are intended to increase the diversity of the results by filtering out very similar output images.

The runtime for the whole process is in the order of a few seconds per query. The most expensive components are the image sources, runtimes are analysed in [4]. Indexing the 500.000 images with 9 machines took about 6 hours.

3.3.2 The runs

Run 2 is the most complete version of our retrieval system. The other runs are described as restrictions of this basic run:

Run 2: The thresholds for the approximate duplicate filters are tight ($t_{\text{txt}} = 0.2$, $t_{\text{im}} = 30$), which means that images that are only slightly similar are let through.

Run 3: here the thresholds are adjusted so that more images are recognized as duplicates ($t_{\text{txt}} = 1$, $t_{\text{im}} = 24$).

Run 1: the text source component ⑤ is removed. The text filter ⑥ remains in effect.

Run 5: this one is similar to run 1, with the approximate duplicate filters ③ and ④ omitted as well.

Run 4: this run removes all text components ⑤, ⑥, ③.

3.3.3 Results

Text+image runs. Relative to other participants, our results are worse on part 1 (rank 10) of the queries than on part 2 (rank 4). This shows that other groups were able to use the cluster titles, that we ignored, in a meaningful way.

Run 5 ($F_{\text{measure}} = 0.762$) obtains better results than Run 1 ($F_{\text{measure}} = 0.758$). Thus, it seems that our attempts to increase the diversity were not fruitful.

Our runs with text sources, Run 2 and Run 3 (maximum $F_{\text{measure}} = 0.737$), have clearly lower results than the image-only sources. This is probably due to our simplistic text analysis: we use no more information than a simple "grep -c" could give. In particular, we cannot order captions that have the same number of occurrences of a query word.

Overall, for text+image, our runs are well behind the best ones from Xerox-SAS ($F_{\text{measure}} = 0.81$), but perform well compared to other participants.

Image only run. Run 4 got the best result for image-only queries, albeit with a small number of participants: we obtain $F_{\text{measure}} = 0.22$ vs. $F_{\text{measure}} = 0.17$ for the runner-up. This is due to our very effective (and efficient) large-scale image indexing system.

References

[1] C. Bishop. *Pattern recognition and machine learning*. Spinger-Verlag, 2006.

[2] M. Guillaumin, T. Mensink, J. Verbeek, and C. Schmid. Tagprop: Discriminative metric learning in nearest neighbor models for image auto-annotation. In *ICCV*, 2009.

[3] M. Guillaumin, J. Verbeek, and C. Schmid. Is that you? Metric learning approaches for face identification. In *ICCV*, 2009.

[4] H. Jégou, M. Douze, and C. Schmid. Hamming embedding and weak geometric consistency for large scale image search. In *ECCV*, pages 304–317, 2008.

[5] H. Jégou, M. Douze, and C. Schmid. On the burstiness of visual elements. In *CVPR*, 2009.

[6] S. Lazebnik, C. Schmid, and J. Ponce. Beyond bags of features: spatial pyramid matching for recognizing natural scene categories. In *CVPR*, 2006.

[7] D. Lowe. Distinctive image features from scale-invariant keypoints. *IJCV*, 60(2):91–110, 2004.

[8] K. Mikolajczyk and C. Schmid. Scale and affine invariant interest point detectors. *IJCV*, 60(1):63–86, 2004.

[9] S. Nowak and P. Dunker. Overview of the CLEF 2009 large scale visual concept detection and annotation task. In *CLEF working notes 2009*, Corfu, Greece, 2009.

[10] A. Oliva and A. Torralba. Modeling the shape of the scene: a holistic representation of the spatial envelope. *IJCV*, 42(3):145–175, 2001.

[11] M. Paramita, M. Sanderson, and P. Clough. Diversity in photo retrieval: overview of the ImageCLEFPhoto task 2009. In *CLEF working notes 2009*, Corfu, Greece, 2009.

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

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

[14] J. van de Weijer and C. Schmid. Coloring local feature extraction. In *ECCV*, 2006.

[15] M. Varma and D. Ray. Learning the discriminative power-invariance trade-off. In *ICCV*, 2007.