Augmenting Bag-of-Words – Category Specific Features and Concept Reasoning

Eugene Mbanya[1], Christian Hentschel[1], Sebastian Gerke[1], Mohan Liu[1], Andreas Nürnberger[2], and Patrick Ndjiki-Nya[1]

[1] Fraunhofer Institute for Telecommunications, Heinrich Hertz Institute
{eugene.mbanya|christian.hentschel|sebastian.gerke|mohan.liu|patrick.ndjiki-nya}@hhi.fraunhofer.de

[2] Data & Knowledge Engineering Group, Faculty of Computer Science, Otto-von-Guericke-University Magdeburg, Germany
andreas.nuernberger@ovgu.de

Abstract. In this paper we present our approach to the 2010 ImageClef PhotoAnnotation task. Based on the well-known bag-of-words approach we suggest two extensions. First, we analyzed the impact of category specific features and classifiers. In order to classify quality-related image categories we implemented a sharpness measure and use this as additional feature in the classification process. Second, we propose a post-classification step, which is based on the observation that many of the categories should be considered as being related to each other: Some categories exclude or allow for inference to others. We incorporate inference and exclusion rules by refining the classification results. The results we obtain show that both extensions can provide a classification performance increase when compared the the standard BoW approach.

1 Introduction

Visual data such as image and video represents the fastest growing data in the Internet today. Photo communities such as Flickr host more than 4 billion photos[3]. Efficient retrieval methods are thus required to provide access to this vast amount of information, which would otherwise be useless. The sheer amount of data, however, renders manual annotation impossible and demands for automatic approaches. The ImageClef Photo Annotation Task is an annual competition, which gathers researchers to meet this challenge and provide solutions for automatic classification of photos taken from the Flickr community into different categories. In this paper we describe our approach to the 2010 ImageClef PhotoAnnotation Task [4].

We follow the widely-used visual codebook approach [5]. We extract local image descriptors, which are an extension of the standard SIFT algorithm by Lowe[3]. As many of the visual concepts provided within the task show a uniform distribution in texture space, instead of extracting features at scale-space

[3] http://blog.flickr.net/en/2009/10/12/4000000000/

extrema the features are densely extracted by computing the descriptor at fixed grid of feature points. Following the visual codebook paradigm, we cluster similar features into groups, each represented by a visual (code)word. This is done by vector quantizing a subset of all features of all images in the training set. The codebook is the set of all visual words. It is used to describe an image in terms of the codeword frequency distribution by assigning features to codewords. Thus, for each image a simple histogram of codewords provides a compact feature representation. An extension of this approach was presented in [2]. As some concepts are more likely to be present in specific regions of an image (e.g. sky is more likely to be present in the upper image part) the authors suggest to split an image into fixed subregions. Different resolutions of subregions are aggregated into a so-called spatial pyramid. Histograms of codewords are computed for each of these regions and distances between images are computed region-wise.

While the described process has the strong advantage of being generic with respect to the extracted features, concept specific peculiarities can hardly be captured. We show that a weighted combination of generic and concept-specific features can increase the overall classification accuracy. As an example we extract a sharpness measure to handle quality-based image categories (such as image blur).

Category learning is done by training an kernel-based classifier. Support vector machines have been widely used for image classification tasks. We apply a χ^2-metric for distance computation as this has shown to provide good results in histogram classification [9].

Finally, we propose a classification post-processing step, which is motivated by the idea of category co-occurrences. An analysis of the different image categories in the training set has shown, that many of the categories interfere with each other in sense that the presence of one category gives evidence to or excludes the presence of another. Category reasoning is implemented as a post-classification refinement step.

In the following, we describe the aforementioned steps in more detail: Section 2 describes the feature extraction and codebook generation process in detail. In section 3 we describe the process of category learning and image classification while section 4 provides information on the aforementioned category post-processing. Finally, section 5 summarizes this paper by giving some results and providing an outlook to future work.

2 Feature Extraction

For classification, two types of features have been used. One is the so-called OpponentSIFT [8] feature, an extension of the Scale-Invariant-Feature Transform (SIFT, [3]) features to the opponent color space while the other one is a sharpness measure for an image to be used in sharpness and blur related categories. As shown in [8], the OpponentSIFT feature with dense sampling was the best performing single feature for an annotation task and therefore has been chosen as a baseline for our submission.

2.1 OpponentSIFT

The general principle of the OpponentSIFT based feature extraction is depicted in figure 1. First, the image is converted to the opponent color space. The opponent color space is given by the following definition from the RGB color space:

$$\begin{pmatrix} O_1 \\ O_2 \\ O_3 \end{pmatrix} = \begin{pmatrix} \frac{R-G}{\sqrt{2}} \\ \frac{R+G-2B}{\sqrt{6}} \\ \frac{R+G+B}{\sqrt{3}} \end{pmatrix} \tag{1}$$

Channel O_3 represents the intensity channel, while O_1 and O_2 represent the color components. Due to the subtraction in the first two channels, these are shift-invariant with respect to light intensity but not scale-invariant [8].

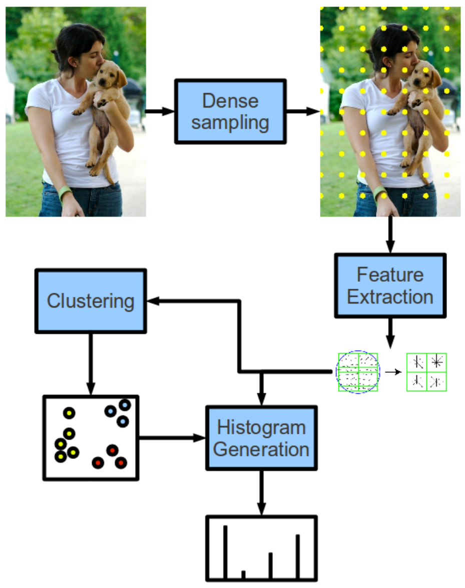

Fig. 1. OpponentSIFT-based Feature Extraction

On each of the channels of the opponent color space, SIFT features are extracted on a dense grid (we use a step size of 6 pixels) at a fix image scale over the image. SIFT features are extracted by first computing a gradient image on each of the color channels. Then, for each feature point, a region of 64×64 pixels around it is considered for its feature vector. This 64×64 pixel block is subdivided into 4×4 cells, each containing 16×16 pixels. For each cell, a histogram of its gradient directions, aligned to the main gradient direction, is computed. These 16 histograms are concatenated to build the feature vector for one point.

After obtaining one 384-dimensional (3 color channels × 16 cells × 8 orientation histogram bins) feature vector for each point, those vectors are quantized using K-Means clustering of a random subset of 800.000 features from all 8000 training images (i.e. 100 random features per image) with 4000 cluster centers (visual words). Then each feature point on the dense grid is assigned to a visual word by using a nearest neighbor classifier. Now that each feature point is described by its visual word, a histogram over these visual words in an image is computed, resulting in a 4000-dimensional histogram for each image. Similar to [2], in addition to this single histogram for the whole image, histograms for parts of the image are created. Therefore, two spatial partitions are defined, one (1×3) consisting of 3 vertically stacked regions and one (2×2) consisting of the four image quadrants. This yields a total of eight histograms: One for the whole image, three for 1×3 partition and four for the 2×2 partition. Each individual histogram is then L_1 normalized, i.e. divided by the sum of the bin population. To equally distribute the influence of different spatial partitions, the histograms of the 1×3 partition are weighted by $\frac{1}{3}$ and the histograms of the 2×2 partition are weighted by $\frac{1}{4}$. Finally, the histograms for a spatial partition are concatenated, resulting in histograms of 4000, 12000 and 16000 bins respectively. These histograms are then used in the SVM for classification.

2.2 Sharpness Features

In addition to the OpponentSIFT based features, a sharpness measure is used as an example for category-specific features. Therefore, the no-reference objective image sharpness measure by Ferzli and Karam described in [1] is used. This metric is based on user studies on blur perception in the human visual system (HVS). It incorporates the fact that the perception of blur in an image part is dependent on the edge statistics in this image part. Thus, a Sobel filter is used to generate an edge image. The edge image is then divided into 64×64 pixel blocks and all blocks are classified as edge or smooth blocks. A block is defined as an edge block where more than 0.2% of the pixels are edge pixels. Only edge blocks are considered when calculating the sharpness measure. The sharpness measure $D_{R_b}$for an edge block R_b is calculated by

$$D_{R_b} = \left(\sum_{e_i \in R_b} \left| \frac{w(e_i)}{w_{JNB}(e_i)} \right|^{\beta} \right)^{\frac{1}{\beta}} \tag{2}$$

where β is a constant fixed to 3.6, $w(e_i)$ is the edge width around pixel e_i in vertical or horizontal direction respectively and $w_{JNB}(e_i)$ is the just noticeable blur width corresponding to the contrast of the block R_b. The sharpness measure for the complete image is then calculated by

$$S = \frac{L}{D} \tag{3}$$

where

$$D = \left(\sum_{R_b} |D_{R_b}|^{\beta} \right)^{\frac{1}{\beta}} \tag{4}$$

and L being the total number of blocks in the image. This resulting sharpness metric $S \in \mathbb{R}^+$,with higher values indicating sharper images, has been used as a scalar input to the classifiers.

3 Category Learning and Classification

Kernel-based Support Vector Machines (SVM) have been widely used in visual codebook-based image classification scenarios (see e.g. [9],[8], [6]).

We use a two-class setting for binary classification, i.e. classifying images depicting a specific concept or not.

As in our case, training and testing samples are histograms of codeword distributions, we use the χ^2 distance, which has shown to provide good results for comparing distributions [9]. Given two histograms $H = (h_1, ..., h_m)$ and $H^{'} = (h^{'}_1, ..., h^{'}_m)$ the χ^2 distance is defined as:

$$D(H, H^{'}) = \frac{1}{2} \sum_{i=1}^{m} \frac{(h_i - h^{'}_i)^2}{|h_i| + |h^{'}_i|} \tag{5}$$

To incorporate this metric into a Support Vector Machine we use a Gaussian kernel:

$$K(H, H^{'}) = exp(-\frac{1}{\gamma} D(H, H^{'})) \tag{6}$$

The normalization parameter γ can be optimized using grid search and cross-validation. However, Zhang et al. [9] have shown, that setting this value to the average distance between all training image histograms gives comparable results and reduces the computational effort. The only parameter we optimize in a cross-validation is the cost parameter C of the support vector classification. We precompute the kernel matrix to to speed up subsequent training of the SVM. Additional speed is gained by computing the matrix in parallel on an 8-Core SMP-System. As described in section 2, we yield three different histograms per image − one per spatial pyramid resolution − which results in three different kernels.

In addition to the codeword histograms, we extracted a scalar quality measure (see sec. 2.2). The distance between to images is based on the absolute value of the difference value of both scalars. We use a linear kernel to compare for the quality measure as this has shown to provide better results than a Gaussian Kernel in our experiments.

Finally, we combine these four kernels by averaging their output into a single kernel, which is then used to train the Support Vector Machine. Here we use the implementation provided by the SHOGUN machine learning toolbox [7]. As the SHOGUN LibSVM implementation does not provide a probabilistic output we simply compute the sigmoid function in order to obtain values in [0..1].

4 Category Post-processing

In its current form, each category is treated independently from all other categories, assuming that dependencies between categories do not exist. This assumption actually does not hold in reality. There are even strong dependencies between different categories, i.e. categories excluding each other (such as *day* and *night*) or categories inferring other categories (e.g. *plant* can be inferred from *flower*). These dependencies have been accounted for by post-processing the classification decisions and confidences from the SVMs. Therefore, dependencies have been extracted from the training set. For exclusion of categories, partitions of categories have been identified, i.e. sets of categories P that do not co-occur and exactly one category of this set appears in the training annotations for each image:

$$\bigcup_{p \in P} C_p = I \tag{7}$$

$$\bigcap_{p \in P} C_p = \emptyset \tag{8}$$

In these definitions, I is the set of all images, P is the set of category indices belonging to a partition and C_p is the set of images annotated with category p. For category inference, pairs of categories are determined such that the statement "If an image is annotated with category a, it is also annotated with category b" holds. Formally, for these category pairs holds:

$$C_a \supseteq C_b \tag{9}$$

After determining the sets of excluding categories and the inference rules from the training set, these rules are used to post-process the output of the SVMs. The new confidence value $c'(i,p)$ for a category p is set to 0 if a higher confidence value for a different category within partition P exists. Only the largest confidence value for a category in a partition is kept.

$$c'(i,p) = \begin{cases} c(i,p) & \text{if } c(i,p) > c(i,q) \quad \forall q \in P \setminus p \\ 0 & \text{else} \end{cases} \tag{10}$$

If there are no other categories in an exclusion set with higher confidence values, the confidence value is kept. The confidence value of an image for a category q that is on the right side of an inference rule $p \Rightarrow q$ is updated if the confidence value of the left-hand side category p is greater than 0.5 and greater than the confidence value for category q:

$$c'(i,q) = \begin{cases} c(i,p) & \text{if } c(i,p) > c(i,q) \land c(i,p) > 0.5 \\ c(i,q) & \text{else} \end{cases} \tag{11}$$

This update rule can only increase the confidence value for a category, not decrease it.

5 Results and Summary

We submitted 5 different runs. All runs use the OpponentSIFT histograms as baseline. The first run (*OpSIFT*) uses the OpponentSIFT features alone. Another run (*OpSIFT+Excl+Inf+Qual*) uses the quality measure as an additional feature and applies category inference and exclusion as post-processing step. A third run (*OpSIFT+Qual*) does no post-processing but uses the quality feature. The fourth run (*OpSIFT+Inf+Qual*) uses the quality feature and applies only the category inference rule as we have noticed that exclusion often decreases the overall classification performance. A final fivth run (*OpSIFT+Inf*) uses the OpponentSIFT features and applies the inference rule on the classification results.

Three different evaluation measures have been computed. For evaluating the classification performance per concept the Mean Average Precision (MAP) was used. The evaluation per example was performed using the example-based F-Measure (F-Ex). Additionally an ontology score based on Flickr Context Similarity was computed (OS-fcs, for a detailed description see [4]). Table 1 shows the average scores achieved for each measure. For all evaluation measures, the runs with extensions to the baseline OpponentSIFT method gave the best results.

Run-Configuration	MAP	Avg. F-Ex	OS-fcs
OpSIFT	0.3492	0.6283	0.6318
OpSIFT+Excl+Inf+Qual	0.3331	0.6341	0.6401
OpSIFT+Qual	0.3495	0.6275	0.6362
OpSIFT+Inf+Qual	0.3495	0.6278	0.6364
OpSIFT+Inf	0.3493	0.6285	0.6319

Table 1. Average evaluation scores for all submitted runs. Highlighted values show the run, which obtained the best score for a specific evaluation measure.

Table 2 shows that using the additional category-specific features yields a gain of 0.00567 in terms of mean average precision for those categories where the sharpness feature has been used (i.e. *motion blur*, *out of focus*, *partially blurred* and *no blur*). The reasons for this little gain are, in our opinion, two-fold: On one side, the weight of the sharpness feature has not been optimized yet, resulting in a potential over- or under-estimation of the importance of the category-specific feature. On the other side, the sharpness measure disregards the orientation of edges, making it harder for the classifiers to distinguish between different kinds of blur as to be expected in the categories *motion blur* and *out of focus*.

Table 3 shows the detailed average precision for each category individually. As not all extensions to the baseline method are applied to all categories, the results for those categories do not differ between runs. For those categories where the results differ, the best performing run is highlighted in the table. In terms of MAP per category, the exclusion of categories often performed worse than the other runs. Especially for categories where the average precision is already low, the exclusion method even diminishes the results. We found that is due

Category	OpSIFT	OpSIFT+Q
Motion Blur	0.2173	0.2096
Out of Focus	0.1715	0.1689
Partly Blurred	0.7009	0.7229
No Blur	0.8913	0.9002
Mean Average Precision	**0.4953**	**0.5005**

Table 2. Average Precision for categories where the sharpness measure has been used. Best scores are highlighted.

to a lack of reliability of the confidence output of the SVMs. The SVM classifier of bad performing categories have a very small output range, e.g. values ranging from 0.94 to 0.96. In those cases, the number of support vectors used for these categories is usually near the total number of training samples. Using these categories for inference or exclusion of other categories yields a significant performance decrease, propagating the error introduced by one categories' classifier to other categories. In future, a check for the reliability of a classifier should be added to avoid this error propagation. This holds also for the category inference post-processing method, where this can happen as well. Error propagation also led to the decision that only inference rules with a confidence of 1.0 were used, missing opportunities were category inference could be used as well but evaluations have shown that in those cases, performance can drop significantly.

Category	**OpSIFT**	**OpSIFT+ E+I+Q**	**OpSIFT+ Q**	**OpSIFT+ I+Q**	**OpSIFT+ I**
Partylife	0.241183	0.241183	0.241183	0.241183	0.241183
Family_Friends	0.476125	0.476125	0.476125	0.476125	0.476125
Beach_Holidays	0.369467	0.369467	0.369467	0.369467	0.369467
Building_Sights	0.518642	0.518642	0.518642	0.518642	0.518642
Snow	0.126314	0.126314	0.126314	0.126314	0.126314
Citylife	0.502739	0.502739	0.502739	0.502739	0.502739
Landscape_Nature	0.769700	0.769700	0.769700	0.769700	0.769700
Sports	0.140956	0.140956	0.140956	0.140956	0.140956
Desert	0.039890	0.039890	0.039890	0.039890	0.039890
Spring	0.119698	0.106550	0.119698	0.119698	0.119698
Summer	0.226241	0.226859	0.226241	0.226241	0.226241
Autumn	0.293647	0.018332	0.293647	0.293647	0.293647
Winter	0.181106	0.020955	0.181106	0.181106	0.181106
No_Visual_Season	0.954757	0.947659	0.954757	0.954757	0.954757
Indoor	0.573047	0.517216	0.573047	0.573047	0.573047
Outdoor	0.875141	0.868957	0.875141	0.875141	0.875141
No_Visual_Place	0.578996	0.563815	0.578996	0.578996	0.578996
Plants	0.695391	0.697349	0.695391	0.697349	0.697349
Flowers	0.383228	0.383228	0.383228	0.383228	0.383228
Trees	0.601623	0.601623	0.601623	0.601623	0.601623
Sky	0.863894	0.865421	0.863894	0.865421	0.865421
Clouds	0.802869	0.802869	0.802869	0.802869	0.802869
Water	0.600010	0.600010	0.600010	0.600010	0.600010
Lake	0.265445	0.265445	0.265445	0.265445	0.265445
River	0.210948	0.210948	0.210948	0.210948	0.210948
Sea	0.486808	0.486808	0.486808	0.486808	0.486808
Mountains	0.493615	0.493615	0.493615	0.493615	0.493615
Day	0.837934	0.820454	0.837934	0.837934	0.837934
Night	0.519675	0.410528	0.519675	0.519675	0.519675
No_Visual_Time	0.757272	0.756426	0.757272	0.757272	0.757272
Sunny	0.426355	0.426355	0.426355	0.426355	0.426355

Sunset_Sunrise	0.722446	0.722446	0.722446	0.722446	0.722446
Still_Life	0.313663	0.313663	0.313663	0.313663	0.313663
Macro	0.463493	0.463493	0.463493	0.463493	0.463493
Portrait	0.606200	0.606200	0.606200	0.606200	0.606200
Overexposed	0.152760	0.025393	0.152760	0.152760	0.152760
Underexposed	0.223520	0.223520	0.223520	0.223520	0.223520
Neutral_Illumination	0.976283	0.971761	0.976283	0.976283	0.976283
Motion_Blur	0.217280	0.034776	0.209853	0.209853	0.217280
Out_of_focus	0.171542	0.016420	0.168916	0.168916	0.171542
Partly_Blurred	0.700893	0.722854	0.722857	0.722857	0.700893
No_Blur	0.891267	0.893303	0.900226	0.900226	0.891267
Single_Person	0.504431	0.485730	0.504431	0.504431	0.504431
Small_Group	0.274982	0.233470	0.274982	0.274982	0.274982
Big_Group	0.362310	0.035081	0.362310	0.362310	0.362310
No_Persons	0.888193	0.879804	0.888193	0.888193	0.888193
Animals	0.394545	0.394545	0.394545	0.394545	0.394545
Food	0.455053	0.455053	0.455053	0.455053	0.455053
Vehicle	0.438675	0.438675	0.438675	0.438675	0.438675
Aesthetic_Impression	0.258722	0.258722	0.258722	0.258722	0.258722
Overall_Quality	0.216578	0.216578	0.216578	0.216578	0.216578
Fancy	0.164452	0.164452	0.164452	0.164452	0.164452
Architecture	0.265967	0.265967	0.265967	0.265967	0.265967
Street	0.332411	0.332411	0.332411	0.332411	0.332411
Church	0.167250	0.167250	0.167250	0.167250	0.167250
Bridge	0.059372	0.059372	0.059372	0.059372	0.059372
Park_Garden	0.381190	0.381190	0.381190	0.381190	0.381190
Rain	0.005603	0.005603	0.005603	0.005603	0.005603
Toy	0.218664	0.218664	0.218664	0.218664	0.218664
MusicalInstrument	0.039265	0.039265	0.039265	0.039265	0.039265
Shadow	0.092194	0.092194	0.092194	0.092194	0.092194
bodypart	0.224537	0.224537	0.224537	0.224537	0.224537
Travel	0.118880	0.118880	0.118880	0.118880	0.118880
Work	0.037389	0.037389	0.037389	0.037389	0.037389
Birthday	0.009230	0.009230	0.009230	0.009230	0.009230
Visual_Arts	0.333149	0.333149	0.333149	0.333149	0.333149
Graffiti	0.062694	0.062694	0.062694	0.062694	0.062694
Painting	0.189944	0.189944	0.189944	0.189944	0.189944
artificial	0.143548	0.143548	0.143548	0.143548	0.143548
natural	0.708399	0.708399	0.708399	0.708399	0.708399
technical	0.062041	0.062041	0.062041	0.062041	0.062041
abstract	0.019354	0.019354	0.019354	0.019354	0.019354
boring	0.074641	0.074641	0.074641	0.074641	0.074641
cute	0.591775	0.591775	0.591775	0.591775	0.591775
dog	0.263080	0.263080	0.263080	0.263080	0.263080
cat	0.069199	0.069199	0.069199	0.069199	0.069199
bird	0.184953	0.184953	0.184953	0.184953	0.184953
horse	0.117973	0.117973	0.117973	0.117973	0.117973
fish	0.020769	0.020769	0.020769	0.020769	0.020769
insect	0.104334	0.104334	0.104334	0.104334	0.104334
car	0.321748	0.321748	0.321748	0.321748	0.321748
bicycle	0.166077	0.166077	0.166077	0.166077	0.166077
ship	0.161543	0.161543	0.161543	0.161543	0.161543
train	0.170937	0.170937	0.170937	0.170937	0.170937
airplane	0.135721	0.135721	0.135721	0.135721	0.135721
skateboard	0.005289	0.005289	0.005289	0.005289	0.005289
female	0.528553	0.528553	0.528553	0.528553	0.528553
male	0.701139	0.700843	0.701139	0.700843	0.700843
Baby	0.156602	0.156602	0.156602	0.156602	0.156602
Child	0.120649	0.120649	0.120649	0.120649	0.120649
Teenager	0.230674	0.230674	0.230674	0.230674	0.230674
Adult	0.494042	0.494042	0.494042	0.494042	0.494042
old_person	0.057177	0.057177	0.057177	0.057177	0.057177
Mean Average Precision	**0.349225**	**0.333119**	**0.349450**	**0.349484**	**0.349260**

Table 3: Average Precision per category. Scores are only highlighted, when any of the extensions provided a performance increase or decrease.

In terms of the exemplar-based ontology score, the run containing all extensions performed best. We think that is due to the fact that the post-processing methods introduce a consistency between labels of an image which is actually evaluated by the ontology score.

Acknowledgements This work was supported in part by Federal Ministry of Economics and Technology of Germany under the project THESEUS (01MQ07018).

References

1. R. Ferzli and L.J. Karam. A No-Reference Objective Image Sharpness Metric Based on the Notion of Just Noticeable Blur (JNB). *Image Processing, IEEE Transactions on*, 18(4):717 –728, April 2009.
2. S. Lazebnik, C. Schmid, and J. Ponce. Beyond Bags of Features: Spatial Pyramid Matching for Recognizing Natural Scene Categories. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition - Volume 2 (CVPR'06)*, pages 2169–2178. IEEE, 2006.
3. D. G. Lowe. Distinctive Image Features from Scale-Invariant Keypoints. *International Journal of Computer Vision*, 60(2):91–110, November 2004.
4. S. Nowak and M. Huiskes. New Strategies for Image Annotation: Overview of the Photo Annotation Task at ImageCLEF 2010. In *Working Notes of CLEF 2010*, Padova, Italy, 2010.
5. J. Sivic and A. Zisserman. Video google: a text retrieval approach to object matching in videos. In *Computer Vision, 2003. Proceedings. Ninth IEEE International Conference on*, pages 1470–1477, April 2003.
6. C. G. M. Snoek and M. Worring. Concept-Based Video Retrieval. *Foundations and Trends in Information Retrieval*, 2(4):215–322, 2009.
7. S. Sonnenburg, G. Raetsch, S. Henschel, C. Widmer, J. Behr, A. Zien, F. de Bona, A. Binder, C. Gehl, and V. Franc. The SHOGUN Machine Learning Toolbox. *Journal of Machine Learning Research*, 11:1799–1802, Jun 2010.
8. K.E.a. van de Sande, T. Gevers, and C. G.M. Snoek. A comparison of color features for visual concept classification. *Proceedings of the 2008 international conference on Content-based image and video retrieval - CIVR '08*, page 141, 2008.
9. J. Zhang, M. Marszałek, S. Lazebnik, and C. Schmid. Local Features and Kernels for Classification of Texture and Object Categories: A Comprehensive Study. *International Journal of Computer Vision*, 73(2):213–238, September 2006.