Logistic Regression of Generic Codebooks for Semantic Image Retrieval

João Magalhães and Stefan Rüger

Department of Computing, South Kensington Campus
Imperial College London, London SW7 2AZ, UK
{j.magalhaes, s.rueger}@imperial.ac.uk

Abstract. This paper is about automatically annotating images with keywords in order to be able to retrieve images with text searches. Our approach is to model keywords such as 'mountain' and 'city' in terms of visual features that were extracted from images. In contrast to other algorithms, each specific keyword-model considers not only its own training data but also the whole training set by utilizing correlations of visual features to refine its own model. Initially, the algorithm clusters all visual features extracted from the full imageset, captures its salient structure (e.g. mixture of clusters or patterns) and represents this as a generic codebook. Then keywords that were associated with images in the training set are encoded as a linear combination of patterns from the generic codebook. We evaluate the validity of our approach in an image retrieval scenario with two distinct large datasets of real-world photos and corresponding manual annotations.

1 Introduction

The growing interest in managing multimedia collections effectively and efficiently has created new research interest that arises as a combination of multimedia understanding, information extraction, information retrieval and digital libraries. In this paper we focus on information extraction algorithms that model keywords such as '*sky'* and '*beach*' in terms of visual features that were extracted from images. These models return the probability of a keyword being present in new unseen images and thus enabling text searches on collections of images.

Most of the existing approaches just model the visual features of images containing a given keyword ignoring the presence of other keywords in the same image and their cross-interference. Some keywords such as '*bird*' or '*plane*' have a visual representation too complex to be captured by just its training data: the presence of different concepts and their cross-interference increases the uncertainty of the extracted information. For example, concepts such as '*sun*', '*outdoor*' or '*indoor*' may be easy to detect but concepts such as '*bird'*, '*boat'* or '*insect'* may be more reliably detected if other, more basic and correlated concepts were detected previously.

Thus, we advocate that one should first detect the most salient low-level visual patterns of the full image dataset in the feature space and then learn the causality relation between these low-level visual pattern co-occurrences and the keywords. To achieve

H. Sundaram et al. (Eds.): CIVR 2006, LNCS 4071, pp. 41 – 50, 2006.

this goal, we formulate the following hypothesis: "*Given a common generic codebook of patterns (codewords) of the full imageset, the keywords can be encoded as a low-complexity linear combination of codewords and exhibit a competitive retrieval performance*". With this hypothesis we aim to achieve a fast, simple, scalable algorithm capable of annotating images with keywords at high precision. The codebook lists the low-level visual patterns of the full image dataset and its contents will be interchangeably referred to as patterns, clusters or codewords.

Section 2 describes related work on image-semantic annotation, Section 3 the proposed algorithm, Section 4 presents the experiments and results, and Section 5 discusses the generic codebook generation and the algorithm characteristics.

2 Related Work

Several algorithms have been proposed to extract semantic information from multimedia content. Single-class-model approaches estimate an individual distribution function for each keyword. Other types of approaches are based on a translation model between keywords and images features (global, tiles or regions). These two groups of approaches assume a minimal relation between the various elements of the same image (words, blobs, tiles). Hierarchical models consider the hierarchical relation or the inter-dependence relation between the elements of an image (words and blobs or tiles) and reflect it in the statistical model.

Single-class-models are a straight-forward approach to the semantic analysis of multimedia content. The idea behind is to learn a class-conditional probability distribution of each single keyword *w* of the semantic vocabulary given its training data *x*. Bayes law is used to invert the problem and model $p(x \mid w)$ the features data density distribution of a given keyword. Several techniques to model the $p(x \mid w)$ with a simple density distribution have been proposed: Yavlinsky et al. [1] deployed a non-parametric distribution; Carneiro and Vasconcelos [2] a semi-parametric density estimation; Westerveld and de Vries [3] a finite-mixture of Gaussians; while Mori et al. [4], and Vailaya et al. [5] apply different flavours of vector quantization techniques. This type of approach only considers the class's own data ignoring the co-occurrence of classes, while the present approach takes that into consideration.

Other types of approaches are based on a translation model between keywords and images (global, tiles or regions). Inspired by machine translation research, Duygulu et al. [6] developed a method of annotating image regions with words. First, regions are created using a segmentation algorithm like normalized cuts. For each region, features are computed and then blobs are generated by clustering the image features for these regions across an image collection. The problem is then formulated as learning the correspondence between the discrete vocabulary of blobs and the image keywords. Following a translation model Jeon et al. [7], Lavrenko et al. [8] and Feng et al. [9] studied a model where blob features $b_I^{(r)}$ of an image *I* are conditionally independent of keywords w_i. Jeon et al. [7] recast the image annotation as a cross-lingual information retrieval problem applying a cross-media relevance model based on a discrete codebook of regions. Lavrenko et al. [8] continued previous work by Jeon et al. [7]

and described the process of generating blob features with continuous probability density functions $P(b_I^{(r)} \mid J)$ to avoid the loss of information related to the generation of the codebook. Prolonging their previous work Feng et al. [9] replace blobs with tiles and model image keywords with a Bernoulli distribution. These methods have the mathematical form of kernel density estimation – the model corresponds to the entire training data – making them computationally very demanding: in contrast our model uses a linear combination of a common codebook for all classes, which is by its very nature computationally much simpler.

In the hierarchical models group Barnard and Forsyth [10] studied a generative hierarchical aspect model, inspired by a hierarchical clustering/aspect model. The data are assumed to be generated by a fixed hierarchy of nodes with the leaves of the hierarchy corresponding to soft clusters. Blei and Jordan [11] describe three hierarchical mixture models to annotate image data, culminating in the correspondence latent Dirichlet allocation model. This model specifies a Bayesian model for capturing the relations between regions, words and latent variables. It combines the advantages of probabilistic clustering for dimensionality reduction with an explicit model of the conditional distribution from which image keywords are generated. In our approach we do not create a fixed hierarchy of nodes/clusters that decreases the flexibility of the method.

3 Algorithm Description

The dataset is composed of a training set and a test set of images, and each image is manually annotated with a vocabulary of keywords corresponding to the visual content of that particular image. The image dataset is initially processed to extract a set of low-level visual features from all images. Once the low-level visual features and the manual annotations are loaded the features are further processed.

The generic codebook is produced with an unsupervised learning algorithm that returns a finite mixture of clusters modelling the full dataset feature space. The set of clusters of the finite mixture is then stored in the generic codebook as *K* codewords, each of which is defined by a set of parameters θ_k. The notation for the probability of a codeword *k* for an image *i*, $q\left(x^{(i)} \mid \theta_k\right)$, will be abbreviated as $q_k\left(x^{(i)}\right)$.

Only in the final step of the algorithm are the annotations used to learn the keyword-model of each keyword w_t, i.e., the weight $\beta_k^{w_t}$ of each codeword θ_k. The model $p(w_t \mid x)$ expresses the probability of a word w_t given the low-level visual features x of an unseen image. This model is defined as a generalized linear model:

$$g\left(E\left[w_t \mid x^{(i)}\right]\right) = \beta_0^{w_t} + \beta_1^{w_t} q_1\left(x_1^{(i)}\right) + \ldots + \beta_K^{w_t} q_K\left(x_K^{(i)}\right), \tag{1}$$

where the link function $g(\cdot)$ allows to model non-linear relations between and the features $x^{(i)} = \left[1, x_1^{(i)}, \ldots, x_K^{(i)}\right]$, or a transformation of it e.g. $q_n(x_n)$, and the $E[w_t \mid X]$. Typical link functions are the identity function for normal linear regression, the *logit* function for logistic regression and the *log* function for log-linear regression. In this paper we consider the logistic regression model.

3.1 Features Processing

The feature processing step normalises the features and creates smaller-dimensional subspaces from the original feature-spaces. Three different low-level features are used in our implementation: **marginal HSV distribution moments**, a 12 dimensional colour **feature** that captures the histogram of 4 central moments of each colour component distribution; **Gabor texture**, a 16 dimensional texture feature that captures the frequency response (mean and variance) of a bank of filters at different scales and orientations; and **Tamura texture**, a 3 dimensional texture feature composed by measures of image's coarseness, contrast and directionality. We used *N*=15 feature sub-spaces. As a common practice we tiled the images in 3 by 3 parts before extracting the low-level features. This has two advantages: it adds some locality information and it greatly increases the amount of data used to learn the generic codebook.

3.2 Learning the Generic Codebook

In the algorithm's second step, the features subspace clustering is done under the assumption that the subspaces are independent. That is, each feature subspace *n* is processed individually and modelled as a Gaussian mixture model (GMM)

$$p(x \mid \theta_n) = \sum_{m=1}^{M_n} \alpha_{n,m} p\left(x \mid \mu_{n,m}, \sigma_{n,m}^2\right), \tag{2}$$

where M_n is the number of Gaussians (clusters) in feature subspace *n*, x is the low-level visual feature, and θ_n represents the complete set of model parameters with mean $\mu_{n,m}$, covariance $\sigma_{n,m}^2$, and component prior $\alpha_{n,m}$. The components priors have the convexity constraint $\alpha_{n,1},\ldots,\alpha_{n,m} \geq 0$ and $\sum_{m=1}^{M_n} \alpha_{n,m} = 1$. We implemented the mixture learning algorithm as proposed by Figueiredo et al. in [12] in C++. Each codeword $q(\cdot \mid \theta_k)$ correspond to a certain cluster $p\left(x \mid \theta_{n,m}\right)$.

The algorithm starts with a number of clusters that is much larger than the real number and gradually eliminates the clusters that start to get few support data (singularities). This avoids the initialization problem of EM since the algorithm only produce mixtures with clusters that have enough support data. This strategy can cause a problem when the initial number of clusters is too large: no cluster receives enough initial support causing the deletion of all clusters. To avoid this situation, cluster parameters are updated sequentially and not simultaneously as in standard EM. That is: first update cluster 1 parameters $\left(\mu_1, \sigma_1^2\right)$, then recompute all posteriors, update cluster 2 parameters $\left(\mu_2, \sigma_2^2\right)$, recompute all posteriors, and so on.

After finding a good fit for a GMM with *k* clusters, the algorithm deletes the weakest cluster and restarts itself with *k-1* Gaussians and repeats the process until a minimum number of clusters is reached. Each fitted GMM is stored and in the end the set of fitted models describe the feature subspace at different levels of granularities. We can then consider generic codebooks with different levels of complexities.

3.3 Learning Keyword-Models: A Logistic Model

As mentioned before, we cast the keyword-models as a generalized linear model. The codebook of clusters modelled the features sub-spaces with great detail so that it can be used now as smoothing functions on the logistic model. The link function $g(x) = \text{logit}(p) = \log\left(p/(1-p)\right)$ defines the log-posterior odds between positive examples and negative examples of a keyword as a linear combination of the codebook output:

$$\text{logit}(E[w_t \mid x]) = \beta_0^{w_t} + \beta_1^{w_t} q_1(x_1) + ... + \beta_K^{w_t} q_K(x_K). \tag{3}$$

Assuming a matrix notation we define the codebook and the parameters as $Q(x) = [1, q_1(x_1), ..., q_K(x_K)]$ and $\beta^{w_t} = [\beta_0^{w_t}, \beta_1^{w_t}, ..., \beta_K^{w_t}]$, respectively. This allows writing the logistic model as:

$$p(w_t \mid x) = \frac{1}{1 + e^{\beta^{w_t} Q(x)}}. \tag{4}$$

We implemented the binomial logistic regression model where one class is always modelled relatively to all other classes. With this choice we achieve some independence between keywords because they only depend on their own β^{w_t}: once the codebook is computed, it is the same for all classes and only the β^{w_t} weights are specific of each class. A second reason for choosing the binomial approach is due to the complexity of the algorithms for fitting multinomial models and their requirements. We tested several methods to compute the β^{w_t} weights, and discuss two of the methods.

3.3.1 Parameter Estimation Using L-BFGS

The only variables we now need to compute using the annotations are the priors β^{w_t} which can be computed by minimizing the log-likelihood of the above model over the entire data set:

$$\beta^{w_t} = \arg\max_{\beta^{w_t}} \sum\nolimits_{i \in I} l(\beta^{w_t}), \tag{5}$$

where $l(\beta)$ is the log-likelihood function, and *I* is the entire training set. We used a Gaussian prior with σ^2 variance to prevent the optimization procedure from overfitting. Thus the log-likelihood function for a binomial logistic model becomes:

$$l(\beta) = \sum\nolimits_{i \in I} \left\{ y_{w_t}^{(i)} \beta^T Q\left(x^{(i)}\right) - \log\left(1 + e^{\beta^T Q\left(x^{(i)}\right)}\right) \right\} - \frac{\beta^T \beta}{2\sigma^2}, \tag{6}$$

where $y_{w_t}^{(i)}$ is 1 if the image *i* has the keyword w_t and 0 otherwise, $x^{(i)}$ is the low-level visual features of the image *i*. To maximize the log-likelihood of each keyword model, we set its gradient to zero and proceed with a Quasi-Newton optimization algorithm:

$$\frac{\partial l(\beta)}{\partial \beta} = \sum\nolimits_{i \in I} Q\left(x^{(i)}\right)\left(y_{w_t}^{(i)} - p\left(w_t \mid x^{(i)}, \beta\right)\right) - \frac{\beta}{\sigma^2} = 0. \tag{7}$$

Because the dataset is quite large and the codebook might hold up to 10000 codewords, algorithms that depend on the computation of the Hessian would require too much memory. It has been shown that for this type of models the limited-memory BFGS algorithm [13] is the best solution. We use the implementation provided by Liu and Nocedal [13].

3.3.2 Parameter Estimation Using Codewords Log-Odds

A simpler and computationally less demanding algorithm is based on the codewords's log-odds. Since we have a fixed codebook with predefined parameters θ_k we estimate the $\beta_k^{w_j}$ weights individually as the corresponding codeword log-odds. The prior of each codeword is then given by the logarithm of the proportion of positive versus negative examples:

$$\beta_k^{w_t} = \log \frac{E[q_k \mid w = w_t]}{1 - E[q_k \mid w = w_t]} = \log \frac{E[q_k \mid w = w_t]}{E[q_k \mid w \neq w_t]}. \tag{8}$$

The expected values of the above expressions are:

$$E[q_k \mid w] = \frac{1}{|I_w|} \sum\nolimits_{i \in I_w} q_k\left(x_k^{(i)}\right), \tag{9}$$

where the set $i \in I_w$ represents the set of images annotated with the keyword w, and $x_k^{(i)}$ is the low-level visual feature k of the image i. The interpretation of Equation (8) is straightforward: if the codeword is more relevant for negative examples the proportion will be < 1 and thus its log will be negative, if the codeword is more relevant for positive examples the proportion will be > 1 and thus its log will be also positive. This way, when evaluating unseen samples each codeword will have a low or high contribution to the overall model probability.

4 Experiments and Results

The algorithm was tested with a subset of the COREL Stock Photo CDs [6] and a subset of Getty Images [1] in a typical information retrieval scenario to evaluate its mean average precision. We conducted an evaluation of our model with a view to study the influence of the granularity of the generic codebook.

COREL Dataset. This dataset was compiled by Duygulu et al. [6] from a set of COREL Stock Photo CDs. The dataset has some visually similar concepts (jet, plane, Boeing), and some concepts have a limited number examples (10 or less). In their seminal paper, the authors acknowledge that fact and ignored the classes with these problems. In this paper we use the same setup as in [1], [2], [7], [8] and [9], which differs slightly from the one used in the dataset original paper, [6]. The retrieval evaluation scenario consists of a training set of 4500 images and a test set of 500 images. Each image is annotated with 1-5 keywords from a vocabulary of 371 keywords. Only keywords with at least 2 images in the test set were evaluated which reduced the number of vocabulary to 179 keywords. Retrieval lists have the same length as the test set, i.e. 500 items.

Fig. 1 depicts the evolution of the mean average precision with the complexity of the generic codebook. On the test set the maximum achieved MAP was 27.7% with an average codebook complexity of 45 clusters per feature per tile – note that these clusters are common to all keywords. The different codebook sizes reflect the different levels of model granularities of the features-subspaces. The different granularities are based on model complexity (number of parameters) of each feature subspace (other criteria could have been used such as likelihood or MDL).

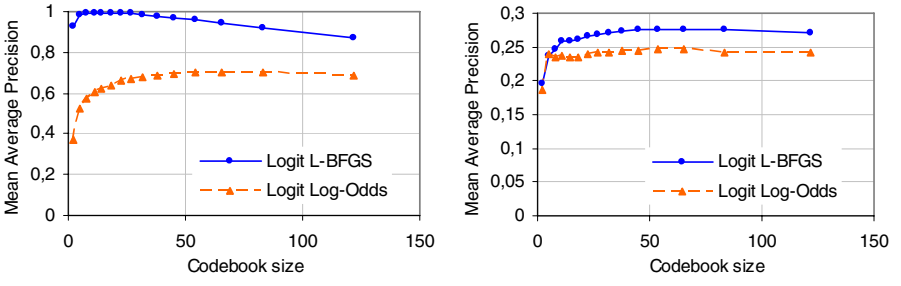

Fig. 1. Evolution of the MAP vs codewords per feature per tile for the Corel collection on the training (left) and on the test set (right)

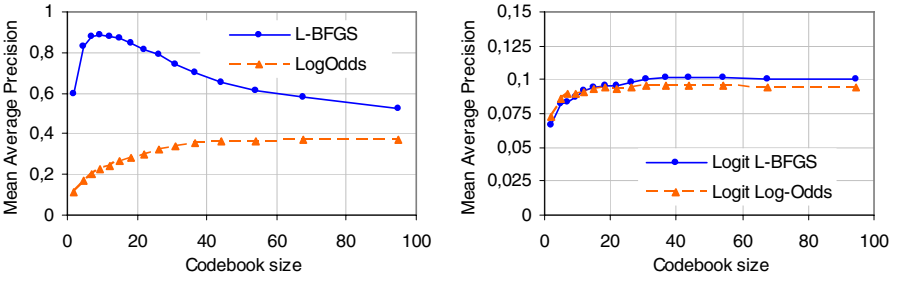

Fig. 2. Evolution of the MAP vs codewords per feature per tile for the Getty-Images collection on the training (left) and on the test set (right)

Getty Images Dataset. This dataset compiled by Yavlinsky et al. [1] is a selection of photographs retrieved by submitting queries with a given selection that result in a random selection of photos, which excludes any non-photographic content, any digitally composed or enhanced photos and any photos taken in unrealistic studio settings. The resulting dataset contains pictures from a number of different photo vendors, which reduces the chance of unrealistic correlations between keywords and image contents. Keywords for Getty images can express subjects (e.g. 'tiger'), concepts (e.g. *'emptiness'*) or styles (e.g. *'panoramic photograph'*).

The retrieval evaluation scenario consisted in a training set of 5000 images and a test set of 2560 images. Only keywords with at least 2 images in the test set were evaluated which results in a vocabulary of 184 keywords. Retrieval lists have the same length as the test set, 2560 items. We use the same setup as in [1].

Fig. 2 depicts the evolution of the mean average precision with the different complexities of the generic codebook. On the test set, the maximum achieved MAP was 10.2% with an average codebook complexity of 36 clusters per feature per tile – note that these clusters are common to all keywords.

5 Discussion

The creation of the codebook is inevitably a generalization procedure, which translates into a trade-off between accuracy and simplicity. Thus, the described algorithm offers an appealing solution for applications that require an information extraction algorithm with a good precision that, at the same time, is simple, economical and robust.

Table 1. MAP measures of the different algorithms

Algorithm	Corel	Getty
Cross-Media Relevance Model [7]	16.9%	-
Continuous-space Relevance Model [8]	23.5%	-
Logistic regression (Log-Odds)	**24.6%**	**9.6%**
Logistic regression (L-BFGS)	**27.7%**	**10.2%**
Nonparametric Density Distribution [1]	28.6%	9.2%
Multiple-Bernoulli Relevance Model [9]	30.0%	-
Mixture of Hierarchies [2]	31.0%	-

Good retrieval precision. The retrieval performance of our approach is competitive: Table 1 compares our algorithm retrieval performance against others. Note that our method uses a simple set of features, a basic tiling method and requires less computational resources than any other method (both in terms of CPU and memory), and it still delivers a competitive retrieval performance. The differences in mean average precision between these two datasets show that Getty dataset is much more difficult than Corel dataset.

Inference scalability. Since the generic codebook is common to all keywords the clusters must be computed only once for all keywords. Thus, the resources required to evaluate the relevancy of an image for each keywords are relatively modest. Apart from the mixture of hierarchies [2] all other methods are some sort of nonparametric density distributions. It is well known [14] that the nonparametric nature of these methods makes the task of running these models on new data computationally demanding: the model corresponds to the entire training set meaning that the demand on CPU time and memory increases with the training data. To infer all the keywords with our best model requires only 36 to 45 clusters per feature-subspace per tile, while method [9] requires 1 Gaussian kernel per tile (24 tiles) per training image (4000). For example, to evaluate all 179 keywords of the Corel dataset our model needs to compute 36×15×9=4,860 Gaussians plus a linear combination for each keyword, while method [9] needs to compute 4000×24=96,000 Gaussian kernels plus a linear combination for each keyword.

Keywords scalability. Assuming that the used set of keywords is a faithful sample of a larger keyword vocabulary it is expected that one can use the same codebook to learn the logistic model of new random keywords and preserve the same retrieval performance. Note that the codebook is a representation of the data feature space: it is selected based on the set keywords.

Little overfitting. The MAP curve on the test set remains quite stable as the common model complexity increases depicting the algorithm's immunity to overfitting. Our model can be interpreted as ensemble methods (additive models) if we consider that each cluster is a weak learner and the final model a linear combination of those weak learners. This means that our model has some of the characteristics of additive models namely the observed immunity to overfitting. It is interesting to note that the simple log-odds estimation of the parameters appears more immune to overfitting than the l-bfgs algorithm. This fact occurs because the optimization procedure fits the model tightly to the training data (favouring large $\beta_k^{w_t}$), while the log-odds estimation avoids overfitting by computing weighted average the expected value of all codewords. Note that when fitting the model we are minimizing a measure of the average classification residual error (model log-likelihood) and not a measure of how documents are ranked in a list (Mean Average Precision). The mean average precision is the mean of the accumulated precision over a ranked list. This contributes to the large difference between the training set MAP and the test set MAP. To the best of our knowledge there are no published results assessing the training set MAP versus the test set MAP at different model complexities and therefore we cannot compare our results with others.

6 Conclusions

This paper's novelty resides in the simplicity of the linear combination of a generic visual vocabulary for image retrieval and the keyword's parameters estimation process: the results show that such a low complexity approach compares competitively with much more complex approaches. This has a bearing on the design of image search engines, where scalability and response time is as much of a factor as the actual mean average precision of the returned results. It is also important to stress the little-overfitting exhibited by the algorithm.

Our aim was to explore the most salient low-level visual patterns of the full dataset feature space and learn the causality relation between these patterns' co-occurrence and the keywords. To achieve this goal, we formulated a hypothesis: "*Given a common generic codebook of patterns (codewords) of the full imageset, the keywords can be encoded as a low-complexity linear combination of codewords and exhibit a competitive retrieval performance*". The evaluation results allow us to conclude that the initial hypothesis is valid.

Acknowledgements. We thank Alexei Yavlinsky for having provided us with the Getty Images dataset low-level visual features and its annotations. This work was partially funded by the Portuguese Foundation for Science and Technology.

References

[1] A. Yavlinsky, E. Schofield, and S. Rüger, "Automated image annotation using global features and robust nonparametric density estimation," Int'l Conference on Image and Video Retrieval, Singapore, 2005.

[2] G. Carneiro and N. Vasconcelos, "Formulating semantic image annotation as a supervised learning problem," IEEE Conference on Computer Vision and Pattern Recognition, San Diego, CA, USA, 2005.

[3] T. Westerveld and A. P. de Vries, "Experimental result analysis for a generative probabilistic image retrieval model," ACM SIGIR Conference on research and development in information retrieval, Toronto, Canada, 2003.

[4] Y. Mori, H. Takahashi, and R. Oka, "Image-to-word transformation based on dividing and vector quantizing images with words," Int'l Workshop on Multimedia Intelligent Storage and Retrieval Management, Orlando, FL, USA, 1999.

[5] A. Vailaya, M. Figueiredo, A. K. Jain, and H. J. Zhang, "Image classification for content-based indexing," *IEEE Transactions on Image Processing*, vol. 10, pp. 117-130, 2001.

[6] P. Duygulu, K. Barnard, N. de Freitas, and D. Forsyth, "Object recognition as machine translation: Learning a lexicon for a fixed image vocabulary," European Conference on Computer Vision, Copenhagen, Denmark, 2002.

[7] J. Jeon, V. Lavrenko, and R. Manmatha, "Automatic image annotation and retrieval using cross-media relevance models," ACM SIGIR Conference on research and development in information retrieval, Toronto, Canada, 2003.

[8] V. Lavrenko, R. Manmatha, and J. Jeon, "A model for learning the semantics of pictures," Neural Information Processing System Conference, Vancouver, Canada, 2003.

[9] S. L. Feng, V. Lavrenko, and R. Manmatha, "Multiple Bernoulli relevance models for image and video annotation," IEEE Conference on Computer Vision and Pattern Recognition, Cambridge, UK, 2004.

[10] K. Barnard and D. A. Forsyth, "Learning the semantics of words and pictures," Int'l Conference on Computer Vision, Vancouver, Canada, 2001.

[11] D. Blei and M. Jordan, "Modeling annotated data," ACM SIGIR Conference on research and development in information retrieval, Toronto, Canada, 2003.

[12] M. Figueiredo and A. K. Jain, "Unsupervised learning of finite mixture models," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, pp. 381-396, 2002.

[13] D. C. Liu and J. Nocedal, "On the limited memory method for large scale optimization," *Mathematical Programming B*, vol. 45, pp. 503-528, 1989.

[14] T. Hastie, R. Tibshirani, and J. Friedman, *The elements of statistical learning: Data mining, inference and prediction*: Springer, 2001.