Visual Object Categorization using Topic Models

Andreas Damianou

Master of Science
School of Informatics
University of Edinburgh
2009

Abstract

This project investigates the applicability of image annotation algorithms based on Latent Dirichlet Allocation (LDA) to a task different from the one for which they were originally designed: visual object categorization.

We developed a topic model which is based on an existing approach and evaluated its performance on four datasets which we created from the PASCAL database using different feature extraction algorithms. We treat class labels as a set of caption words for the images and compare the performance of the LDA-based model with that of an SVM classifier.

The research strategy followed is a combination of a case study and a series of experiments whose results are analyzed in detail. We demonstrate how this approach can be used in order to take effective decisions in every step of the overall modeling process. We also identify a set of critical issues, such as parameter estimation and dataset properties, which affect the performance of the LDA-based model.

Our results show that the LDA-based model is outperformed by the traditionally used SVM, but it achieves comparable scores despite the fact that it is designed for a different purpose. We identify strengths of the topic-model approach, such as efficient data representation using the topic space, robustness to small training sets and ability to control smoothing, which could be combined with a more domain specific algorithm and potentially produce better results.

Acknowledgements

First of all I would like to thank my supervisor Frank Keller. I am extremely grateful to him for his invaluable assistance and guidance.

My deepest gratitude is to my parents and brother, for their love and for being supportive in everything I do. This work is dedicated to them.

Finally, I feel that I owe much to the friends I met here, in Edinburgh, for making this year truly unforgettable.

Declaration

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

(*Andreas Damianou*)

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

Automatic image annotation (or simply, image annotation) and *visual object categorization* (or simply, object categorization) are two very similar kinds of problems. Given an unobserved image, the first task is about deciding which words of a learned vocabulary would best describe it (thus, create a caption for the image), whereas the second task is about identifying the existence of certain objects (belonging in a predefined set) depicted in it. The words "describe" and "identify" of the above informal definitions show the different directions at which the two corresponding kinds of algorithms would aim. Automatic image annotation algorithms try to capture a more general concept of the depicted image. For example, the word "war" could be learned from a picture with soldiers and a battlefield.

Topic models constitute a very common choice for the basis of an image annotation algorithm. They are capable of representing the coocurrences of terms (for example "war") with images in an intermediate *topic space*, something which is useful as it captures general information about these coocurrences and reveals the underlying "aspects" of pictures. On the other hand, *image classification* is a popular method for solving object categorization problems, as it has to do with implicitly learning how to discriminate between classes given a set of image representations (usually as real-valued vectors) with their accompanying class-labels.

The connection of the image annotation and object categorization tasks becomes even more obvious by considering that the first approaches for image annotation were closely related to image classification ([Smeulders et al., 2000] , [Vailaya et al., 2001a]). And reversely, there are many efforts which modify specific topic-model

based techniques originally developed for pure image annotation tasks, for image classification ([Bosch et al., 2006], [Fei-Fei and Perona, 2005], [Wang et al., 2009], [Blei and Mcauliffe, 2007]).

However, although these approaches may share common grounds, rarely the implementation designed to solve specifically the one kind of problems (image annotation) has been tested on the other (object categorization). The efforts made so far towards this direction modify the models to such an extend that they can no longer be regarded as image annotation methods, but they are simply based on topic models. We claim that the two types of problems can be almost identical under certain circumstances (a intuition which was confirmed by very recent research [Wang et al., 2009]) and we believe that it would be interesting to attempt solving object categorization tasks by utilizing algorithms traditionally used for image annotation. That is, to attempt solving the object categorization problem by applying a slightly modified model which remains, though, image annotation oriented. Such an effort would result in defining formally the conditions (which involve dataset properties and objectives pursued) under which the two kinds of tasks can be thought as being similar. It is reasonable to believe that in cases like this, a topic model which performs better than another one in an object categorization task would have an advantage when applied to the image annotation task. Moreover, as described in section 2.3.2, methods from different research fields (information retrieval, machine translation, machine learning, natural language processing, object detection) inspired the development of algorithms for image annotation. So, it is reasonable to think that the opposite can also work: methods that were created, evolved and tested mainly for the image annotation problem, could also inspire new approaches for a similar problem, namely the object categorization one.

For all the above reasons, in this project we investigate the applicability of a particular image annotation method based on a topic model called *Latent Dirichlet Allocation (LDA)* for object categorization tasks. In order to do that, we treat the caption-words, that the model normally emits in order to tag images, as class-labels for our dataset. Its performance is compared with an SVM classifier which does not detect the position of the object in the image but, instead, learns from the whole picture and assumes that all classes are independent.

1.2 Aims

The **primary objective** is to describe the way in which topic models originally developed for image annotation can be applied to object categorization problems, to evaluate their performance and identify their weaknesses for this kind of tasks. In particular, various tests will -hopefully- reveal the issues that must be taken into consideration when applying this kind of methods to object categorization tasks (for example optimal tuning of the parameters, dataset properties etc).

The **secondary objective** is to demonstrate in detail all the stages involved in the procedure which must be followed in order to apply our algorithm on a challenging, multi-class dataset: from feature extraction to application of evaluation measures and analysis of the results. We will try to show the way in which a series of experiments along with a good research review can help taking effective decisions for each stage of the procedure and for the specific task of object categorization. For this reason, the whole procedure is, hopefully, described in such a way that the project can also be seen as a case study and a reader with knowledge of the domain could replicate the whole process.

While trying to achieve the aims mentioned, we hope that some insight will be provided for the following issues: firstly, based on this work, one would be able to conclude to what extent the one method can be combined with the other (also investigated in [Wang et al., 2009]) and would draw numerous other relevant conclusions about the similarity of the tasks; secondly, one can notice that in a sense, in order to solve our vision task we borrow an image annotation model which, in turn, borrows a language model (LDA). Therefore, this work, as a whole, will also provide a small demonstration of how algorithms originally developed for a specific domain can be extended in order to solve different kinds of problems.

1.3 Project Outline

The structure of the project is the following:

- **Chapter 2** presents a detailed overview of the literature that is related to this work. The task of object categorization (treated as image classification) and the automatic image annotation one are defined and explained, as well as the concept of topic modeling and the relevant approaches for automatic image annotation.

- **Chapter 3** describes the image database used in our experiments and focuses on the process of extracting a good feature set, which is used by our models for object categorization.
- **Chapter 4** refers to the Support Vector Machine model used for comparison with our LDA-based method. In particular, we analyze how model selection is done and how the multi-class problem is approached by using this binary classifier.
- **Chapter 5** describes our LDA-based model, analyzing its design and implementation details as well as the approach used for parameter estimation.
- **Chapter 6** presents and analyzes the results obtained from various experiments conducted in order to compare the two models and identify their strengths and weaknesses to the object categorization task.
- **Chapter 7** summarizes the findings and conclusions of this work. Potential directions for future work are also discussed.

Note on the implementation: The software programmed for the purpose of this project is split into numerous segments. In that way, users can easily select the appropriate part of code according to the subtask that they wish to accomplish. As the subtasks that are involved in the whole process described in the next chapters are very heterogeneous, for each one the most appropriate programming language is chosen. As a result, the whole package includes code in Java, Perl/Python, Matlab, standalone executables and shell scripts.

Chapter 2

Background

2.1 Object categorization as image classification

2.1.1 Defining the task

Visual object categorization refers to the task of learning to identify objects (which belong to predefined groups, such as "person", "car" etc) depicted in images, as in figure 2.1.

Figure 2.1: *Pictures grouped according to the object type that they depict. In contrast to a segmentation task, we don't care about drawing a bounding box around the object. source: [Everingham et al., 2008]*

In contrast to visual object recognition, which concerns the identification of particular object *instances*, categorization concerns the identification of object *types*, taken from a pre-defined set. For instance, as [Csurka et al., 2004] explains, recognition would distinguish between images of two cars with different structure, while catego-

rization would place them in the same class under the label “car” (although the definitions are not absolute, for example object recognition is very often interpreted as object categorization). Moreover, in contrast to a visual segmentation problem, where we additionally care about the position of the identified objects (i.e, the task is to draw a bounding box around the pixels that form the visual object), in object categorization we are only interested in the task of answering with “yes” or “no” to the question “Does a given image contain an object X?”, where X is expressed in words. This is in contrast with *content-based image retrieval*, where a query image is used and the system must return pictures similar to it, according to some citerion.

Image classification is a simple and natural method for successfully solving object categorization problems. More sophisticated models might use classification with boosting or other techniques. We decided to use a simple image classification model as a baseline for comparison with the performance of our LDA-based model. In the simple case, it is given a number of individual images that depict objects and place them into predefined groups (classes), so as each one contains images that depict the same kind of objects, assuming that each class is independent from the others. Intuitively, this assumption is naive, because if the algorithm is able to identify with high confidence that, for example, a bird is present in a picture then this should bias the prediction that a tree is also present. However, on the one hand the combinations of such correlations are vast and difficult to capture, and on the other hand this approach would require some prior knowledge on the specific dataset where the algorithm is applied. For this reason, and because we wish to compare our topic model with a simple baseline rather than a sophisticated technique, our method for image classification will simply involve the training and tuning of a **Support Vector Machine (SVM)**, which is, in general, a powerful classification model. However, in the dataset that we will use, a picture can belong to more than one classes at the same time (for example it might depict a motorbike as well as a person). This is called *multi-label classification* and is, in general, a difficult task. The method what we used in order to deal with this issue is described in detail in chapter 4.

2.1.2 SVM classifier

Support Vector Machines (SVMs), based on *statistical learning theory* principles [Vapnik, 1999] are widely used for challenging classification tasks because of their attractive properties that make them so effective. Firstly, they use the *kernel-trick* (which

was first introduced by [Aizerman et al., 1964]) which allows traditional algorithms to be extended in interesting ways, as explained later in this section. Therefore, SVM's can be thought as being an extension of the simple linear models such as *logistic regression*, which can only represent linear boundaries and are, thus, inappropriate for most of the practical problems. It is interesting that although the non-linear kernel function is used to transform the original input space into a new one with more dimensions, no computations are involved in that high-dimensional space, and thus, classical algorithms can be applied without adaptation [Baudat and Anouar, 2001]. With this non-linear mapping, the decision boundary learned is linear in the new space (*feature space*) but not in the original, *data space*, thus being more discriminant. Secondly, SVM's produce *sparse solutions* and therefore the predictions for unseen input depend on the evaluation of the kernel function only on a small number of training instances (called *support-vectors*), a property which minimizes the execution time. Moreover, as figure 2.2 shows, the support-vectors are used to find the separation hyperplane that maximises the *margin* between the dataset (known as the *optimal hyperplane*), leading in a classifier with low *generalization error*, as proven in [Anguita et al., 2000]. Finally, the model parameters are found by solving a *convex optimization problem*, so local solutions are also global optima [Bishop, 2006].

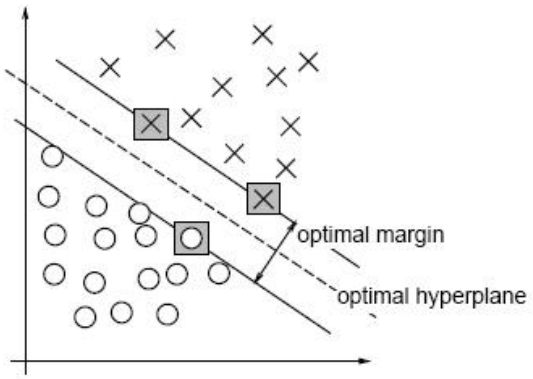


Figure 2.2: *A binary separable problem: the maximum margin is defined by the support vectors (grey squares) and constitutes the largest separation between the two datasets. source: [Cortes and Vapnik, 1995]*

An SVM algorithm requires significant tuning (parameter setting). In this project we based our parameter settings on some experiments and relevant papers. In order to understand the way in which the parameter selection is done, some deeper knowledge of the theory behind SVM's is necessary, so as to understand the role and significance of each variable. To explain briefly how SVMs work, let us suppose that we have a

training set $\mathbf{x} \in R^{n \times d}$, i.e there are a total of n training instances with dimensionality d and that the corresponding targets are given by a vector $\mathbf{y} \in R^n$, with $y_i = \pm 1$, so we have a **binary classification problem**. The SVM algorithm tries to find a **hyperplane** (the "optimal" one) that separates the two classes and is described by an equation of the form:

$$y(\mathbf{x}) = \mathbf{w}^T \phi(\mathbf{x}) + w_0 \tag{2.1}$$

The *weight vector* $\mathbf{w}$ and the *bias parameter* w_0 are the unknowns. ϕ is the *transformation function* of the input space where $\mathbf{x}$ originally lies. In order to find the unknowns we must solve an optimization problem. However, as explained in [Luenberger, 1984], instead of working with the original problem, we can solve its dual. It can be shown ([Bishop, 2006]) that the solution of the dual problem can be found by optimizing the following function:

$$\begin{aligned} \min_{\mathbf{w}, w_0, \xi} \quad & \frac{1}{2}\mathbf{w^T w} + C\sum_{i=1}^{n} \xi_i \\ subject\ to: \quad & y_i\left(\mathbf{w^T}\phi(\mathbf{x_i}) + \mathbf{w_0}\right) \geq 1 - \xi_i \\ & \xi_i \geq 0,\ i = 1...n \end{aligned} \tag{2.2}$$

This is a *quadratic programming problem*. The parameters ξ in this equation are called *slack variables*, and are used in order to perform a trade-off between classification error against minimizing the original optimization function, when the problem is a *non-separable* one. The parameter C is the penalty or *cost variable* which defines the scale of this trade-off. The SVM algorithm using the optimization function of (2.2) is often called *C-SVM* in order to be distinguished from the *v-SVM* [Scholkopf et al., 2000], which contains an additional parameter v which allows to control the number of support vectors and errors. The solution to the above optimization problem (2.2) is proven to be:

$$\hat{\mathbf{w}} = \sum_i \alpha_i y_i \phi(\mathbf{x}_i) \tag{2.3}$$

where the α_i's are defined during training by solving a quadratic programming problem (different than the one described in (2.2)), which is described in [Karatzoglou et al., 2006] and [Boser et al., 1992] in detail.

Given an unseen input $\mathbf{x}$, the predictions of the model are made by computing the quantity:

$$sgn(f(x))$$
$$where \qquad f(x) = \hat{\mathbf{w}}\phi(\mathbf{x}) + w_0 \tag{2.4}$$

The term $\phi(\mathbf{x})\phi(\mathbf{x}_j)$ that arises in the summation when substituting $\hat{\mathbf{w}}$ from (2.3), can be replaced with an equivalent *kernel function* $k(\mathbf{x}, \mathbf{x}_j)$, applying in that way the *kernel-trick* explained in the beginning of this section. Since we can choose any transformation function ϕ that we want, we can choose any kernel function that we want as well. In fact, we can even use some of the already known kernel functions even if we don't know to which transformation function they correspond. Different kernels produce a different feature space and give rise to different algorithms. The four basic **kernel functions** are the following: (described in [Hsu et al., 2003])

- linear: $k(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i^T \mathbf{x}_j$
- polynomial: $k(\mathbf{x}_i, \mathbf{x}_j) = (\gamma \mathbf{x}_i^T \mathbf{x}_j + r)^d, \gamma > 0$
- radial basis function (RBF): $k(\mathbf{x}_i, \mathbf{x}_j) = exp(-\gamma \|\mathbf{x}_i - \mathbf{x}_j\|^2), \gamma > 0$
- sigmoid: $k(\mathbf{x}_i, \mathbf{x}_j) = tanh(\gamma \mathbf{x}_i^T \mathbf{x}_j + r)$

Summarizing all the above, when deciding to use an SVM algorithm, there are numerous issues that must be resolved:

- The choice of the kernel function
- The choice of the gamma parameter for the kernel function (if it is not chosen to be a linear one)
- The choice of the c (cost) parameter of equation (2.2)
- The choice of the specific algorithm which is used in order to solve the quadratic problems and apply heuristics for efficiency
- The way which the original multi-class problem is *binarized*, i.e transformed to multiple binary problems (because all the equations presented above require that the target variable $y_i \in \pm 1$).

The choices that were made for the above issues in this project, are discussed and justified in chapter 4.

More information about SVMs can be obtained from [Cortes and Vapnik, 1995] and [Burges, 1998], with the second being a very interesting tutorial.

2.1.3 Multi-class classification

The discussion in section 2.1.2 was made under the assumption that there are just two classes, +1 and -1. However, in our task, as well as in most practical applications, we are required to distinguish between more than two classes. This problem is called *multi-class classification* and the corresponding dataset *multi-class dataset*. Moreover, each instance of the multi-class dataset might belong to more than one class at the same time (for example, in our dataset it may depict a person and a dog as well). This problem is called *multi-label* problem. In order to utilize a binary classifier to solve this kind of problems, we can follow two types of approaches, discussed below.

The first, "naive" approach suggests modifying the dataset in a way that a binary classifier can still be applied. [Tsoumakas and Katakis, 2007] describe and compare the ways in which this can be achieved. We present only the methods that are suitable for our task:

- Subjectively or randomly select one of the multiple labels of each multi-label instance and discard the rest. This is proven to be (and obviously is) a choice that would produce bad or even unacceptable results.
- For each instance x with label set Y, decompose each pair (x,Y) into $|Y|$ new pairs (x,l) for every l in Y. For example an instance:

  $$class1\ class4\ <features>$$

  would be transformed into two instances (with the same feature-set):

  $$class1\ <features>$$
  $$class4\ <features>$$

  One of the problems of this approach is that it would require the tesing set to be pre-processed in the same way, something that is not always feasible.

The second approach, is to build a classifier which can produce **probability outputs** for each candidate class, i.e a vector $p_1, p_2, ..., p_n$, where n is the number of different classes, such that p_i is the probability that the particular instance belongs to $class_i$. Of course, $\sum_{i=1}^{n} p_i = 1$. This approach is far more preferable, not only because

it results in better classification and has better mathematical grounds, but also because many applications and evaluation techniques do require this kind of output.

In order to take advantage of the second method described above, we need to proceed in two steps: In the first step, the problem must be decomposed into binary tasks, for example with one of the methods described in [Gidudu et al., 2007]. In the second step, probability estimates from all the binary tasks must be found and represented as a single probability for each class. [Platt and Platt, 1999] describes and compares the ways of doing that. The choice of a particular method for the issues just described, is a matter of implementation and, thus, is described in chapter 4.

2.2 Topic models

Topic models are generative models which can learn in an unsupervised way the underlying mixture of (unobserved) topics that compose a text document or any kind of observation that can be represented as a *bag of words* (a model described in the next paragraph). Although they were first used for linguistic purposes, as text documents can naturally be represented as a bag-of-words, since the first probabilistic version [Hofmann, 1999] they have been widely used in various contexts. Currently, *Latent Dirichlet Allocation (LDA)* [Blei et al., 2003] is the state of the art and has served as a basis for numerous other extensions. In the context of automatic image annotation, extensions of the LDA model can be designed in order to correlate in a supervised way the correct caption-words with the topics that are found in an unsupervised manner by the basic LDA algorithm, given the visual features extracted from the image. The most important of these extensions are discussed in section 2.4.

2.2.1 The Bag-of-Words model and the dyadic domain

LDA assumes that the data to which it is applied are *dyadic* and represented as a *bag-of-words*. A **bag-of-words (BoW)** is a representation where an object is represented as an unordered collection of terms, with every term being independent from all the others. For example, a document may be represented as a collection of words, without providing any information about any syntactical, grammatical or structural properties that these words might define within the document. As discussed in [Lewis, 1998], this *assumption* is basically the one made by the Naive Bayes classifier when applied to text data. Obviously this is a naive assumption, but is proven to be a good choice

for the same reasons that Naive Bayes is found to be so effective: [McCallum and Nigam, 1998] explains this paradox and proves that the independence assumption, when applied to a large number of attributes for classification purposes, simplifies the learning, as the parameters for each attribute can be learnt separately. In other words, complicated models are very likely to "confuse" the learning algorithm. In principle, the BoW representation has many variations, discussed in [Lewis, 1998]. One of these, is to represent each document as a vector $w_1, w_2, ..., w_V$, where V is the size of our fixed vocabulary. w_i can be either 1, if $word_i$ is present in the document, or 0. Alternatively, w_i can be the number of times that $word_i$ is present in the document, or even the normalized number of times. As will be discussed in the next chapters, image data can also be represented as a bag of **visual words**.

According to [Hofmann, 1998], "**dyadic data** (or two-mode data) refers to a domain with two finite sets of objects in which observations are made for dyads, i.e pairs with one element from either set". For example, if a document is represented as a bag-of-words, then the observations would be made in pairs of the form (d, w), meaning that document d contains word w. In other words, the concern about dyadic data is to model their *co-occurences*.

2.2.2 Notation and definitions

LDA can be applied to any kind of data which can be expressed over a *dyadic domain*, thus being utilized in many fields, like text-based information retrieval, computer vision etc. However, since the most intuitive type of dyadic data are documents represented as bag-of-words in a corpus, we will use notation and definitions from this domain. It is obvious, though, how the notation and definitions described below could be adapted to better describe the dyadic data of other domains (for example the term *word* could be replaced with *visual word* if the domain is computer vision). We use the following definitions and notation:

- A *word* (or *term*) is the basic item of the whole structure, taken from a predefined vocabulary of size V.
- A *document* d is a collection of M words, represented as a vector $w_1, w_2, ..., w_M$.
- A *topic* is a latent variable which represents a distribution over words. Given that this distribution may define a larger probability for specific words, as well as that there are sets of words that co-occur more often in documents, in linguistic terms

a topic would be what we intuitively understand as the *aspect* of a document. However, topics are latent variables and for our purposes we don't care about their intuitive meaning.

- A *dataset* or *corpus* is a collection of D documents.

2.2.3 The generative process

The generative process that LDA follows, attempts to connect the particles of dyadic data (for example documents and words) *only through the latent topics*. So, a document is represented as a mixture of topics and each topic is represented as a distribution over words, by following the procedure described below:

Generative Process

Parameters	
α,β	given hyperparameters
M	# of words
K	# of topics
D	# of documents

Dimension of vectors and matrices	
z	$D \times M$
w	$D \times M$
θ	$D \times K$
ϕ	$K \times M$

Algorithm

For each topic k:

$\phi_k \sim Dir(\beta)$

For each document d:

1) $\theta \sim Dir(\alpha)$

2) For each of the M words:

a) Sample topic $z_m \sim Mult(\theta)$

b) Sample word w_m from $p(w_m|z_m, \beta)$:

$w_m \sim Mult(\phi_{z_m})$

Note that the multinomial distribution is chosen because the model basically relies on the computation of discrete counts of co-occurrences of words in documents, and that the Dirichlet distribution is chosen because it is conjugate to the multinomial distribution, something which makes the computations for inference and parameter estimation easier.

The above process results in the following joint distribution:

$$p(\theta, \mathbf{z}, \mathbf{w}|\alpha, \beta) = p(\theta|\alpha)\prod_{i=1}^{M} p(z_i|\theta)p(w_i|z_i, \beta) \tag{2.5}$$

which is also presented graphically in figure 2.3. The posterior distribution of the latent variables, which must be computed for inference given a new unseen document, is derived from equation (2.5), but the calculation is intractable and, thus, approximation methods must be used [Blei et al., 2003].

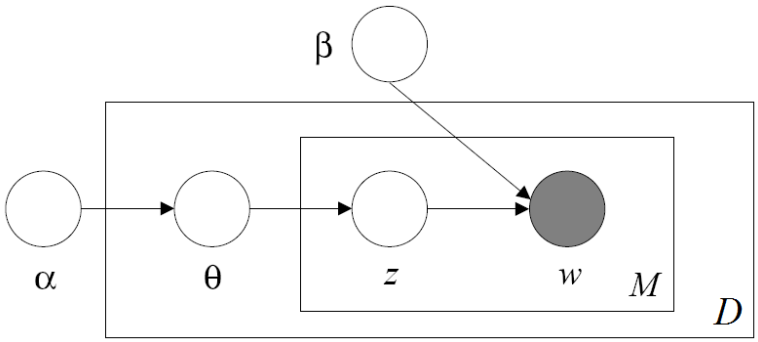


Figure 2.3: *The plate notation for the LDA generative process. The outer plate is repeated D times, for each document and the inner M times, for each word. The observed variable w is shaded. source: [Blei et al., 2003]*

If the generative process is reversed and the algorithm is applied to an existing corpus, its main outputs would be the ones described in table 2.1:

The distribution of documents over topics	$p(topic\|document)$
The distribution of topics over words	$p(word\|topic)$

Table 2.1: *The distributions obtained be applying the LDA algorithm for estimation.*

2.3 The Image Annotation task

2.3.1 Defining the task

As discussed briefly in 1.1, *automatic image annotation* (or *image tagging*) refers to the task of automatically creating captions for previously unobserved images by assigning words to them, taken from a pre-defined vocabulary. This is primarily used for the efficient management of large image databases. Images which contain captions can

be searched using keywords, while non-annotated images can only be searched with *content-based* techniques ([Das et al., 1998], [Vailaya et al., 2001b]) which are less efficient and easy to use. Automatic image annotation has generated a lot of recent interest, as the whole process is related to many domains, such as computer vision (for image representation as features), information retrieval, natural language processing and more. The following description of different approaches used so far is indicative of the way in which standard approaches in one field inspire new methods for a different domain.

2.3.2 Overview of the approaches used for image annotation

Different approaches have been used to solve this difficult task. Two of the first proposed were not very different than simple image classification ([Vailaya et al., 2001a], [Smeulders et al., 2000]). According to these, images are assigned a label which describes them and for each label a binary classifier is trained. In contrast, [Mori et al., 1999] proposed a model known as *the co-occurence model*, because it bases its predictions on the frequency by which words were co-occuring with image regions.

Topic models also try to capture the co-occurence of words with image regions, but they introduce latent variables. We chose to utilize this kind of models in our project, so a more detailed description is necessary and will be given in section 2.4.

[Duygulu et al., 2002] developed the *Translation Model*, obviously inspired from Machine Translation, which operates in two steps: Firstly, each image is represented as set of *blobs* (image regions with their corresponding visual features) by applying a segmentation algorithm to it, extracting features from each segment and clustering the total set of features found. Then, standard machine translation algorithms which use EM are applied in order to learn a "translation" from the word set to the blob set.

Finally, *Relevance Models* ([Jeon et al., 2003a], [Jeon et al., 2003b]), inspired by Information Retrieval have also been proposed and are considered to be the state of the art. The concept of this idea is to perform the following two steps: firstly, find the most similar images to the test image; then, find the most frequent words of these images' captions and use them to form the test image's caption.

It is worth mentioning that the model developed by [Jeon et al., 2003b] as well as some topic models described in section 2.4 can model continuous features directly, as they occur in image data, in contrast to [Jeon et al., 2003a], translation model([Duygulu et al., 2002]) and co-occurence model([Mori et al., 1999]). In that way it

can take advantage of continuous features (such as normalized frequencies etc) without having to discretize them, and thus, loose precision.

2.4 Topic models for image annotation

2.4.1 General design and principles

As mentioned in the previous section, many image annotation algorithms are based on topic models, which were described in 2.2. The key idea behind this kind of extensions is that a collection of images can also be treated as a collection of dyadic data [Hofmann, 1998]. For example, [Csurka et al., 2004] describes how images can be represented as a *bag of visual keypoints* and treated as dyadic data.

A learning algorithm for image annotation tasks, would have to be applied to a database of image-caption pairs, i.e two sets of dyadic data (because a caption can be treated as a text document, an array of words). So, we extend the notation that was defined in 2.2.2 as follows:

We use the following **definitions and notation:**

- A *word* (or caption-word) is taken from the predefined vocabulary of length V_M, and is part of the caption that describes the document.
- A *blob* is an *image region* with its corresponding features.
- A *visual-word* (also called *visterm*) may be a single *visual feature* of the image or a *blob*, depending on the algorithm used. In any case, the whole set of visual-words is used to represent the image. In total V_N different visual-words exist in the vocabulary.
- A *term* is the most general category of words and visual-words. When the distinction between a "word" and a "visual word" is not a matter or when we want to refer to both at the same time, we will simply use the word "term".
- A *caption* is composed of a bag of M words (which form the document's caption), and is represented as a vector $w_1, w_2, ..., w_M$, while an *image* is composed of a bag of N visual-words, and is represented as a vector $v_1, v_2, ..., v_N$.
- A *dataset* or *corpus* is a collection of D documents. We use a training and a test dataset.

- A *document d* belonging in the training dataset is a collection of pairs $(caption, image)$, i.e $document_i = caption_i \cap image_i$. A document belonging in the test dataset consists only of the *image* part, as the *caption* part must be predicted by the algorithm. When the distinction between a *document*, a *caption* and an *image* is obvious (the context in which we refer to it is obvious), we will use for all the term *document*. For example, a document from the test set is basically an image.

Notice that in the above terminology, we use the terms "caption" and "word" although in our task these would correspond to "label-set" and "class" (or "label") respectively. However, this section describes topic models in general, and will borrow the terminology which is usually used, i.e the one taken from computational linguistics.

The key idea of the existing models which are based on extensions of LDA is to define a generative process which generates both, images and captions, providing some connection between them. This is achieved by performing the LDA learning step twice, once for the dyadic pair $(caption, word)$ and once for the pair $(image, visual\ word)$, where the caption is represented as a bag-of-words and the image is represented as a bag-of-visual-words. Thus, when the generative process of this extended model is reversed, the outputs of table 2.1 will now be augmented to the ones described by the table 2.2.

The distribution of captions over caption-topics u	$p(topic_u \mid caption)$
The distribution of topics u over words	$p(word \mid topic_u)$
The distribution of images over image-topics z	$p(topic_z \mid image)$
The distribution of topics z over vis.words	$p(vis.word \mid topic_z)$

Table 2.2: *The distributions obtained be applying an extended LDA algorithm for annotating an unseen image.*

The most important **differences among the existing LDA-based algorithms**, have basically to do with the following:

- The distributions that are used in order to sample **z**,**u**,**v**,**w**
- The way in which **z** and **u** are connected in a supervised way (there must be some connection for the algorithm to actually learn)
- The order in which images and captions are generated by the overall generative process. This has to do with the *exchangeability assumptions*, explained below.

A sequence of random variables is said to be **exchangeable** if any order of samples could be drawn with the same probability. Exchangeable random variables are to *Bayesian statistics* what i.i.d variables are to *frequentist statistics*. A brief and intuitive explanation is that according to *De Finetti's theorem* [Diaconis, 1977], an exchangeable sequence of random variables, not necessarily i.i.d, can be expressed as a mixture of underlying i.i.d sequences. Exchangeability is, thus, an implicit assumption that may be used by statistical models.

[Blei et al., 2003] describes this concept in the context of LDA: it is assumed that topics are infinitely exchangeable within a document. Given this, defining conditional distributions of topics over words is like assuming that words are generated by topics. By applying de Finetti's theorem the joint distribution of words and topics may be obtained. In the context of LDA models for image annotation, the exchangeability concerns caption-words and visual-words, as explained in [Barnard et al., 2003]. A model which assumes that caption-words and visual-words are exchangeable (for example [Barnard et al., 2003]) implies that these can be generated in any order. Thus, in order to predict caption words based on the image's visual features, a higher level mixture component should be used in order to impose some dependency and "tie" the topics learned for images with the topics learned for their corresponding captions. This can be avoided by assuming *partial exchangeability*, as in [Blei and Jordan, 2003]. In this case, images are generated first and then captions are generated from them, by learning the appropriate conditional distributions.

2.4.2 Existing models based on LDA

Most of the existing LDA-based models proposed for solving image annotation tasks, follow the general structure and principles described in 2.4.1. [Blei and Jordan, 2003] describes and compares three mixture models suitable for describing dyadic data, and thus being applicable to the image annotation problem. Their **Gaussian-multinomial mixture (GM-Mixture)** model is not a pure extension of LDA, but it is a latent factor model which follows similar principles. The method samples a latent topic z just once per image/caption pair, and then the image and the caption are both generated by sampling visual words (or regions) and caption-words respectively conditioned on this single z. On the other hand, the **Gaussian-multinomial LDA (GM-LDA)** model breaks this "extreme correspondance" of the topic-image and topic-caption distributions and improves the modeling of the joint probability of images and captions. It achieves this

by sampling a random variable θ from a Dirichlet distribution which represents a distribution over both kinds of latent variables and then generating both, image-regions and words with θ held fixed. In that way, the latent variables (topics) for each word or region are allowed to vary. However, the authors of [Blei and Jordan, 2003] argue that a good modeling of the joint distribution of words and images is not always indicative of the quality of word-predictions made given an unseen image. This is because such predictions require a good modeling of the *conditional probabilities* involved while developing the models. Indeed, in the GM-LDA model the dependency of the topics learned for images and the ones learned for captions is too loose. For this reason, the authors develop a robust, LDA-based model which combines the good correspondance of GM-Mixture with the flexibility and the good joint distribution modeling ability of GM-LDA. The resulting model, **Correspondence LDA (Corr-LDA)**, works in two phases: in the first phase, it generates the image following the generative process of the traditional LDA algorithm; in the second phase, caption-words are sampled by the following iterative procedure: one of the regions of the already generated image is chosen and a caption-word is generated conditioned on the latent factor that was used to generate the region during phase one.

[Barnard et al., 2003] proposes the **Mixture of Multi-Modal LDA (MoM-LDA)** which attempts to provide the same level of correspondance as Corr-LDA but in a different way. First of all, the authors implicitly assume that blobs and words are exchangeable, and so they develop a generative procedure where blobs and words can be generated in any order, as discussed in section 2.4.1. However, this assumption would imply no dependency between the latent factors (topics) used to generate the image and the caption. Indeed, these factors are sampled from the same distribution but without being connected in any other way, like in GM-LDA described in the previous paragraph. So, they pose a dependency by sampling words and blobs conditioned not only on this latent factor, but also on a **global** mixture component c.

It is worth mentioning that, of course, the existing LDA-based models are not limited to the ones described, but these are the most popular and "pure", as many other methods use various modifications inspired from different domains.

Chapter 3

Datasets

3.1 Description and challenges of the dataset

The dataset that is used for all of our experiments is taken from the **PASCAL Visual Object Classes Challenge 2008** competition [1] [Everingham et al., 2008]. It contains images which depict objects of 19 different visual classes [2] in realistic scenes and each class corresponds to one object type, such as those depicted in figure 3.1 (for example: person, sheep, dog, car etc) [3].

The dataset is considered to be very challenging, since objects are depicted in various different angles, scales and lighting conditions. Moreover, each image can contain none, one or more objects from the same or different classes and certain parts of an object may not be visible (for example they may be occluded by another object). Objects have been manually annotated with bounding boxes in each image, but for classification purposes the set of class labels accompanying each picture, forming thus a caption, is the only annotation that is really needed.

The PASCAL competition's organizers also propose a splitting of the dataset into training, validation and test set. Of course for the test set no annotations are provided. Therefore, for our purposes, we used the proposed training set and we split the proposed validation set into validation and test set. This resulted in a training set of 2093 images, a validation set of 350 images and a test set of 759 images. The distributions

[1] http://pascallin.ecs.soton.ac.uk/challenges/VOC/voc2008/index.html

[2] Actually the original dataset contains a set of 20 different labels, but for our purposes we ignored the class-label "train" and added a special class "no_object". The removal of the "train" label and a single file in which all annotations for all pictures are gathered, was taken from: http://www.inf.ed.ac.uk/teaching/courses/iaml/

[3] More example images are available at:
http://pascallin.ecs.soton.ac.uk/challenges/VOC/voc2008/examples/index.html

Figure 3.1: *A sample of the image set used. The annotations are placed as yellow bounding boxes, but for classification only the class-label is needed. source: [Everingham et al., 2008]*

of images and objects by class do not vary significantly across the training, validation and test set, as one can see in figure 3.2.

However, as also can be seen in figure 3.2, even in the training set there are a few images which contain no objects from the predefined classes. One option, motivated by the intuition that a classifier cannot actually learn anything from them, is simply to completely remove these instances, but if this was the option of choice the organizers would not have included these instances in the dataset in the first place. For this reason, we chose to add a special class **"no_object"** to indicate that images belonging there do not depict any of the 19 other object types. The addition of a label like this is a very popular choice for multi-class classification tasks and it is also convenient in terms of implementation as most of the existing classification software cannot receive as input instances without labels. We don't know if instances with no objects are included in the original PASCAL test set because its annotations are kept as a secret, but the organizers do not include a "no_object" class in their evaluation reports. In any case, since in our test set there are images without objects we chose to treat the "no_object" class as any other class, even in the evaluation. Thus, we have a total of **20 classes**.

For some datasets, object recognition and object categorization (for the distinction between the terms see section 2.1.1) are problems almost equally difficult to solve. An example of a dataset of this kind is one where all images of a particular type of object might look very similar (low intra-class variability). However, [Opelt et al., 2004] explains than when object categorization is applied to datasets with certain properties, the actual task is *generic* object categorization, which is even harder to solve. These properties include:

- Images with high **background clutter**. While the human cognitive recognition system is very effective in extracting relevant information from images that depict objects in a cluttered environment, artificial vision systems find it difficult

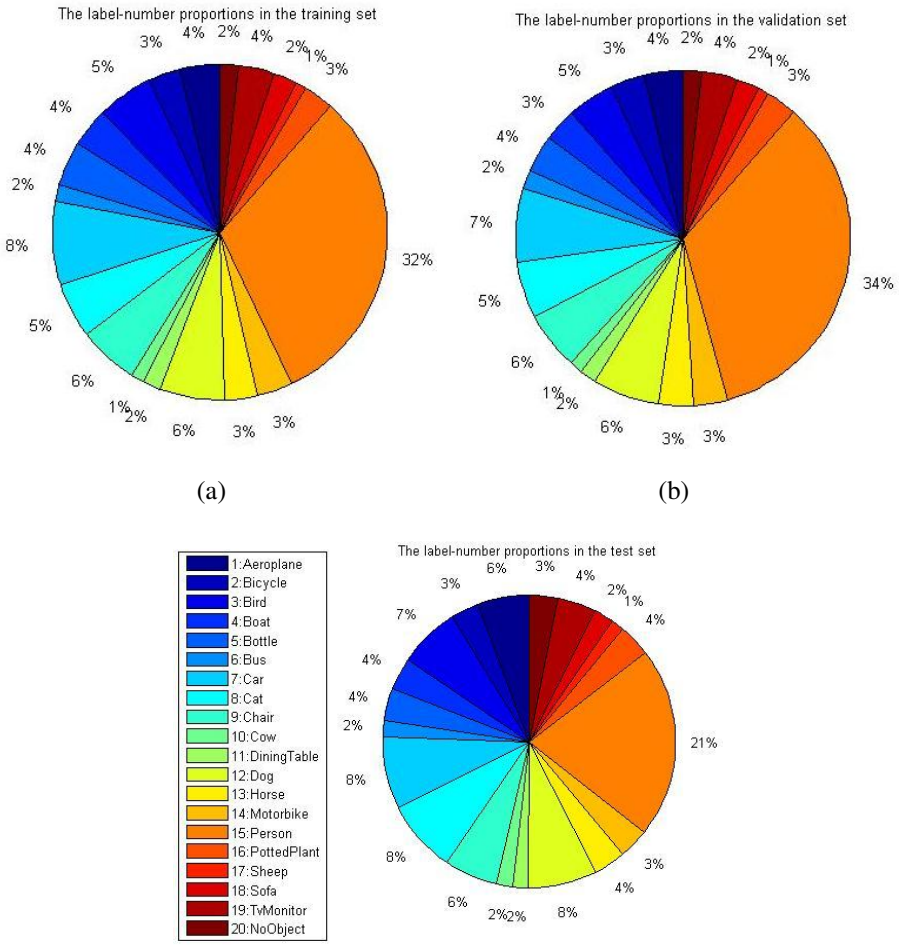


Figure 3.2: *The class-number proportions of all training (a), validation (b) and test (c) instances.*

to avoid being confused by irrelevant information.

- High **intra-class variability**. This means that the objects occurring in images of the same class, seem in fact very different visually. This is not only due to the natural different structure of objects (for example two persons cannot be identical) but also due to the different conditions in which they are depicted, namely lighting conditions, scaling, angle etc. As [Opelt et al., 2004] claims, even two identical objects could be hard to recognize under this kind of conditions.

- **Occlusions**, i.e when only a part (sometimes even less than 5%) of the object is visible in a picture. Indeed, in the PASCAL dataset there are pictures annotated as “person” although the only part of the person depicted is a head with a helmet which leaves only the eyes to be visible.

All of the above properties exist in the PASCAL dataset, as images are taken from natural scenes (they are taken from *flickr* [4]). Finally, another challenge of the dataset is that of **unbalanced data**, meaning that there are certain classes which are far more popular than others, among instances. For our particular separation into training, validation and test set, we have the proportions described in figure 3.2. Note that these proportions, for example for a class c, have been created by counting all instances belonging to class c and dividing with the total number of *labels* existing in the sets, rather than with the total number of instances (these two numbers are different because there are instances which belong to more than 1 class).

As one can see, the "Person" label is extremely dominant, occupying about the $1/3$ of the whole label-set for training and validation set, and approximately the $1/5$ for the test set. Moreover, there are labels such as "Sheep" and "Sofa" which occupy only the 1% of the pie-chart. In other words, a person is as much as 34 times more likely to appear in a picture than a sheep. This imbalance results in **three major problems**. Firstly, for classes with a small percentage in this pie chart the learning algorithm has too few examples to learn from. The basic intuition behind all learning algorithms is to use a pile of images with an object X and another pile without this object and extract the relevant information co-occuring with the particular object. So, the greater the number of images in the pile, the better. Secondly, most of the learning algorithms, as well as the SVM model and the LDA-based one that we use in this project, tend to have a *bias* for the most dominant classes when the task is multi-class classification. This problem is analysed in detail in the evaluation section. Finally, such imbalance in the data makes the evaluation of the algorithm a difficult task. When a single number of the performance of the algorithm across all classes is needed, there is the option of weighting the performance for each class equally or according to its prior. For such differences in the priors these two methods would produce very different results, and further analysis is required. This problem is, again, described in the evaluation section.

3.2 Feature Extraction

In order for a learning algorithm to be able to process the visual dataset, each image must first be transformed into a feature-set which reflects its visual contents in the best possible way. This problem (called *feature extraction from images*) is purely a computer vision one and forms the basis and most important part of the entire process of

[4] http://www.flickr.com/

object categorization. This is obvious if we consider that a feature extraction process which could -ideally- produce features containing only the relevant information about the depicted objects in an image, would in fact solve the object recognition problem itself. However, as this is not possible, further algorithms (like classification) are required in order to learn from the statistical properties of these feature set.

The overall procedure of extracting and representing features is, basically, a sequence of 3 steps, as shown in figure 3.3.

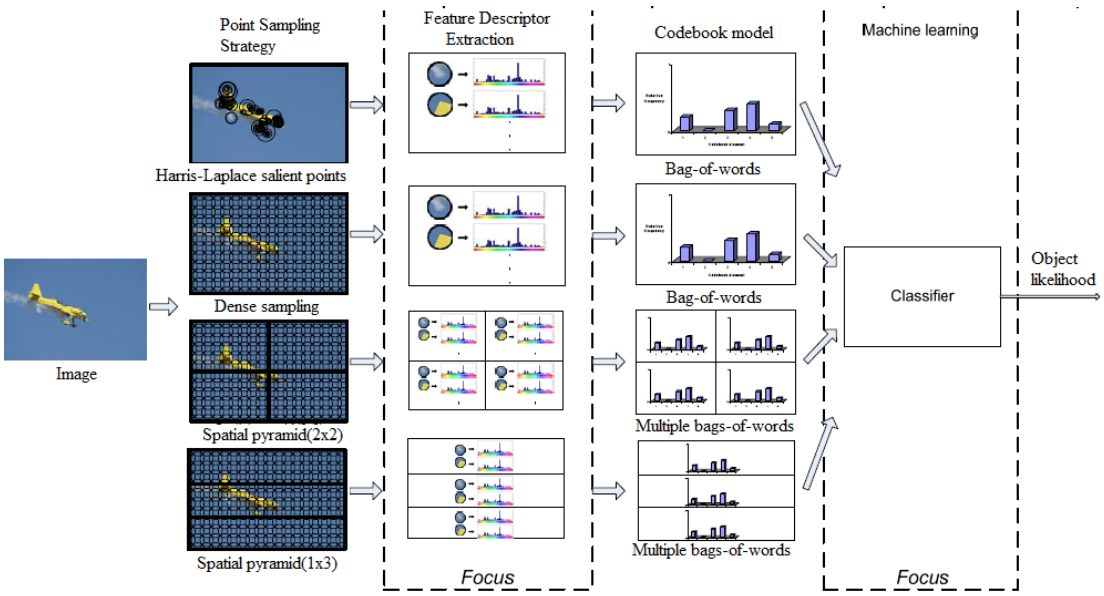


Figure 3.3: *The overall procedure for object categorization, with a focus on feature extraction. source: http://staff.science.uva.nl/ ksande/research/*

Firstly, a ***feature detector*** is applied to the image in order to find its *interesting parts* (for example edges, smoother areas etc). Then, a ***feature descriptor*** creates the final features by finding a way to describe the properties (which include color, texture etc) of the interesting parts detected in the previous step. The set of all features is called ***codebook***. Finally, a ***visual codebook representation*** is created, which basically is the set of attributes that a learning algorithm would receive as input. Different kinds of codebook representations imply different assumptions about the dependencies between the features. Note that for this section only we use the term “attribute” (which refers to the codewords, the input of the learning algorithm) to distinguish it from the term “feature” which is basically a descriptor of an interest point. These notions are similar in the sense that features form codebooks and a codebook model is applied in order to form the final attributes. Features characterize the image, while attributes are learned from the whole database, based on the feature-set. In general, though, when the use of “feature” is clear within a context, the term can be used in both cases.

Further preprocessing can be applied to the feature-set or to the attribute-set, upon the completion of the above process. For example, *dimensionality reduction* techniques could reduce the dataset size as well as the embeded "noise" in the data. Furthermore, *attribute selection* [Guyon and Elisseeff, 2003], which is basically a form of dimensionality reduction, could be applied in order to select the attributes which are the most informative and useful for the learning algorithm. A lot of other statistical techniques could further pre-process the dataset, especially after performing *exploratory data analysis* on the data and acquiring specific, domain knowledge. However, in this project we limit the feature extraction procedure to the basic steps of feature detection and description, codebook creation and, finally, image representation based on the codewords. The reasons why we chose this path are the following:

- The purpose of the project is to evaluate the performance of a particular kind of methods to object categorization, by comparing the results with a baseline model. Thus, our efforts are focused on comparison and result analysis, rather in building the optimal learning algorithm.

- Given the above, we also focus on the general principles of the algorithms being discussed, and we want the whole procedure to be as generic as possible.

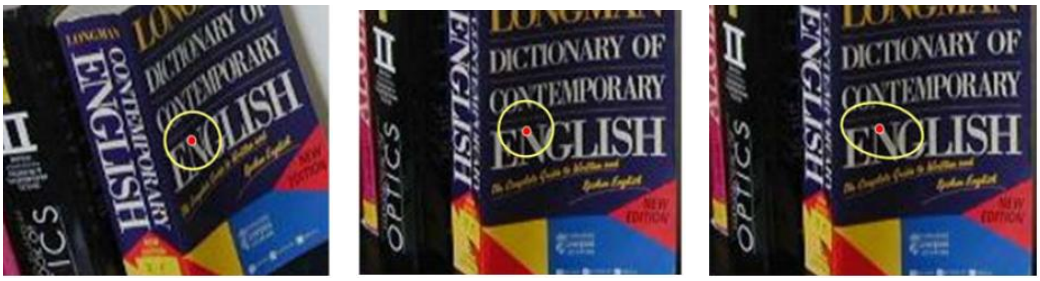
- Most of the pre-processing techniques are mostly suitable for datasets with two or, in general, few classes. For example, feature selection is usually done when the objective is to find those features that lead to the best prediction of a particular class.

Obviously, feature detection and description are low-level image processing methods but, fortunately, a lot of work has been done in this field. This allows researchers to build a feature extraction framework by relying on existing implementations which are responsible for the low-level procedures required for image processing. What is more, many publications (like [Grangier et al., 2006]) compare the existing approaches in a higher level. By relying on publications and software like the ones described we are able, thus, to choose a particular approach based on our specific needs, without having to perform a deep research into the low-level image processing algorithms and theory. In the rest of this chapter we explain in greater detail the feature extraction steps in the depth that is needed in order to be able to reason about our approach; we discuss about the possible options as well as the ones that were finally selected for this project, we reason about our selection and describe the implementation details.

3.2.1 Feature Detectors

Feature detection is also referred to as *sampling*, because it seeks for a very small sample of pixels given the whole image. A basic property that such methods should fulfill, is *repeatability*, i.e if there is a transformation between two instances of an object in different pictures then the corresponding points should (ideally) be detected [Csurka et al., 2004]. For the purpose of image classification, it is obvious that high repeatability is desirable, as we would expect similar detections for images which depict objects of the same class. In figure 3.4 we can see three images along with a detected region (yellow circle): the original on the left and two transformations in (b) and (c). In case (b) the detector uses a fixed-scale scanner and is unable to capture the same part of the image as in (a), while in case (c) the detector has high repeatability and uses a scanner which re-scales in order to capture the same part of the image even after the transformation has been applied.

[Lindeberg, 1996] show that scanning the images in different scales may result in very different outputs. Thus, there is a single scale for which the results are better. In order to define it, one option would be to scan the image multiple times, at many scales. However, this is time consuming and suffers from various other problems. *Automatic scale selection* is a technique which is used to avoid multiple scanning in many scales and to achieve high repeatability. It refers to "an explicit mechanism which automatically adapts the scale levels to the local image structure" [Lindeberg, 1996], even when the algorithm is applied to unknown environments.



(a) (b) (c)

Figure 3.4: *An example which illustrates repeatability: a transformation is applied to the original image (a) resulting in images (b) and (c). The detector applied in (b) is lacks of repeatability.*

There are five basic approaches for feature detection: *edge detection*, *corner / interest point detection*, *blob / region detection*, *ridge detectors* and simple *dense sampling*. However, the distinction between these groups is not clear and there are approaches

which combine these ideas, as well as approaches which are classified to a different category by different researchers. In particular, corner detectors and blob detectors share some common properties. Note that ridge detection is not commonly used for object recognition purposes, and for this reason it will not be further explained.

Edge detection (figure 3.5) is the technique of identifying edges in an image. An **edge** is typically defined as a discontinuity in the physical, photometrical and geometrical property of a visual object ([Ziou and Tabbone, 1998], [Lindeberg, 1996]). This kind of discontinuities are usually a result of a variation in the material properties, illumination, depth and orientation of surfaces. The motivation behind this idea, is that identifying such points would result in finding a boundary between two image regions. In turn, the sets of boundaries -represented as connected curves- is likely to form the boundary of objects depicted in images. [Ziou and Tabbone, 1998] groups typical edge-types into *steps*, *lines* and *junctions* according to their "properties" and the type of phenomena that causes them. For example steps, for which the authors argue that is the most common edge-type, are the points where the grey level discontinuities occur, something that is usual in the parts where two objects meet or a shadow is cast.

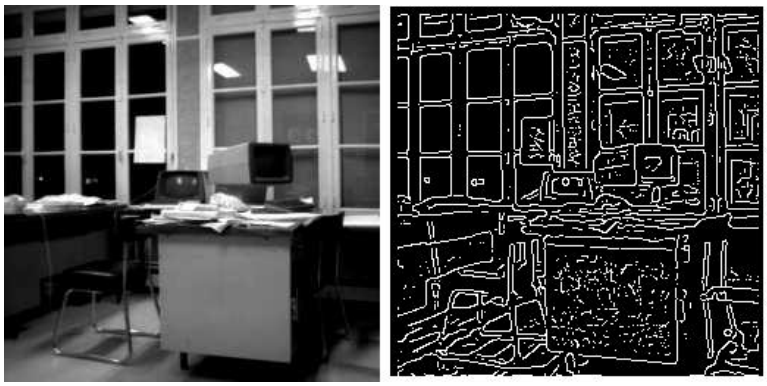


Figure 3.5: *The original image and the one with the edges detected by an edge detector. Obviously, sets of detected pixels form the outline of objects. source: [Ziou and Tabbone, 1998]*

Interest point detection (also, **salient point detection**) seeks for *interest points*, which can informally be described as points in an image which have a concrete mathematical definition (for example a corner), well-defined location and are invariant to *viewpoint changes* such as scaling and rotation -thus having high repeatability. There are also detectors which are invariant to affine transformations too [Mikolajczyk and Schmid, 2004] (in fact, scale transformations are a special case of affine transformations). However, the term "interest point" is general of its nature, and the relative ap-

proaches should define what kind of information the detected points provide. **Corner detection** (figure 3.6) is a special case of interest point detection, as a corner fulfills the properties described above. Scale and affine invariance are, thus, the most important properties that an interest point detector should have. Intuitively, these properties suggest that the algorithm should be able to identify the similarity of the same regions in different images, even if affine or scale transformations have been applied to them. However, the requirement for a detector to be truly scale and affine invariant is just ideal. In practice, the magnitude of these factors' invariance is one of the measures used to evaluate detection algorithms [Schmid et al., 2000]. A comparison and explanation of such methods can be found in [Mikolajczyk and Schmid, 2004].

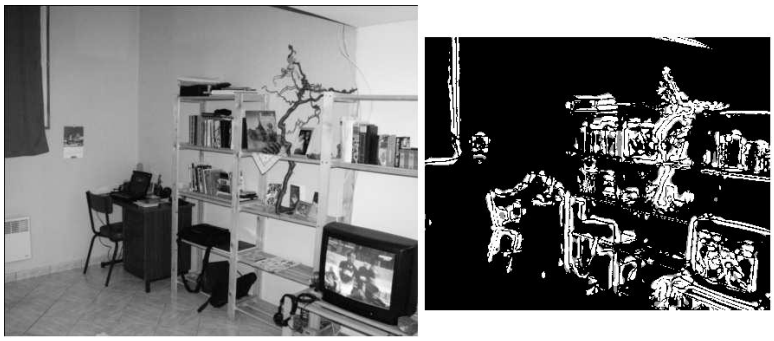


Figure 3.6: *The original image and the one with interest points (coreners) detected by the harris-laplace corner detector. source: [Mikolajczyk and Schmid, 2001]*

Blob / region detection (figures 3.7 [5] and 3.3) is a method similar to interest point detection. Indeed, there exist algorithms that are classified in both groups. These approaches detect interest points too, however, they can also detect regions of interest points which are too smooth for a corner detector to capture. [Mikolajczyk et al., 2005] explains that the term "region" in this context refers to a set of pixels, i.e any subset of the image. The authors also underline that this is opposed to the traditional use of the terms "region" or "blob" in the context of *image segmentation*, since the region boundaries do not necessarily have to correspond to discontinuities of the surface's texture or color. Moreover, as it is evident from the experimental results of [Mikolajczyk et al., 2005], regions detected by such algorithms are usually very small in comparison to the whole picture. For example, in [Grangier et al., 2006] one can see this different usage of the term "blob". It is worth mentioning that corner detectors combined with automatic scale selection, such as the *Harris-Laplace corner detector [Mikolajczyk and Schmid, 2001]* (which is also depicted in 3.3) are very common.

[5] source: http://www.kki.yamanashi.ac.jp/õhbuchi/online_pubs/ACM_CIVR2009/CIVR09_web.pdf

Figure 3.7: *The regions detected by the region detector Harris-Affine.*

Finally, **dense sampling** (for example used by [Horster et al., 2008]) refers to the simplest method: a grid (of any shape) is applied over an image and samples pixels. The most unsophisticated version is when every d pixels are sampled, with d being an input parameter, called *the spacing parameter*. Typical dense sampling algorithms automatically adjust the scale at which they sample by taking into account the image's dimensions and the spacing parameter. However, there is also the option of sampling at more than one scales. An example of dense sampling is shown in figure 3.3 as well as in figure 3.8 [6].

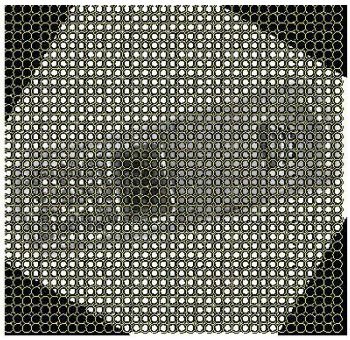


Figure 3.8: *A grid applied to an image for dense sampling.*

It is worth noting here, that the *spatial pyramids* concept [Schmid, 2006], shown graphically in figure 3.3, has to do more with the codebook creation rather than the sampling process itself. Although any feature detection algorithm can be used in the first step of the feature extraction process when spatial pyramids are used, slight modifications must be applied in order to form the spatial pyramids model, and this is the reason why it is shown in the first part of the process as well.

[6] source: http://www.kki.yamanashi.ac.jp/õhbuchi/online_pubs/ACM_CIVR2009/CIVR09_web.pdf

3.2.2 Feature Descriptors

While feature detection finds the regions or points in an image which are the most "interesting", feature descriptors extract the actual features out of these locations. There are two basic approaches for producing a feature set. The first one concerns the computation of *local* features, i.e features which are computed over a limited spatial support [Csurka et al., 2004]. The feature detection algorithms described above are usually combined with local features producing compact vectors which characterize the region around interest points, and this is why we will focus our description on them. However, in section 3.2.3 we discuss about a more general kind of features. For this subsection, the term "features" will refer to local features.

Given the great importance of a good feature-set for any task that requires it, feature descriptors must be repeatable (a term used for feature detectors as well). The concept of repeatability is, thus, extended for the context of feature descriptors and -in this context- can be described as the property which defines the following: if there are two images depicting the same object under different viewpoints and their corresponding interest points are detected, then we would ideally expect the corresponding values for the features to be identical. For image classification tasks a high degree of repeatability is necessary, and is achieved with methods that are invariant to changes of viewpoint (scaling transformations etc). Other desirable properties of feature descriptors include robustness against geometric and illumination variations [Csurka et al., 2004].

Unlike feature detectors, descriptors cannot be easily grouped into categories in a higher level. A feature descriptor is basically characterized by the type of properties which tries to extract from the detected areas (for example, texture, color etc), by the specific algorithm used in order to achieve this (many algorithms use heuristics) and, finally, by their robustness to variations of viewpoint. Therefore, a reasonable way to start a research in order to find the best descriptor for a particular task is to examine the state of the art and focus on the experimental results acquired by studies which investigated similar tasks.

Scale-invariant feature transform (SIFT) features [7] [Lowe, 1999] are a very popular choice for vision tasks which involve image categorization ([Grangier et al., 2006], [van de Sande et al., 2008], [Csurka et al., 2004], [Horster et al., 2008]). They represent the local shape of a region as a sequence of 128-dimensional vectors using edge orientation histograms. The way in which they are computed are beyond the

[7] this algorithm is patended by the University of British Columbia, http://user.cs.tu-berlin.de/ nowozin/autopano-sift/

scope of our research in this project, but it is worth mentioning the algorithm's attractive properties which made it so popular. These, as already discussed, result in high repeatability. More specifically, SIFT are invariant to image scaling, rotation, translation and partially invariant to illumination changes and affine or 3D projection. They are also highly distinctive (only a few number of SIFT features computed on a visual object can capture a lot of "useful" information about it) and robust to noise. Although they are easy and fast to extract, they produce good results in the object recognition task even if images are partly occluded and in a cluttered background, as proved in [Lowe, 1999].

However, the first step of an algorithm for SIFT-feature extraction is to convert the input image to grey-scale. This means that they provide an inadequate description of the region's color properties despite the fact that for many object recognition tasks this is essential (for example the skin or the sky can better be identified in images if richer color information is present). Consequently, many approaches try to develop new algorithms, or even to extend the SIFT algorithm in order to include richer color information as well. Many versions exist and their main difference is the *color model* that they use for color representation, for example *HSV, color moments* etc. The term "**color descriptors**" can be used for the whole category of these algorithms, and a very interesting description and comparison can be found in [van de Sande et al., 2008].

3.2.3 Global versus local features

In section 3.2.2 the discussion was focused on local features. However, **global features**, which are applied in larger areas of a picture and capture more general properties of it, can also be used. In section 3.2.1 we made a distinction between blob-like regions detected by blob/region feature detectors and blobs created via segmentation. Only in this section, we will refer to a **blob** as being *the feature set of a large part of an image created via segmentation*, i.e we won't use the term with the meaning that was assigned to it in section 3.2.1. The usage of such features are more usual in the context of automatic image annotation. [Grangier et al., 2006] and [Barnard et al., 2003] are two examples.

Blobs are created via image segmentation, for example with the *Normalized Cuts (N-cuts)* algorithm of [Shi and Malik, 1997] and tend to be large color-uniform regions with features describing its colors, texture, shape and location [Grangier et al., 2006]. Just like feature detectors, segmentation can be thought as a uniform-region detector.

However, the standard way of treating such features is not a bag-of-words representation, but instead iterative procedures such as the one described in [Duygulu et al., 2002].

3.2.4 Visual codebook model

The Bag-of-Words model is a usual choice for representing visual codebooks of local features. When this generic model is applied to visual features, the resulting representation is also called *bag of visual words* or *bag of visterms (BOV)* [Quelhas et al., 2005] or *bag of keypoints* [Csurka et al., 2004]. Unfortunately, given that this method follows the assumptions mentioned in section 2.2.1, it disregards all information connected with the spatial layout of the features and thus, is less descriptive. By treating each individual feature independently, properties of the whole or a large part of an object, like its shape, can never be modeled, because this would require several features to be related in some way. Even worse, the separation of an object from its background is not obvious. However, overcoming this caveats is considered to be a challenging task and only few efforts have been made towards this direction [Schmid, 2006].

3.3 Implementation and final datasets

In this section we reason about our choices regarding the issues described in 3.2 and provide implementation details. As already mentioned, the main objective of this project is the performance evaluation of a topic model based algorithm by comparing it with a baseline. Thus, our focus has not been given into extracting the best possible features, but the main interest regards the development of a feature-set that is as proper as possible for the particular task that we have, i.e performing object categorization using an LDA-based algorithm from the image annotation domain. This does not mean that we bias our choices to favor the LDA-based model. In fact, the SVM model used as a comparison is just a baseline, and the only way in which we try to boost its performance is by finding a feature set which is “objectively” good, while for the LDA-based classifier we do want to determine the feature-set’s aspects that are more appropriate for our model. All these aspects are explained in the rest of this chapter.

Local features

In this project we decided to use local features instead of blobs for the reasons described below. Firstly, because in connection with the standard way of being represented (a bag of words model) have demonstrated "impressive levels of performance" for object recognition tasks [Schmid, 2006] as well as for tasks which involve the training of a topic model [Horster et al., 2008]. The fact that [Horster et al., 2008] claim that PLSA performs better using local image characterizations gives a reasonable argument that this kind of features would lead to better performance for our LDA-based model as well. The authors present the opinion that local features are more flexible than global ones and manage to reveal more meaningful patterns in images. Global features are usually used when the objects depicted in images are greatly correlated with their background and, thus, modeling the entire image can actually help recognition. However, in a challenging dataset like PASCAL where objects are presented in cluttered backgrounds and can be occluded, we need more flexible descriptors.

Corner detection and dense sampling

For **feature detection** we chose to experiment with the corner sampling and the dense sampling strategy for the reasons explained below. Regarding dense sampling, it seemed like a good option for our case, as [Fei-Fei and Perona, 2005] proved that this kind of feature detection is better than interest point sampling when the classifier is LDA-based. This can be explained intuitively, if we consider that a grid which is applied to an image can collect information about uniform regions such as sky or grass. Although this property is highly desirable for the scene categorization task of [Fei-Fei and Perona, 2005], in our case it is not obvious how it could lead to a better feature set. However, given that our LDA - based model is capable of capturing some very general relations among the features of the codebook, we expect that this approach would have good results. For example, the presence of a visual word which is highly related with sky could be thought as a bias towards predicting the class "aeroplane". Moreover, [Nowak et al., 2006] found that even *random sampling* is often more discriminative than a keypoint-based one for object categorization tasks, supporting the idea that for such problems simpler solutions may be better.

As for the **implementation**, given that the winners of the PASCAL challenge 2008 [van de Sande et al., 2008] provide for free their feature detection and description software [8], it is a reasonable choice for our project as well. We used it to sample every 6th pixel of each image. The software selects automatically a good scale for sampling.

[8] software available at: http://staff.science.uva.nl/ ksande/research/colordescriptors/

As all images in our experiments are approximately of the same size, the same number of points is computed for each image (about 4971), which is convenient for the clustering that follows the feature detection and description procedures, as explained later in this chapter.

However, it is reasonable to test a more sophisticated method as well. Therefore, we also created a different feature set using the Harris-Laplace [Mikolajczyk and Schmid, 2001] corner detector (with the same software) in order to take advantage of its invariance properties. The motivation for this choice is that our challenging dataset requires a detector more robust to changes of viewpoint. Using this detector we obtained various number of interest points for different pictures. For most of the pictures this number ranged between 600 and 2500.

SIFT and transformedColorSIFT feature descriptors

As for feature descriptors, we chose SIFT and a variation of SIFT which also encloses color information. More specifically, [Csurka et al., 2004] present various arguments as to why SIFT features can be used in conjunction with a bag of visual words representation. For example, the authors argue that since these features are simple linear Gaussian derivatives, they are expected to be less affected by noise, in comparison with descriptors which are higher Gaussian derivatives. But the most dominant reason why we think SIFT will be ideal for our task, is that previous research has shown that SIFT outperform other approaches in tasks similar to the one attempted in this project. For example [Mikolajczyk and Schmid, 2005] compared different approaches in an image database similar to the one that we use and found that SIFT perform the best. This kind of descriptors is also preferred from researchers, for the particular dataset (PASCAL) that we use in this project [Nowak et al., 2006] or for the particular kind of classifier (one based on topic models) that we also have [Quelhas et al., 2005].

However, as already mentioned in section 3.2.2, SIFT descriptors transform the input image into grey-scale and consequently ignore valuable color information. [van de Sande et al., 2008] compared 15 different color descriptors on the PASCAL dataset 2007 and found that only the ones which are combined with SIFT can outperform pure SIFT. Given that the authors of [van de Sande et al., 2008] were the winners of the PASCAL 2008 challenge, i.e the classification competition on the exact dataset that we also use, it is rather reasonable to base our own descriptor selection on their approach. They actually use a combination of 5 different color descriptors with equal weight, but for our purposes we believe that using one of these color descriptors is enough. We chose

the *transformedColorSIFT*, which is proven to outperform pure SIFT in the PASCAL dataset because, as the authors argue, the extra color information which they include can be used in order to increase discriminative power and illumination invariance. This kind of descriptors add extra information to the 128-length vector that normally SIFT produces and this results in a 384-length descriptor vector for each interest point.

Bag of visual words codebook representation

Since local features were chosen for this project, a bag of visual words (BoV) is a natural choice for the codebook. Although in section 3.2.4 many caveats are mentioned for this type of representation, it still is a good option. It is true that there are, theoretically, better approaches such as spatial pyramids [Schmid, 2006], however, none of these alternatives has been tested to the extend that BoV has been tested and proved its effectiveness. Moreover, for the special case of applying topic models for object categorization purposes, it has been shown [Quelhas et al., 2005] that, in a sense, they disambiguate the bag-of-words representation. More specifically, a direct effect of the implied independence that the BoV imposes, is that problems of polysemy (one visual word may represent two or more parts of different objects) and synonymy (the same part of an image may be represented equally by many different visual words) arise, exactly like one would expect in the linguistics domain. For example, in figure 3.9 each row represent sets of similar visual words and polysemy is obvious because, as can be seen, completely different aspects (eyes and windows/city environment) are grouped together.

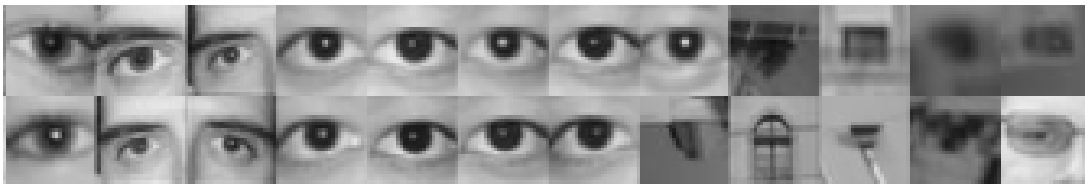


Figure 3.9: *Polysemy in visual terms. source: [Quelhas et al., 2005]*

However, as [Hofmann, 1999] clearly proves, topic models are capable of handling successfully polysemy and synonymy for text data, and we expect the same to happen for visual data as well.

As for the **implementation** details of how each image is represented into a bag of codewords, we followed the classical procedure described in [Csurka et al., 2004]. Firstly, we use K-means to cluster all feature descriptors from all images in the *training* database using K centers. The choice of K is very important because it will constitute

the number of attributes in the final representation of our training dataset. Given that there is an average of about 1500 descriptors per image with each descriptor being a vector of 128 (for SIFT) or 384 (for transformedColorDescriptors) elements and that all images in the training database are more than 2000, the system requirements, mainly in memory and computational speed, exceed the available ones. Thus, we have to select a representative subset of all features and perform clustering on these only. Many researchers select this subset randomly. In our project, we select every X features from each image uniformly. We set X to a value such that at least 5 features are sampled from each image and the final number of features does not overpass the limits of our computational resources for clustering (i.e the maximum matrix dimensions that MATLAB can handle). In the next step, we use MATLAB to vector-quantize descriptors. Every local descriptor d is mapped to number i, i.e the label of the cluster C which is closest to it by using the neighbor rule:

$$d \mapsto i \leftrightarrow dist(d,C_i) \leq dist(d,C_j), \forall j = 1,...,K \tag{3.1}$$

Finally, each image is represented as a single vector $< I_1, I_2, ..., I_K >$, where I_i is equal to the number of times that any of its descriptors was mapped to the class-label i divided by the total number of descriptors in that image, i.e it is a **normalized count vector** and all of its elements sum to 1. Alternatively, we could have used a count vector without normalization, but when the number of features is large normalization usually helps. Using an unnormalized count vector results in discrete attribute values, while using a normalized count vector results in continuous attribute values, an issue which will also be discussed in section 5.1, as for each case different distributions are most appropriate.

Returning to the issue about how to set K, i.e the codebook size, [Nowak et al., 2006] and [Csurka et al., 2004] suggest that intermediate to larger codebooks result in lowest classification error. Thus, we clustered the descriptors with 300 centers, the highest possible number given our computational resources, since the points to be clustered are thousands and high dimensional.

Final datasets

Given all the above selections for feature extraction, we ended up with 3 different feature-sets plus an additional one [9] obtained from *http://www.inf.ed.ac.uk/teaching*

[9] Although this feature-set was not created by us, detailed description of the steps followed during its creation are available at: http://www.inf.ed.ac.uk/teaching/courses/iaml/assts/a108.pdf , and thus, it is comparable to our own feature-sets.

/courses/iaml/. All of the feature-sets are also summarized in table 3.1. Note that we used the default parameters of the software for the Harris-Laplace detector. Also, a code-name has been assigned to each feature-set as an id.

Feature Detector	Detector Parameters	Feature Descriptor	Codebook size	code-name
Dense Sampling	15 pixel-grid	SIFT	500	dSIFT500
Dense Sampling	6 pixel-grid	SIFT	300	dSIFT300
Harris-Laplace	harrisThreshold=1e-9 harrisK = 0.06 laplaceThreshold = 0.03	SIFT	300	hSIFT300
Harris-Laplace	harrisThreshold=1e-9 harrisK = 0.06 laplaceThreshold = 0.03	transformed ColorSift	300	hCSIFT300

Table 3.1: *The final feature-sets produced.*

Note that for each different feature-set the same splitting technique is applied, as described in section 3.1, resulting in one training, one validation and one test set for each.

The datasets described above are ready for general use. They are basically text files where each row represents a document whose features (K in total, where K is the number of clusters used in the procedure described above) are separated with a space. However, different algorithms need the input in different formats. For the SVM program we only had to add the class labels of the training set. For the case of our LDA-based algorithm, we had to represent each dataset as two parts: the first contains only the labels and is treated as a corpus of text, while the second contains only the visual features. There is a 1-1 corresponence between rows from each dataset: a row i in the labels' file is actually a "caption", and is the right one for the image of line i of the visual features' file. This is done for reasons that will become clearer in chapter 5.

Chapter 4

SVM model

In this project, we treated the visual object categorization challenge as a traditional classification one, where each object's name corresponds to a class label. Various researchers, such as [Foody and Mathur, 2004], found that SVM classifiers are ideal for image classification. Consequently, we chose this kind of classification algorithms for our purpose. However, as SVM's are binary classifiers, the original multi-class problem has to be decomposed to binary (2 class) problems. We chose the 1-vs-1 method to do this, as described in [Friedman, 1996], and multi-class probability estimates are acquired for the whole set of classes. The award-winning libSVM toolbox [Chang and Lin, 2006] was used for the implementation.

The term SVM refers to a general family of algorithms rather than to a specific method. Different choices for its kernel, its parameters and its way of solving the quadratic problems result in very different models (as described in section 2.1.2). For this reason, we experimented with different combinations of these options (our experimental set-ups were based on findings described in existing publications), and we based our final selection on these results.

The rest of the chapter explains in more detail the properties and principles discussed above and analyses the approach and methodologies used in this project for the specific problem and multi-class dataset that we have.

4.1 Notes on the implementation and the software

The libSVM [1] toolbox [Chang and Lin, 2006] was used for our purposes. As mentioned above, there are many issues to consider when using a SVM model and each

[1] available at: http://www.csie.ntu.edu.tw/ cjlin/libsvm/

combination of choices leads to different models. The libSVM software has some built-in features which cannot be changed without interfering with the source code but the authors argue that they embedded the most optimal choices. For example, it uses a specific approach for decomposing the multi-class problem (although we implemented a perl script for a second one) and for solving the quadratic problems. However, it also leaves some freedom as regards parametrization. What is more, the fact that it is not based on an API makes it extremely flexible. We found this flexibility to be very important in our experiments, because it allowed us to combine the program with multiple Perl, system or even MATLAB scripts. Finally, the software is limited to the actual classification problem and is incapable of performing evaluation measurements, especially for the multi-class problem. Thus, we had to program in Java and in MATLAB our own evaluation code.

Finally, it is worth mentioning that libSVM uses "an SMO-type Decomposition method" for solving the quadratic problem (2.2) and the one needed to find the α's in equation (2.3). The analysis of this method is beyond the scope of this project and can be found in [Chang and Lin, 2006].

4.2 Dealing with the multi-class problem

There are many approaches which can be used to solve a multi-class problem with SVM's, which were initially designed for the special case of 2-class tasks. In general, there are two kinds of methods: the first considers interfering with the optimization function (2.2) of the model in order to be able to extend its capabilities for more classes [wei Hsu and Lin, 2001]. However, this leads to a difficult and complex optimization problem. The second, and most common solution, is to *decompose* ("binarize") the original problem into many binary ones. Then, probability estimates can be found for each of these subtasks which are finally transformed into probabilities for each class in general. In this project we use the second approach, as was made obvious from the background section 2.1.3.

4.2.1 Decomposing the multi-class problem

The two most popular methods for transforming a multi-class problem into many binary ones, are *one-against-all (1-vs-All)* and *one-against-one (1-vs-1, or pairwise coupling)*.

1-vs-All is the most common strategy [Gidudu et al., 2007] and defines the algorithm in which the original problem with C classes is transformed into C new binary problems which are solved in the manner described in section 2.1.2. More specifically, for every class label $c \in C$, a new binary problem is constructed which involves classifying instances with labels c and "not-c". For example, for the class "Person" in our dataset, every instance belonging to this class is assigned label "1" and the rest are assigned label "-1", without examining which other objects are also depicted in the same image. The score of classification for class c is simply the score achieved in the specific subproblem created for this label. In terms of implementation, thus, this approach involves creating C copies of the original dataset and modifying the class-labels assigned to instances in each one.

On the other hand, **pairwise coupling** decomposes the original problem to $N = C(C-1)/2$ binary ones, where each subproblem involves learning to distinguish between a pair of classes (ignoring the rest $C-2$). This can be achieved by modifying the learning rule of equation (2.2) in the way shown by [Chang and Lin, 2006]:

$$\begin{aligned} \min_{\mathbf{w}^{ij},\mathbf{w}_0^{ij},\xi^{ij}} \quad & \frac{1}{2}\left(\mathbf{w}^{ij}\right)^T \mathbf{w}^{ij} + C\sum_{t=1}^{n}\xi_t^{ij} \\ subject\ to: \quad & y_t^{ij}\left(\left(\mathbf{w}^{ij}\right)^T \phi(\mathbf{x}_t^{ij}) + \mathbf{w}_0^{ij}\right) \geq 1 - \xi_t^{ij} \\ & \xi_t^{ij} \geq 0, t = 1...n \\ if\ \mathbf{x}_t^{ij}\ belongs\ to\ class\ i: \quad & y_t^{ij} = 1\ else\ y_t^{ij} = -1 \end{aligned} \tag{4.1}$$

In the testing phase, a test point is classified with all N classifiers obtaining, thus, a vector of N labels and a voting system is used in order to get the final prediction. More specifically, the label which is most frequent in the vector of N labels is the one emitted as the final prediction (a voting strategy that is also known as "Max Wins"). However, one should decide how to break potential ties.

In this project we had the freedom to choose any of the approaches mentioned above. libSVM implements the pairwise coupling one but we also implemented a perl script for the 1-vs-all approach, since it is very straightforward. However, we chose to use the first method, based on the findings of [wei Hsu and Lin, 2001] and [Gidudu et al., 2007]. More specifically, although these studies were conducted independently and on a different kind of datasets, they both conclude that 1-vs-1 and 1-vs-all result in almost the same performance with 1-vs-1 being slightly better. However, we noticed that 1-vs-1 is faster during training time, which at first seems to be a paradox, since it builds $C(C-1)/2$ new problems in contrast with the 1-vs-all approach which creates

only C. Our observation, though, is consistent with [wei Hsu and Lin, 2001], which goes one step further and explains that this is due to the fact that the sub-tasks created by pairwise coupling are actually smaller, since they concern a subset of the whole dataset, i.e instances which belong to any of the two classes to be tested as a pair. Given that finding good parameters is a far more critical issue and that there is a trade-off between training time required and number of experiments conducted, we decided to follow the pairwise coupling technique.

4.2.2 Probability outputs

SVM's base their prediction for a binary classification problem on the sign of the real valued function f (equation (2.4)). However, as already mentioned, we need to find a way to predict probabilities. More formally, for a data point $\mathbf{x}$ and for each of the C different classes we aim at obtaining:

$$p_c = p(y = c|\mathbf{x}), \;\; c = 1, ..., C \tag{4.2}$$

Given that we follow the one-vs-one approach, the problem can be formalized as follows:
Firstly, the pairwise class probabilities must be estimated:

$$r_{ij} \approx p(y = i|y = i \;\; or \;\; j, \mathbf{x}), \;\; i, j = 1, ..., C \tag{4.3}$$

This can be done by simply mapping the real values of function f (see equation (2.4)) into ones belonging in the interval [0,1]. Fitting a sigmoid to the output of f is a straightforward solution. However, the libSVM software, which we use, follows an improved technique proposed by [tien Lin et al., 2003]:

$$r_{ij} \approx \frac{1}{1 + e^{Af+B}} \tag{4.4}$$

where A and B are parameters to be estimated using the Maximum Likelihood (ML) approach for the training dataset, as described in [tien Lin et al., 2003] in more detail.

Secondly, the actual class probabilities p_i as defined in equation (4.2) can be computed based on r's. Again, various approaches exist for this issue. [fan Wu et al., 2003] review two of them and suggest another two techniques. libSVM software implements the second approach described in this publication and is based on the idea that:

$$p(y = i|y = i \;\; or \;\; j, \mathbf{x}) \cdot p(y = j|\mathbf{x}) = p(y = j|y = i \;\; or \;\; j, \mathbf{x}) \cdot p(y = i|\mathbf{x}) \tag{4.5}$$

This observation leads into the following optimization problem:

$$\min_{\mathbf{p}} \quad \frac{1}{2}\sum_{i=1}^{C}\sum_{j:j\neq i}\left(r_{ji}p_i - r_{ij}p_j\right)^2 \tag{4.6}$$

$$subject\ to \quad \sum_{i=1}^{C} p_i = 1, p_i \geq 0,\ \ i = 1,...,C$$

libSVM software uses an iterative procedure (which is described in [Chang and Lin, 2006]) in order to solve this optimization problem.

4.3 Choosing a kernel

In section 2.1.2 the formulations of the basic kernel functions were introduced. It was also mentioned that choosing a suitable kernel is vital because different kernels result in different mappings of the data space to the feature space. libSVM allows to use any of the kernels mentioned in section 2.1.2 and, thus, we had to base our selection on some experiments, designed on the basis of research in the relevant literature. For the reasons explained below, we decided to experiment with a *linear* and an *RBF* kernel.

First of all, an RBF kernel is a common choice as it is able to capture the nonlinear relationships between the actual class values and the attributes. [Eichhorn et al., 2004] test combinations of kernels and show that a RBF kernel (used as a minor in that case) is the best for SIFT features extracted for object categorization. On the other hand, a linear kernel cannot perform a mapping to a higher dimensionality space. Moreover, as [Hsu et al., 2003] argues, a linear kernel is actually a subcategory of an RBF kernel, for specific parameters. However, the same publication suggests that for a large number of attributes (compared to the number of instances) a linear kernel might perform better, because the data space has already many dimensions. In our project, we have 4 different feature sets with number of attributes being 500 for the first and 300 for the rest. Our database consists of 2093 training instances and only 350 and 759 validation and test instances respectively. Thus, the attribute number is large when compared to the number of instances. In fact, we believe that such a ratio suggests that both, a linear and a RBF kernel should be tested.

As for the two other kinds of kernels, (i.e sigmoid and polynomial), [Hsu et al., 2003] argues that a sigmoid kernel is proven to perform as good as an RBF one for some parameters and [Kahsay et al., 2005] tested various non-linear kernel choices for the specific task of visual object recognition (which is similar to ours) and found that

there exist no significant changes in the performance. However, another argument for not using a polynomial kernel is that it has more parameters.

The results from the experiments conducted in order to select the optimal kernel are presented in the following section (4.4).

4.4 Model selection

In this section we will describe the experimental procedure and the results obtained during the model selection process, i.e the process to define the kernel and the parameters to be used. Before we present the experimental results, it is worth mentioning that, since the 1-vs-1 approach is used for binarizing the problem, arguments given to the program as parameters are shared from all of the $C(C-1)/2$ classifications created.

Experiments for defining the kernel

In our experiments we tested a linear and an RBF kernel. Section 2.1.2 presents the forms of these kernels where it is obvious that the RBF one has a parameter γ. Moreover, one must also define the cost paramter c of equation (2.2). The experimental setup involves testing the SVM models with the following sets of parameters:

$$c \in \{2, 16, 64, 128, 1024, 4096\} \tag{4.7}$$

$$gamma \in \{0.0078125, 0.05, 0.125, 0.5, 2, 64\} \tag{4.8}$$

Note that the different parameters tried are in a logarithmic scale, since this level of precision is considered to be enough. The linear kernel model for a specific c_i was compared against the RBF kernel model with $c = c_i$ and the best gamma found for this c_i. The tests were performed on the validation set of a single feature-set (more specifically of the dSIFT500 - see table 3.1) and the evaluation measure used was Macro Average Precision (MAP) which is described later, in section 6.1. For now, it is enough to keep in mind that this measure is a summary of the performance over all classes.

As for the experimental results, we observed the following phenomenon: although **the RBF kernel resulted in better performance** (about 2-3% better than the linear one) for every c and for the best γ for this particular c, there existed some gamma values for which the performance was unacceptably low -even the $1/3$ of the performance obtained from the linear kernel. In contrast, none of the c values tried resulted in such an unacceptable performance for the linear kernel. In addition, [Hsu et al., 2003] presents

some example applications of their software and, indeed, some bad choices for the parameters of the RBF kernel result in accuracy even below 50%. Consequently, we decided to use the RBF kernel keeping in mind, though, that enough tests must be conducted in order to define good parameters, as this can determine dramatically the performance.

Experiments for defining optimal parameters

Since the RBF kernel was chosen, there are two parameters to define:

- *c*, which is the cost parameter of equation (2.2)
- *gamma*, which is the γ parameter appearing in the RBF kernel function described in 2.1.2

Although there is a theoretical explanation about how changing these values affect the model, the most common procedure followed in order to find the optimal parameters is to try as many as possible and compare the results. However, in this case we wish to define the best parameter setting for *each* of the feature-sets of table 3.1, because the final comparison (chapter 6) is performed for each one of them separately. Thus, for computational reasons we had to reduce the different combinations of c and gamma of formulas (4.7) and (4.8). In table 4.1 the different parameter combinations used for the experiment are presented. Smaller values of γ were generally found to be better in the experiments for kernel selection, described previously, and that is why we use this subset. In table 4.2 one can see the combination id's ordered according to the corresponding model's performance in the validation set of the dataset of column 1. Square brackets indicate equal performance.

Combination id	c	gamma
1	1	(1/#features)
2	2	0.0078125
3	1	0.002
4	16	0078125

Table 4.1: *The different parameter combinations tested for the SVM model.*

Dataset	Best combination (id) ordered
dSIFT500	2,[1,3],4
cSIFT300	2,4,1,3
dSIFT300	2,4,3,1
hSIFT300	2,1,4,3

Table 4.2: *The ordered id's of different combinations of c and γ parameters.*

Obviously, combination 2 (i.e c=2, γ = 0.0078125) is the best in all feature-sets. However, the performance for the rest of the combinations varies significantly.

Chapter 5

LDA based model

This chapter includes details about the design and implementation of our extended LDA model for image-caption pairs. Its structural parts form, in turn, a different kind of pairs: word-caption and visterm-image dyads. This kind of models is traditionally used for image annotation tasks and that's why we will refer to "word-caption" and "visterm-image" pairs, i.e a dataset which contains captions as a bag of words and a dataset which refers to images represented as a bag of visual words (visterms). In order to keep terminology uncluttered and underline the similarities with the basic LDA model (described in 2.2) we will maintain the terms "word" and "caption", however, it is important to keep in mind that since our model has been applied for an object categorization problem, one can interpret the term "word" as "class label" and the term "caption" as "label set".

5.1 Design

The design of our LDA-based model follows the same pattern used in most of the existing implementations described in section 2.4.1. Since the code for any of the LDA-based models for image annotation is not publicly available, we had to develop our own version and make our own assumptions and choices. Actually, our model is a slight modification (the implicit modifications made are mentioned later in the chapter) of the *GM-LDA* model, described in section 2.4.2. For these reasons, we will refer to our model as "the extended LDA model" or simply "our model" and not as the "GM-LDA model", without claiming in any way, though, that the basic idea of its structure is ours. Finally, in this chapter we will use the notation introduced in section 2.4.1.

5.1.1 Selecting a base model

To start with, the extended-LDA model is a generative process heavily based on the classical LDA-model, as described in the background section 2.2.3. Remember that LDA defines topics, i.e distributions over wordss. In our case, a document consists of two parts: its label and the vector of visual features. Thus, we define two kinds of such distributions: $\mathbf{z}$ for visual words and $\mathbf{u}$ for words (class labels). Our main purpose is to model the joint probability of the set of words $\mathbf{w}$, visterms $\mathbf{v}$, the distributions over them $\mathbf{u}$ and $\mathbf{z}$ respectively and the distribution over the latent factors θ, i.e $p(\mathbf{w},\mathbf{v},\mathbf{z},\mathbf{u},\theta)$. This can be achieved by applying twice the traditional LDA algorithm; once for visterms-image pairs and once for words-caption pairs, as described in 2.4.1. This is done with the order mentioned, because of the exchangeability issues also mentioned in the background section. In that way, the first sub-part of the generative process produces the distributions $p(image\_topic|image)$ and $p(visterm|image\_topic)$, as a normal application of LDA only on this part of the data would produce , and similarly the second sub-part produces the distributions $p(caption\_topic|caption)$ and $p(word|caption\_topic)$.

In order to learn the connection that exists between captions and images in the training set, we have to actually connect -in some way- the latent factors (topics) of images and captions. The reason for this is a direct consequence of using traditional LDA as a basis: the whole concept of topic models is to "transfer" the connection of terms and documents into the latent space. Consequently, in our extended version we have to transfer the connection of captions and images into this, new space. The existing ways of doing that have already been described in section 2.4.2. We chose to base our own version on the *GM-LDA* model for the reasons analyzed below.

Firstly, we had to reject the option of basing our model on the one proposed by [Barnard et al., 2003] because it was mainly developed and tested for a task different than object classification: it focuses on associating words with *particular regions* of an image, after first segmenting it. Moreover, they do not compare their results with other LDA-based models and their evaluation measures are difficult to compare with other studies. In contrast, [Blei and Jordan, 2003] present a detailed and comparative evaluation of their three models (also discussed briefly in section 2.4.2) and present results in terms of perplexity versus latent factor number, which provides more insight about the actual capabilities of the model in the image annotation task. In addition, they test their models in the *Corel* dataset for the particular task of image annotation.

Consequently, we found the idea to base our model on one of those mentioned in [Blei and Jordan, 2003] a reasonable one. The authors compare their models and reason about the results in detail. They prove that the *Corr-LDA model* is superior in all tests. However, we had to reject this option too, for two reasons: First, the publication describes the pure statistical concepts behind this technique and does not provide any information about how in step 2 the latent factor that generated the corresponding region is found during training. We find, thus, that although the statistical basis is well developed and easily understood, the implementation of this method is not so obvious. Second, as its creators proved, the method is (slightly) prone to overfitting and further techniques (such as *empirical Bayes*) are required for smoothing. More specifically, the authors have to assume a Dirichlet distribution on the word multinomial parameters β, i.e $\beta_i \sim Dir(\eta, \eta, ..., \eta)$ and use a *variational EM* procedure to learn the posterior distribution $p(\beta|D)$, where D represents the caption/image dataset. The need for all this process makes the model less generic and increases the effort needed for parameter optimization. Finally, it is the less generic method and it would be difficult to experiment with any variations of it. For this project we need a model which can be as generic as possible and easily tuned. In that way, we can generalize our results easier and, in addition, having less *hyperparameters* helps drawing more certain conclusions about the tuning of the regular parameters (which are the same for all models and can provide insights for the whole family of LDA-based models).

The need for smoothing is the reason why we rejected GM-Mixture as well. According to the results presented in [Blei and Jordan, 2003], GM-Mixture is much more prone to overfitting in comparison to Corr-LDA, and the need for smoothing is a major one. Without smoothing it performs worse that GM-LDA, and with smoothing it performs better in terms of perplexity, but in terms of precision-recall for text-based image retrieval (which is a task relevant to our own, i.e object categorization) it performs almost as well as GM-LDA. For these reasons, we chose to base our model on GM-LDA, although it has its own caveats as well (in fact, it is *oversmoothed*) but, at least, it is simple, avoids overfitting and for relatively small values of the topic-number parameter it has a reasonable performance.

GM-LDA gives, therefore, the solution to matching captions and images in the latent space. As already mentioned in section 2.4.2, this is achieved by sampling a random variable from a Dirichlet distribution and then generating both, image-regions and words with this variable held fixed. As for the conditional distributions, a multinomial one is a reasonable choice for $p(w|u, \beta)$, i.e the caption words conditioned on

the sampled topic and the hyperparameter, because words are discrete. In addition, a Gaussian distribution can be used for $p(v|z,\mu,\sigma)$, since visterms v are continuous. However, in order to demonstrate the close connection of LDA with its extended version, we chose to transform the continuous visual words into the discrete space and use a multinomial distribution for them too. In that way, given that LDA models the co-occurence of terms and documents via topics, it is obvious how the triplet: $< word-topic-document >$ of LDA model becomes $< caption-topic-image >$ in the extended version, with caption and images being, in turn, represented as $< term-topic-document >$ triplets as well.

There are two ways to build a dataset with image data represented as discrete features. Firstly, we can achieve that by creating a dataset with this property from the very beginning, i.e during feature extraction. More specifically, as mentioned in section 3.3, we can use an unnormalized count vector for the attribute-values. However, in our project we followed a different approach, which also allows to benefit from the information that normalization yields. Therefore, the following trick was used: the whole feature-set is initially scanned in order to find the lowest value v_{min} from all features. Then, in a second pass, each feature value v is replaced with the output of the function:

$$f(v) = \left\lceil \frac{v}{v_{min}} \right\rceil \tag{5.1}$$

In that way, the new smallest value in the dataset is 1 and the rest of the counts are scaled as appropriate. Unfortunately, we loose some of the original information enclosed in the continuous data but in a very small degree and this assumption is actually unimportant.

5.1.2 Mixture and generative modeling

Our extended model is, basically, a *mixture model* in the same way that LDA is one [Heinrich, 2004]. That is, data can be generated by one of different components which form a *convex combination*, i.e a sum where each term is one of the components weighted by a *mixture proportion*, and all proportions sum to unity. Indeed, in the extended LDA model a term t (i.e either a caption word or a visual-word) is a mixture of multinomial distributions $p(t|z)$ weighted by the topic probabilities $p(z)$, since

given a specific topic $\tilde{z}$ each term instance $\tilde{t}$ is likely to be generated with probability:

$$p(\tilde{t}) = \sum_{z} p(\tilde{t}|\tilde{z})p(\tilde{z}) \quad , \quad \sum_{z} p(\tilde{z}) = 1 \tag{5.2}$$

Obviously, the term-topic distributions are the mixture components and the probabilities $p(z)$ are the mixture proportions that weight each component. [Heinrich, 2004] refers to LDA as an *admixture model* because one step further is actually added in the formulation (5.2), i.e topic instances $\tilde{z}$ are in fact conditioned on specific document instances $\tilde{d}$ (because documents are, in turn, viewed as a mixture of various topics). So, LDA is a mixture model whose components are, in turn, mixture models of different variables, a property that increases its flexibility. Consequently, we can think of our extended LDA model as a structure which connects two different mixture models. Being able to represent a method as a mixture model is convenient, because standard methodologies exist for estimating their parameters (for example EM).

The extended LDA model can also be viewed as a *generative model*, i.e a model which is able to generate data given some latent parameters by defining the joint distribution of data-variables and latent factors. Given this joint distribution if we have observed variables we can actually reverse the process and perform inference by using Bayes rule to form conditional probabilities or by integrating out the unknowns. For example, as it will be described later in this chapter, our algorithm models the joint distribution of observable and latent variables $p(\mathbf{w}, \mathbf{v}, \mathbf{z}, \mathbf{u}, \theta, \phi^{word}, \phi^{vis}|\alpha, \beta)$, so it is a two-level generative process, as it merges the joint distributions $p(\mathbf{w}, \mathbf{u}, \theta, \phi^{word}|\alpha, \beta)$ and $p(\mathbf{v}, \mathbf{z}, \theta, \phi^{vis}|\alpha, \beta)$, defined in the two major steps of the whole process, into a single joint probability distribution. Intuitively, the interpretation of our generative process is the following: each document (image/caption pair) is generated by choosing a distribution over topics. Then, given this distribution, the algorithm generates the specific topic for each visual word and based on this, visual words are generated. The same process is followed for words, with the parameter that generated the topic distribution held fixed.

Given all the above, we have the following algorithm (generative process) for our extended LDA-based model:

Algorithm

For each topic k:

$\phi_k^{word} \sim Dir(\beta)$

$\phi_k^{vis} \sim Dir(\beta)$

For each document d:

1) $\theta \sim Dir(\alpha)$

2) For each of the N visual words:

a) Sample topic $z_n \sim Mult(\theta)$

b) Sample visual word v_n from $p(v_n|z_n,\beta)$:

$v_n \sim Mult(\phi_{z_m})$

3) For each of the M caption words:

a)Sample topic $u_m \sim Mult(\theta)$

b)Sample word w_m from $p(w_m|u_m,\beta)$:

$w_m \sim Mult(\phi_{u_m})$

Parameters	
α,β	given hyperparameters
M	# of all caption words
N	# of all visual words
K	# of topics
D	# of documents (image-caption pairs)

In addition to the parameters stated above, we also have: V_M and V_N which are the number of all different words and visual words respectively, i.e the words' vocabulary size and the visual-words' vocabulary size.

The above process results in the following joint probability distribution of words, visual words and latent variables:

$$\begin{aligned} p\left(\mathbf{w},\mathbf{v},\mathbf{z},\mathbf{u},\theta,\phi^{word},\phi^{vis}|\alpha,\beta\right) &= p(\theta|\alpha)p(\phi^{word}|\beta)p(\phi^{vis}|\beta) \\ &\cdot\left(\prod_{n=1}^{N} p(z_n|\theta)p(v_n|z_n,\beta)\right) \\ &\cdot\left(\prod_{m=1}^{M} p(u_m|\theta)p(w_m|u_m,\beta)\right) \end{aligned} \tag{5.3}$$

The plate notation of the model is depicted in figure 5.1

This figure can be interpreted as follows: circles denote variables and arrows denote the dependency between them. Shaded variables are the observed ones. Plates denote iterations. For example, the bigger outer plate denotes an iteration for each of the D documents, and in each loop all of the iterations defined by the two inner plates are

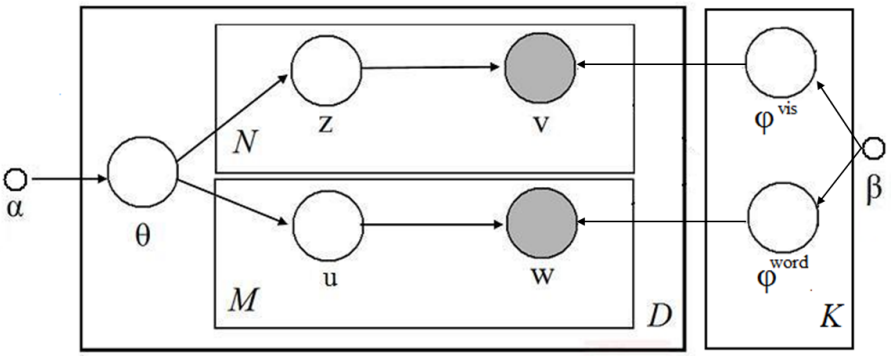


Figure 5.1: *The LDA model of images and captions.*

executed.

5.1.3 Details and assumptions

We will explain in more detail the above algorithm, describing the usage and meaning of each variable as well. Firstly, the hyperparameters α and β are chosen independently and define the nature of the priors on the per-document topic distribution and the per-topic term distribution respectively. In contrast to [Blei et al., 2003] where β is assumed to be a $K \times V$ matrix (where V is the vocabulary size) and α is a K-dimensional vector, we chose to follow the approach described in [Griffiths and Steyvers, 2004] where *symmetric* Dirichlet priors are used and α, β are treated as single values. This is convenient for our extension because we can use the same β for both sampling steps. Choosing a symmetric prior is also a typical approach ([Steyvers and Griffiths, 2006]) and has to do with our *prior* beliefs about the topic-document and word-topic distributions. More specifically, we assume that all words are equally probable of being assigned to a topic and all topics are equally probable of being assigned to a document. This is clearly not the case or real datasets, because not all words are equally popular. However, if we don't want to embed any prior knowledge about their frequencies then the symmetric priors is a reasonable choice. Also, the reason why the same β is used to generate both ϕ's is explained in section 5.3.2.

The parameters ϕ and θ define the distributions over terms and over topics respectively. We use the notation ϕ^{word} and ϕ^{vis} to distinguish between the one over words and the one over visual words. More specifically, ϕ is a matrix where $\phi^{word}_{m,k} = p(w_m|u_k)$ and $\phi^{vis}_{n,k} = p(v_n|z_k)$ and θ is also a matrix where $\theta_{d,k} = p(topic = k|document = d)$.

The process then generates each caption-image pair in two steps: firstly it generates all N visual words and then it generates all M words for captions. Another assumption

that we made in the design of the generative model is that N and M are known. This does not affect the rest of the procedure because, as [Blei et al., 2003] notices for the case of LDA, these variables are independent from the latent ones, which are responsible for generating data. [Blei et al., 2003], thus, chooses a Poisson distribution for the word-number of the LDA but argues that this is not critical and that other representations could be used as well. Finally, the parts where terms are generated involve sampling the topics and the terms from the appropriate Multinomial distributions. The reason why the Dirichlet distributions were chosen for θ and ϕ to be drawn, is that a Dirichlet is conjugate to the Multinomial, a property which can be used for more convenient further computations, described later in this chapter.

5.2 Implementation

Our aim is to train a model which, given an **unseen** image $\mathbf{v}$, is able to predict relevant words. An image is represented as $\mathbf{v}$, i.e the set of all of the visual words that form it. In order to achieve prediction, we have to compute:

$$p(w = m|\mathbf{v}) = \sum_k p(z = k|\mathbf{v})p(w = m|u = k) \tag{5.4}$$

In other words, we marginalize out the latent variable from a product of factors which the model is supposed to compute. Using the notation of ϕ and θ for term-topic and topic-document distributions respectively (and using superscripts to denote if they correspond to the word-caption or to the visterm-image pair), the factors of the summation can be written as $\hat{\theta}_k^{vis}$ and $\phi_{k,m}^{word}$ respectively, in order of appearance. Note that we distinguish θ^{vis} from θ^{word}, because as it will be explained later in this chapter, our algorithm uses separate representations for them which are finally tied in a way that will be also described later. Note also that in this case θ has a single index k, because we are now talking about a single testing document. Finally, the symbol "hat" is only used in θ, in order to denote that this variable must be inferred from **unseen** data, while for ϕ, we use the one which was estimated from the learning process. The connection between the topics u and z that we provided (in the rest of the chapter we describe how we imposed such a connection) allows us to use the same index -in this case k-, whenever we want to retrieve this relation. Remember from section 5.1.1 that in our case this connection is a very loose one, whereas for GM-Mixture, for example, it is so strong (actually they use the exact same topic) that the model overfits the training data in an unacceptable degree.

5.2.1 Finding the appropriate distributions

The problem now is how to compute $\hat{\theta}$ and ϕ, or in general θ, ϕ, $\hat{\theta}$ and $\hat{\phi}$. There are many approaches to this problem. Most of them, treat θ and ϕ as parameters that need to be estimated. In contrast, [Griffiths and Steyvers, 2004] prove that they can be acquired by calculating and then sampling from the appropriate posterior distributions of latent variables given observed data. For example, for the case of simple LDA the posterior $p(\mathbf{z}|\mathbf{w},\alpha,\beta)$ needs to be found. [Blei et al., 2003] claim that finding this distribution is the most common goal for most practical applications as well. In our case, for the extended LDA model we need to calculate the posterior distributions $p(\mathbf{u}|\mathbf{w},\alpha,\beta)$ and $p(\mathbf{z}|\mathbf{v},\alpha,\beta)$ Normally these can be computed from the full joint distribution that our model finds, but since these two parts are independent given α and β the computation can be split and each distribution can be calculated independently. We will describe below how the computation is done for the first of these and the exact same steps can be followed for the second one.

In order to compute the probability distribution:

$$p(\mathbf{u}|\mathbf{w},\alpha,\beta) = \frac{p(\mathbf{w},\mathbf{u})}{\sum_u p(\mathbf{w},\mathbf{u})} \tag{5.5}$$

we can follow the steps described in [Heinrich, 2004]. Only the most important steps will be mentioned here. Firstly, we can write that

$$p(\mathbf{w},\mathbf{u}|\alpha,\beta) = p(\mathbf{w}|\mathbf{u},\beta)p(\mathbf{u}|\alpha) \tag{5.6}$$

because $\mathbf{w}$ is independent of α and $\mathbf{u}$ of β, and since the conditioning sets are different each factor can be computed separately. Although the generative process is a way of formulating the algorithm, in practice we have observed documents and, thus, observed word counts on the associated topics (this is actually ϕ) and similarly for θ. We will discuss later how ϕ and θ can be acquired by an iterative, EM-like process only by using count variables. So, the factors of equation (5.6) can be thought as having ϕ and θ respectively in the observed set.

We will now compute each of the factors of (5.6) separately. For the first, it can be easily shown that it is equal to the observed word counts on the associated topics. That is:

$$p(\mathbf{w}|\mathbf{u},\phi) = \prod_{k=1}^{K}\prod_{t=1}^{V_M}\phi_{k,t}^{n_k^{(t)}} \tag{5.7}$$

where $n_k^{(t)}$ is the number of times that term t has been observed with topic k. We can

now integrate over ϕ and acquire $p(\mathbf{w}|\mathbf{u},\beta)$. The integral can be computed as shown in [Heinrich, 2004] or in [Griffiths and Steyvers, 2004] and the result is:

$$p(\mathbf{w}|\mathbf{u}) = \left(\frac{\Gamma(V_M\beta)}{\Gamma(\beta)^{V_M}}\right)^K \cdot \prod_{k=1}^{K} \frac{\prod_w \Gamma\left(n_k^{(w)} + \beta\right)}{\Gamma\left(\sum_w n_k^{(w)} + V_M\beta\right)} \tag{5.8}$$

where $\Gamma(\cdot)$ is the standard Gamma function, V_M is the number of distinct words (the vocabulary for caption-words), K is the total number of topics and $n_k^{(w)}$ is the number of times word w was observed with topic k in the vector $\mathbf{u}$. In terms of implementation, this counter is a $K \times V_M$ matrix.

For the second factor of (5.6), $p(\mathbf{u}|\theta)$, we can work in a similar way by noticing that:

$$p(\mathbf{u}|\theta) = \prod_{d=1}^{D}\prod_{k=1}^{K} \theta_{d,k}^{n_d^{(k)}} \tag{5.9}$$

with D being the total number of documents and $n_d^{(k)}$ denoting the number of times that topic k has been observed with document d, i.e with *any* word of document d. In terms of implementation, this counter is a $D \times K$ matrix. Again, we compute the integral over θ and the result is proven to be: [Griffiths and Steyvers, 2004]

$$p(\mathbf{u}) = \left(\frac{\Gamma(K\alpha)}{\Gamma(\alpha)^{K}}\right)^D \cdot \prod_{d=1}^{D} \frac{\prod_k \Gamma\left(n_k^{(d)} + \alpha\right)}{\Gamma\left(\sum_k n_k^{(d)} + K\alpha\right)} \tag{5.10}$$

where $n_k^{(d)}$ is the number of times that any word from document d was observed with topic k.

The process described above (which can be found in more detail in [Heinrich, 2004] or [Griffiths and Steyvers, 2004]) results in obtaining a solution for $p(\mathbf{w},\mathbf{u})$ which can be replaced now in equation (5.5). Unfortunately the computation of the resulting fraction is intractable using traditional algebraic methods. One way to find a good approximate solution is to use *Gibbs sampling* as [Griffiths and Steyvers, 2004] proposed. This publication, as well as [Heinrich, 2004] in more detail, describes how the introduced formulation which involves counts can be used in order to find the **update equation** from which topics are sampled using the Gibbs sampler. As [Griffiths and Steyvers, 2004] showed, this equation is the following:

$$p(u_i = k|\mathbf{u}_{-i}, w_i, d_i) \propto \frac{n_k^{(w_i)} + \beta}{\sum_w n_k^{(w)} + V_M\beta} \cdot \frac{n_k^{(d_i)} + \alpha}{\sum_k n_k^{(d_i)} + K\alpha} \tag{5.11}$$

The subscript -i indicates all instances but i. $n_k^{(w_i)}$ and $n_k^{(d_i)}$ are the counters as described in equations (5.8) and (5.10). Note that actually the counters n should be n_{-i} to indicate that the count does not include the current assignment of u_i, however we omitted this to keep the notation uncluttered. As [Steyvers and Griffiths, 2006] describes, the first factor is basically the probability of word w under topic k whereas the second one is the probability of topic k under document d. So, the equation can be written as:

$$p(u_i = k|\mathbf{u}_{-i}, w_i, d_i) \propto p(w_i|u_i = k) \cdot p(u_i = k|d = i) \tag{5.12}$$

This interpretation is very useful when we attempt to connect the topics for words and for visterms by holding fixed the paramter of the Multinomial from which they are drawn. In the rest of the chapter we will present more details about this issue. For now, returning to our initial notation, we can symbolize the first distribution of the RHS as $\phi_{k,i}^{word}$ and the second one as $\theta_{i,k}^{word}$, where "word" (or "vis") is added as superscript in order to make the distinction between counters for word-caption and for visterm-image pairs.

Analyzing equation (5.12) reveals **the factors influencing the way in which topics are assigned to specific words**. If many instances of a word w_i existing in the vocabulary have already been associated with a particular topic k, then $p(w_i|u_i = k)$ in the RHS will be large and the quantity in LHS (the probability of assigning again topic k to w_i found in document d_i) will also become larger. Moreover, if the same topic k has been frequently used in a particular document d_i, then $p(u_i = k|d = i)$ will be larger and the probability that any word in document d_i is assigned to topic k becomes greater.

By following the exact same procedure, we can find the posterior distribution for visterm - image pairs, i.e $p(z_i = k|\mathbf{z}_{-i}, v_i, d_i)$. All we have to do is to replace u with z and w with v in the above equations. So, in terms of implementation we will also have the distributions ϕ^{vis} and θ^{vis}. Hence, the corresponding equation, in the case of visterms - image pairs, is:

$$p(z_i = k|\mathbf{z}_{-i}, v_i, d_i) \propto \frac{s_k^{(v_i)} + \beta}{\sum_v s_k^{(v)} + V_N\beta} \cdot \frac{s_k^{(d_i)} + \alpha}{\sum_k s_k^{(d_i)} + K\alpha} \tag{5.13}$$

Here, to avoid confusion we denote the count variables as s instead of n, to underline that different count matrices are stored for each word-caption and visterm-image pairs.

What we have shown actually, is how to compute the update equations for Gibbs sampler which can be used in order to find the final approximations of the posteriors.

Gibbs sampling is a *Markov Chain Monte Carlo (MCMC)* procedure used in order to produce random samples (in fact, *pseudo-random* as [Bishop, 2006] underlines, as they have to pass some tests of "randomness") from an unknown target distribution. In general, all MCMC procedures first initialize the Markov chain to some state and then the chain is run obtaining a new state in each iteration, until a stop criterion is reached, in our case convergence (i.e the chain converges to the target distribution). The transition between states is algorithm specific.

For *Gibbs sampling* we need an update equation which defines this. Suppose for example that we want to approximate the distribution p($\mathbf{u}$), where $\mathbf{u} = (u_1, u_2, ...u_k)$. After initializing the chain, each Gibbs sampling iteration will update a single variable u_i assigning to it a new value sampled from the distribution of u_i conditioned on the values of the rest variables $u_j, \forall j \neq i$. So, in each step, u_i is replaced by a value $\hat{u}_i$ sampled from $p(u_i|\mathbf{u}_{-i})$. This procedure can be followed for every variable and for a fixed number of iterations or until convergence. In our case that we have observed data $\mathbf{w}$ (i.e a document), the sampling for a specific variable is done by the distribution of equation (5.11) (and the corresponding distribution for visterm-image pairs, i.e (5.13)).

Finally, we also have to determine the way in which the parameter θ of the generative process described in section 5.1.2 is held fixed when generating topics for both, words and visterms. Our approach is based on the intuitive explanation of the distributions involved in (5.12). Since the second one is the the probability of topic k under document d and since we assume that visual words are generated first and then caption-words, we can use the probability learned first for both cases. More specifically, given equations (5.11) and (5.13), we change the first one according to the second, and we obtain the final update equation for caption words:

$$p(u_i = k|\mathbf{u}_{-i}, w_i, d_i) \propto \frac{n_k^{(w_i)} + \beta}{\sum_w n_k^{(w)} + V_M\beta} \cdot \frac{s_k^{(d_i)} + \alpha}{\sum_k s_k^{(d_i)} + K\alpha} \tag{5.14}$$

To sum up, equations (5.13) and (5.14) are the final update equations for Gibbs sampler used to sample topics for visual and for caption words respectively.

5.2.2 Inference for previously unseen data

Although ϕ and θ are integrated out in the learning step, there is a way of obtaining their distributions based on the same statistics of the dataset. As [Heinrich, 2004] explains, this corresponds to finding the multinomial parameter sets that correspond to the state of the Markov chain $M = [\mathbf{w}, \mathbf{z}]$. Notice that in this case the observed

variables **w** come from the (previously) unseen data. However, we assume that unseen data come from the same vocabulary as the training data, so the pair $[\mathbf{w}, \mathbf{z}]$ is already "learned". [Griffiths and Steyvers, 2004] shows that the distributions of θ and ϕ for unseen data, symbolized as $\hat{\theta}$ and $\hat{\phi}$, can be obtained as follows:

$$\hat{\phi}_{k,i}^{word} = \frac{n_k^{(w_i)} + \beta}{\sum_w n_k^{(w)} + V_M \beta}, \hat{\theta}_{d,k}^{word} = \frac{n_k^{(d_i)} + \alpha}{\sum_k n_k^{(d_i)} + K\alpha} \tag{5.15}$$

where the counters $n^{(w)}$ and $n^{(d)}$ are like ones defined in equations (5.11). The similarity with (5.11) is obvious, but for this last case, the computation is done for new instances of word w. This is feasible because we assume that unseen data are composed from instances of words already included in our vocabulary. More specifically, the first fraction is the predictive distribution of sampling a new instance of w_i from topic u_k and the second fraction is the predictive distribution of sampling a new word in d_i from topic u_i.

Again, we note that in a similar way $\hat{\theta}^{vis}$ and $\hat{\phi}^{vis}$ can be obtained. To sum up, for new data we have the estimates of table 5.1.

$\hat{\theta}_{d,k}^{vis}$	represents	$p(topic = k \mid image = d)$
$\hat{\theta}_{d,k}^{word}$	represents	$p(topic = k \mid caption = d)$
$\hat{\phi}_{k,v}^{vis}$	represents	$p(visual\ feature = v \mid topic = k)$
$\hat{\phi}_{k,w}^{word}$	represents	$p(caption\_word = w \mid topic = k)$

Table 5.1: *The distributions obtained be applying the extended LDA model for estimation.*

We also underline that θ's are created to be tightly related to each other and, so, this agrees with our generative process. We can now return to our original problem of how to emit the most probable caption-words given a set of visual words. The solution is the equation (5.4) and as was demonstrated, the RHS can be computed by replacing the first distribution with the $\hat{\theta}_k^{vis}$ obtained from doing inference on the unseen data **v**, and the second one with the $\phi_{k,m}^{word}$ learned during training for the vocabulary of caption-words. We then marginalize over all topics and get the emission probability for each word in our vocabulary. This kind of output gives us some freedom as to how to use it. One approach is to emit the l most probable words. Another idea is to emit word w_i if $p(w_i|\mathbf{v}) \geq t$, with t being some threshold. However, for object categorization purposes we can benefit from the fact that the output contains probabilities in order to apply measures such as recall and precision, as described in a later chapter.

5.2.3 Notes on the programming code

For the implementation of Gibbs sampler for LDA, we used (and extended) the package *GibbsLDA++* [1], developed for the purposes of [Phan et al., 2008]. The provided code is an implementation of the simple LDA model and uses Gibbs sampling for parameter estimation and inference, taking advantage of the theoretical results already described in this chapter. The fact that equations in the form of (5.13) are used for the Gibbs sampling, where the only information required are counts of the number of times a word is observed with a topic and of the number of times a topic is seen in a document, makes the implementation very efficient.

In more detail, the code requires the following input arguments:

- The hyperparameter alpha (α)
- The hyperparameter beta (β)
- The total number of iterations to run the Markov Chain
- The total number of topics
- The input file represented as a bag-of-words. The program assumes that each line is a document represented as an arbitrary number of words, separated by white characters.

For each corpus (which is represented as a Java class), the program defines some internal fields: Firstly, the distributions ϕ and θ are internally represented as a matrix "phi" of dimensions $K \times V_M$, and a matrix "theta" of dimensions $D \times K$ respectively. In order for these to be computed, as we already described, some counters are needed. These (i.e $n^{(w)}$ and $n^{(d)}$ from equation (5.11)) are also represented as matrices *NW* and *ND*. The chain is represented as a vector Z for each document in the corpus, where $Z_{d,i} = k$ means that the i-th word of document d has been assigned topic k.

Given the above, the program firstly initializes the Markov chain Z with random topics. Then, for each word, the count matrices NW and ND are firstly decremented by one for index k, where k is the topic which is currently assigned to them in vector Z. Then, equation (5.11) is used in order to sample a new topic $\acute{k}$ according to the counter's values and, finally, the count matrices are incremented by one for index $\acute{k}$. A full iteration involves the above procedure for all of the V_M vocabulary's words. Obviously, in each iteration the new topic is selected based on the current values of the count matrices which, in turn, are incremented based on the newly selected topic. The concept behind this procedure is similar to the EM approach.

[1] available at: http://gibbslda.sourceforge.net/

The outputs of the program are the following:
For training data (estimation):

- A file which contains ϕ, the word-topic distributions, i.e $p(word|topic)$
- A file which contains the topic-document distributions θ, i.e $p(topic|document)$
- A file containing the final topic assignments for each word in the vocabulary.
- A file which contains the most T likely words of each topic.

For inference on unseen data similar outputs are obtained but are, of course, computed on the unseen data. It is worth mentioning that the output containing the most likely words by topic can be useful for understanding the nature of the corpus.

In this project, we had to modify the code described above for our own task which involves working with collections of words (captions) as well as visual words (images). The modifications made reflect exactly the ones described in the theoretical part of this chapter. More specifically, we model captions and images separately and we treat both as separate corpora. Hence, the process described above for a single corpus was replicated twice. The connection between the topics for images and for captions was provided exactly as demonstrated in section 5.2.1. Moreover, some code had to be added for calculating the probabilities of the word emissions (equation (5.4)) as well as for calculating some evaluation measures, as will be explained in the next chapter. Finally, the program outputs were augmented to include the information for both corpora, the probabilities of each word to be emitted given a particular image and the evaluation measures.

5.3 Parameters of the model

Our program for implementing the extended LDA-based model, needs the following parameters to be defined (apart from the arguments concerning functionality issues):

1. The hyperparameter alpha (α)
2. The hyperparameter beta (β)
3. The number of topics K
4. The number of Gibbs sampling iterations L
5. The threshold t used for word-emission (for each unseen image, words with probability larger than t will be reported as being associated to it)

The first three are the actual model parameters, as they also appear in the generative process defined in section 5.1.2. The rest can be considered as "external" parameters. L regards our specific implementation which uses Gibbs sampling. Moreover, the threshold t can be omitted if the full file containing the probabilities for each word is the desired type of output. There are various ways of finding the optimal parameters for a model. However, before analyzing this issue, it is vital to understand the role and importance of each of these parameters independently.

5.3.1 The role of each parameter

Larger values for **hyperparameters** α and β means larger smoothing on the multinomial parameters θ and ϕ respectively. In turn, this results in less certain assignments of topics to documents and of terms to topics respectively. In detail, since θ represents $p(topic|document)$ and ϕ represents $p(word|topic)$, the probability masses of these distributions will be more equally distributed over each specific topic-document and word-topic pair and the corresponding matrices will be less sparse. This effect can be intuitively explained as imposing a bias such that the model tends to assign more terms to each topic, leading in topics which contain terms less "similar" to each other. Obviously the effect that hyperparameters have on the model is heavily dependent on the number of topics. For example, having a small K would (theoretically) require some smoothing (larger values for the hyperparameters) because if many terms have to be distributed among too few topics it is reasonable that the "similarity" criterion for assigning two terms to the same topic should be less strict. Of course, this explanation is intuitive and does not mean that a the topic number K and the hyperparameters can be modeled easily as a function of each other. On the contrary, this relation is also dependent on the specific nature of the dataset, as different kinds of topics are learned for different kinds of documents.

The **topic number K** is also critical for the model's performance. It is not obvious how to find an optimal K, as it depends on the dataset's nature, something that cannot be quantified. For example, a corpus of scientific publications is expected to need a smaller K than a corpus of a newspaper archive. In our case where we perform object categorization with our extended LDA model, a large K will result in sparser matrices for ϕ and θ, which means that it would be difficult for co-occurrences of class-labels with visual words to be captured. On the other hand, having too few topics will make the algorithm less discriminative, as it will have to assign less similar terms to the same

topic.

5.3.2 Parameter Estimation

5.3.2.1 Experimental Procedure

The analysis of each parameter's role suggests that a reasonable **approach to estimate the model** is by searching for the optimal α and β *in combination with K*. This can be achieved in two ways: firstly, by choosing fixed α and β and learn the *optimal* K from the dataset. Hyperparameters can be set to specific values by relying on previous research or heuristically. For example, [Griffiths and Steyvers, 2004] showed that $\alpha = 50/K$ and $\beta = 0.01$ is proven heuristically to be a good combination in general. Secondly, one can try many different combinations of α, β and K together, so that hyperparameters can also be learned from the data. There is a trade-off though, because in that case it is not the optimal K that is learned, but just the best from the ones tried. These methods are explained in detail separately in the next two paragraphs.

The first method is implemented using Bayesian methods to *learn* K from data, a process followed in [Griffiths and Steyvers, 2004]. Such standard methods are able to select the model M (which is basically represented as a set of parameters) with the highest posterior $p(M|Data)$ instead of basing the selection on the likelihood. Of course, this requires finding the likelihood $p(Data|M)$ by integrating out over all possible parameters. In our case, for example, for the part of the model concerning words we would have to compute $p(\mathbf{w}|K)$, if α and β are given. The integral is intractable and sampling techniques (taking advantage of the already computed formula of equation (5.8)) can be used. However, this approach requires α and β to be given and so, they must be found heuristically. Of course one could repeat the process for different combinations of α and β, but this would be more time consuming.

The second method, is more interesting because it involves inferring from data not only K, but also α and β. It is generally true that different kinds of corpora need different calibrations regarding topic issues. For example, a dataset of images (represented as bag of visual words) depicting a limited number of objects without background is very different in nature than a dataset of natural scene images. Thus, tuning the model in a more data-driven way is desirable. Basing the hyperparameter optimization on data is also useful because since α and β which are found to be good for a specific dataset, they can provide useful insight into its properties in the latent space. For example, the value of α is a measure of similarity between different documents in the

topic space, and the value for β measures the size of the sets composed of co-occurring terms in the documents [Heinrich, 2004].

Although using Bayesian parameter estimation would potentially result in better parameters to be found, we would have to use fixed values for α and β, as we would not be able to repeat the process many times for different hyperparameters, due to limited computational resources and lack of time. However, we considered that it is vital to learn the hyperparameters in a more data-driven way. The project's results are tested in four different feature sets (table 3.1) and we believe that it is interesting to estimate the hyperparameters from the data and investigate how these vary for different feature sets of the same database. So, instead of choosing the optimal K for given hyperparameters, we preferred to experiment with different combinations of hyperparameters *and* K. This is a trade-off: in order to estimate the hyperparameters from data, we have to try less values for K. However, in order to search for as many combinations as possible (given our time available and computational resources), we use the Gibbs sampling iterations parameter L and our datasets in a clever way, as it will be described shortly.

Also, some ad-hoc tests made with a very small number of Gibbs sampling iterations revealed that there is not significant change in the performance for α values different than $50/K$, a value found heuristically from [Griffiths and Steyvers, 2004]. This fact as well as that this heuristic rule is a function of K are the reasons why we used the setting $\alpha = 50/K$ for all of our experiments, although it may not be the optimal choice. So, β and K are the only parameters to be estimated. [Philbin et al., 2008] follows an approach similar to ours: they find α, β and K as a combination by using non-Bayesian methods, using heuristic methods. Consequently, our experimental set-up is sketched as follows:

1. Come up with a set of parameters β and K to be tested as combinations. We denote with $\vec{\beta}$ and $\vec{K}$ the corresponding sets and explain shortly how they are chosen.
2. For every different combination of topic number $k \in \vec{K}$ and $\beta \in \vec{\beta}$:
   (a) Apply the model with parameters k and β on the validation set of *one* of the four available feature-sets. We chose dSIFT500 for this purpose, because from informal experiments was proved to be the best dataset.
3. Create a matrix associating each $k \in \vec{K}$ with its best $\tilde{\beta}_k$ found in step 2a.
4. For every combination of feature-set and $k \in \vec{K}$:

   (a) Apply the model with parameters k and $\tilde{\beta}_k$ on the corresponding validation sets

5. Create a matrix associating each feature-set f with its corresponding best $\tilde{k}_f$ (and consequently with its corresponding best β), based on the results of step 4a.
6. For each feature-set f
   (a) Apply the model with parameters $\tilde{k}_f$ and $\tilde{\beta}_k$ on the test set created for f and get the final results.

Note that different training, validation and test sets exist for each feature-set. Also, the evaluation of the models was done using the *Mean Average Precision (MAP)* measure described in section 6.1 and more specifically in equations (6.5) and (6.6). However, the details are not critical at this point, and it is enough to keep in mind that this measure summarizes the performance over all classes.

Breaking the process in subtasks allows to vary the **number of Gibbs sampling iterations** dependently on the significance of each phase. This is necessary in order to gain in computational time (note that for larger K -and of course for larger iteration number L- this time increases) because the models are trained and evaluated 24 times for step 2 (4 different values for β $\times$ 6 different values for K) and 24 times for step 4 (4 different feature-sets $\times$ 6 different values for K). This results in 48 $\times$ 2 $\times$ L Gibbs sampling iterations (as one series of iterations is needed for training and one for inference). For example, in step 2 of the process described above we use L = 15 iterations for the training phase and the same for the inference one. For step 4, where the topic-number is selected (and is a more important parameter), we increase the number of iterations to 20. The L parameter is small for phases 2 and 4 because we are only interested in comparing the results, in contrast to the testing phase where we actually want to achieve as good results as possible.

Moreover, these numbers are selected based on some of our ad-hoc experiments, which define the *"burnin" period* for Gibbs sampler to be less than 15 for the particular datasets that we have. With the term "burnin" we refer to the initial period where the samples obtained are poor estimates of the target distribution. However, for the testing phase of step 6, we want to define a number of iterations as close to convergence as possible. Again, based on some of our ad-hoc experiments we found that for our particular datasets, increasing the number of iterations to more than 150 for training and to more than 200 for inference does not yield significant change. However, as our

final results are very important for drawing conclusions, we selected 500 training and 1100 inference iterations in order to be somehow far from this "safety" limit.

Finally, the procedure described above needs the testing set of β's ($\vec{\beta}$) as well as the testing set of k's ($\vec{K}$) to be defined. In our project we formed these sets based on existing publications for similar tasks, and then we further reduced them based on the results of more ad-hoc experiments (that is, with a small L). More specifically, $\vec{\beta}$ was formed by selecting some values around 0.01 which is the optimal value found by [Griffiths and Steyvers, 2004]. We noticed that we had better performance for larger values, so our final set contains:

$$\tilde{\beta} = \{0.04, 0.07, 0.1, 0.16\} \tag{5.16}$$

Selecting the set of candidate topic numbers K is a slightly more complicated issue. We had to take into account the theoretical considerations for the role of this parameter (subsection 5.3.1), the findings from published results and the trade-off between a big K and a smaller number of Gibbs sampling iterations L, because more topics add more delay to each iteration. Finally, as [Blei and Jordan, 2003] proved, GM-LDA which forms the basis of our model tends to overfit training data for large K, while for smaller values gives reasonable results. This can be explained as follows: the fact that GM-LDA provides such a loose connection between word/topic and visterm/topic pairs, increases the probability that a word is assigned to topics which did not contribute at all into generating the corresponding image. For a large K, this probability increases.

To start with, [Blei and Jordan, 2003] evaluate how well their image annotation models fit the data by measuring *negative log probability* and perplexity versus the topic number K. Better models have lower negative log probability, meaning that they assign higher likelihood to the test set. As for perplexity, lower numbers are again more desirable. As can be seen in figure 5.2, values of K less than 50 yield better results.

For perplexity, the conclusion is the same i.e smaller values for K are better. For this reason, we tested different K in the range of [1...100]. More specifically, the set of different K's to be tested is:

$$\tilde{K} = \{5, 15, 20, 25, 35, 100\} \tag{5.17}$$

Finally, during the experiments for selecting K, if the evaluation output for two different numbers of topic K_1 and K_2, with $K_1 \leq K_2$ were very similar, we selected K_1, i.e we imposed a slight bias towards smaller K for the reasons stated above as well as for computational efficiecny purposes.

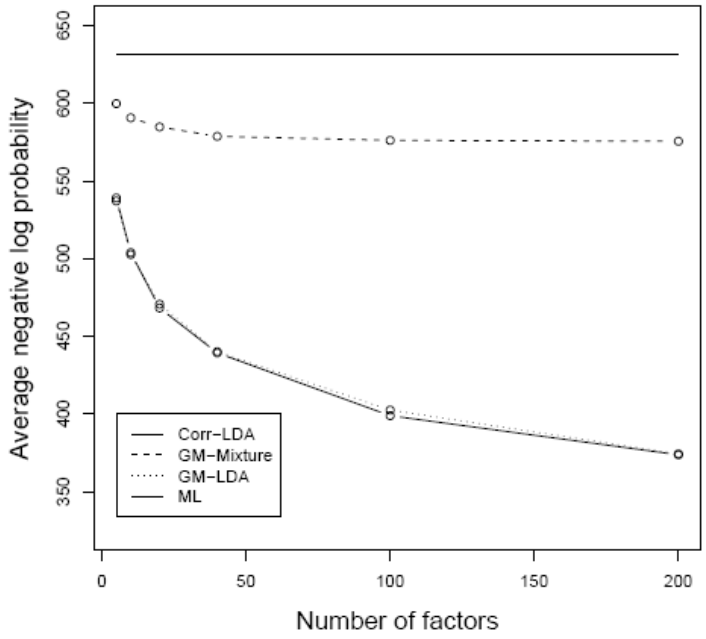


Figure 5.2: *The per-image average negative log probability for the validation set as a function of the total number of topics K. Obviously, smaller K is better.source: [Blei and Jordan, 2003]*

5.3.2.2 Assumptions

This experimental procedure involves some **assumptions**. Firstly, for discovering the optimal $\tilde{\beta}_k$ for each topic k, we used only one feature-set, the *dSIFT500*. Since all feature-sets come from the same database, and given that k and β are associated in a way that the nature of the database defines, we assumed that the same best β's would be discovered for the corresponding topics for any feature-set. Some informal experiments confirm this.

Moreover, (as it is obvious from figure 5.1) we use the same β for both parts of the generative process: the one generating captions and the one generating images. The reason for this assumption is that, following the theoretical discussion of section 5.3.1, β is mainly related to K and to the dataset's nature. Since K is fixed for word-caption and visterm-image pairs, we just have to consider the datasets' nature, which we assume that is similar for both cases: captions and images contain the same number of instances; moreover, we assume that captions and images are somehow two different representations of the same document, meaning that we treat images as collections of depicted objects and anything else is just noise, while captions are a pure representation of the image, containing only the class-labels of the corresponding objects. Of course, in contrast to the caption data, multiple topics might be extracted from a visual object which suggests that possibly different β values should be used for each dataset part.

For example, for a person image, one topic might concern eyes' characteristics, one skin's texture etc. However, topics are first generated for images and then (providing some explicit connection) for captions. In any case, allowing the algorithm to use different β's might be an interesting extension to be further studied.

5.3.2.3 Optimal parameters selected

The results of the experimental procedure described in the previous section, are summarized in tables 5.2 and 5.3. Values in square brackets indicate that for each number in the set the performance was almost identical.

Topics' number	**β values ordered according to best MAP**			
5	0.16	0.04	0.10	0.07
15	0.04	0.10	0.16	0.07
20	0.07	0.16	0.04	0.10
25	0.16	0.07	0.10	0.04
35	0.10	0.16	0.04	0.07
100	0.10	0.04	0.07	0.16

Table 5.2: β *values for each topics' number K, sorted starting from the one which results in best performance.*

Dataset id	**The value of K ordered**
dSIFT500	[35, 100], 15, 20, 25, 5
cSIFT300	[100, 15], 25, 35, 20, 5
dSIFT300	15, 25, 100, [5, 35, 20]
hSIFT300	[15, 100], 25, 35, 20, 5

Table 5.3: *The topics' number parameter K for each dataset, sorted starting from the one which results in best performance.*

We can observe some similarities between the ordered β vectors for K's which are close. For example, for K=5 and K=15, $\beta = 0.07$ is the worst and $\beta = 0.04$ is the best for K=15 and the second best for K=5. For K=20 and K=25, different values dominate: $\beta = 0.07$ and $\beta = 0.16$. For the large K's tried, i.e K=35 and K=100, $\beta = 0.1$ is the best in both cases. However, this connection is not strong enough to allow us find some underlying rule, because we can also see that, for example, the β-vectors for K=15 and K=20 are very different, although the number of topics only differs by 5. What is more, the relative values of K's do not seem to imply some connection with the relative values of β's. For example, when K is only 5, the best β found is 0.16, i.e the largest of all, whereas for K=15 the best β is found to be 0.04, i.e the smallest of all. Thus, we confirm that the optimal value for hyperparameter β is indeed connected with K. Nevertheless, a single rule for selecting the best β given K cannot be learned. Consequently, conducting many experiments in order to define the optimal

β is a process which cannot be avoided.

As for the results of table 5.3, they give rise to two conclusions: Firstly, we can see that K depends mostly on the nature of the data rather than on the particular feature-set, as K=5 and K=20 are the worst choices in all cases. Moreover, the fact that there are a lot of "ties", i.e similar performances (indicated with square brackets around the corresponding K's) indicates that there may exist almost equivalent combinations of (β,K) pairs, even for very different K's. This underlines the need for correctly choosing the hyperparameters.

Chapter 6

Evaluation

In chapters 4 and 5 the SVM-based model and the LDA-based one were described. In this chapter, we demonstrate the experimental procedure which was followed in order to compare the performance of each model on the same object categorization task, using our four datasets that were created as explained in chapter 3. We also conduct a comparative analysis on the results, something that will form the basis of the conclusions drawn from this project. The comparison concerns not only the relative performance of the two models, which is the main objective, but also the relative quality of the created datasets.

Firstly we describe the evaluation measures used in order to quantify the performance of each model to each dataset. This is done while explaining the experimental procedure followed in order to acquire the results. In the second part, the results are presented an analyzed.

6.1 Evaluation measures and experimental procedure

The object categorization task is applied on the already described PASCAL dataset which contains images depicting objects of 19 different classes. As explained in section 5.3 for the LDA-based model and in 4.2.2 for the SVM-based one, their outputs are actually probability vectors for each document, where element i is the probability that class i is true for the particular document. In order to use these probability estimates we need to impose a threshold t and select only classes whose probability is over t and then count the correct outputs. A more reliable approach, though, is to perform evaluation for each threshold in the interval [0,1] and summarize as necessary. After obtaining evaluation results for each class and for all thresholds, a single number

which summarizes the performance over all classes can be computed. The evaluation results for each class separately as well as the summaries for every class are useful for analysis, if correctly presented. Finally, since we have created 4 different feature sets for the same dataset (table 3.1) we can repeat the above process for each one of these.

The way in which the procedure explained above can be implemented is depicted in figure 6.1 and is explained below.

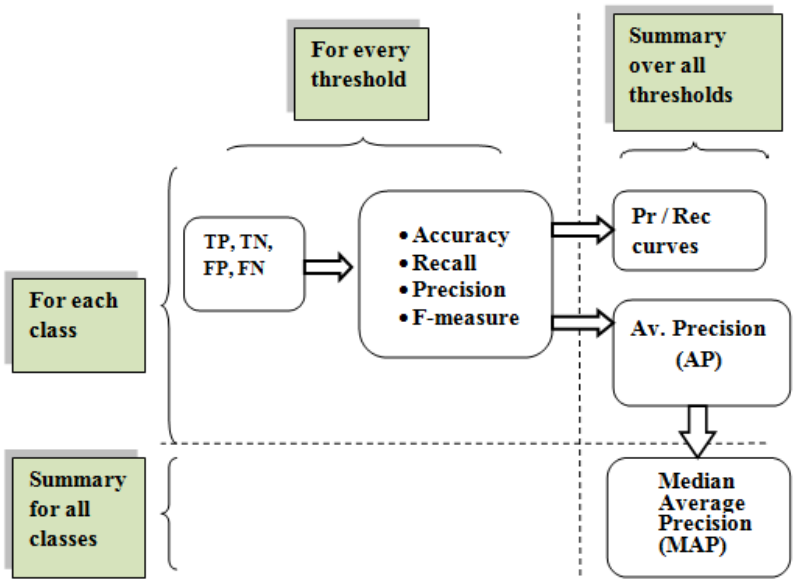


Figure 6.1: *The stages followed in order to carry out the evaluation procedure for a multi-class classification task.*

Given a specific model (classifier) and dataset, a **confusion matrix**, also known as *"contingency table"* (containing counts for True Positives: TP, True Negatives: TN, False Positives: FP and False Negatives: FN) is firstly created for each threshold in the interval [0,1]. Since this interval is a continuous one, we chose to perform the computations by beginning at zero and reaching unity with step 0.01, i.e we discretized the interval in 100 parts. *For each threshold* t and based on the confusion matrix, one can compute the following measures (where $P = TP + FN$, $N = TN + FP$):

$$Accuracy = \frac{TP + TN}{P + N} \tag{6.1}$$

$$Recall = \frac{TP}{P} \tag{6.2}$$

$$Precision = \frac{TP}{TP + FP} \tag{6.3}$$

$$F - measure = 2 \cdot \frac{Recall \cdot Precision}{Recall + Precision} \tag{6.4}$$

Accuracy is not a reliable measure, especially for unbalanced datasets like ours, be-

cause it simply constitutes the proportion of correct decisions and does not distinguish if a correct decision is a TP or a TN. For example, if a dummy classifier learns nothing about a class whose total labels represent only the 5% of the dataset and always predicts false for it, it will have accuracy equal to 95%. As can be seen from figure 3.2 there are many classes with small proportions.

High **recall** means that most of the documents which belong to class C are correctly identified and that is why it is also referred to as *"sensitivity"* or *"true positives rate"*. High **precision** means that most of the documents which are assigned class C are indeed in this category and that is why it is also referred to as *"Positive predictive value"*. However, the first measure does not "penalize" the large number of irrelevant documents which were also identified as class C, and the second does not penalize the large number of documents which are actually in class C but not identified as belonging to it. For this reason, recall and precision are very complementary to each other in analyses and are popular measures in information retrieval when studied together. The harmonic mean of recall and precision, namely **F-measure** can be used in order to find a single number which represents the quality in terms of recall and precision simultaneously.

For a fixed threshold t, obviously there is a trade-off between precision and recall, because for a t close to 1, TP and, hence, Recall will be close to zero but FP appearing in the denominator of precision will also be close to 0, meaning that precision will be larger. Thus, instead of reporting these two measures for every possible threshold, it is more useful to plot **precision-recall (P-R) curves** (as an example, see figure 6.2) because both, high precision and recall are desired. A good classification score would be represented as a P-R curve which is close to the upper and right corner of the plot. The summary provided by such a plot can also be quantified in a single number, namely **Average Precision (AP)**, defined as the average of precision over the entire range of recall. It is a measure very popular in information retrieval as well. More specifically, if precision is expressed as a function of recall, i.e $Prec(r)$, then:

$$AP = \int_0^1 Prec(r)dr \tag{6.5}$$

If precision and recall are given as vectors "*prec*" and "*r*" respectively, a practical way to find the average precision is with the following code in MATLAB: (provided with the development kit of the PASCAL challange [Everingham et al., 2008])

```
ap=0;
for t=0:0.1:1
```

```
    p=max(prec(r>=t));
    if isempty(p)
        p=0;
    end
    ap=ap+p/11;
end
```

So far we have described how to evaluate each class separately. A single number that characterizes the average score of a model over threshold for every class is needed, especially if there are many classes or if we want to compare many models. There are two ways of averaging this kind of results: **Macro averaging** and **Micro averaging**. Assume that we calculate some of the measures mentioned above (for example AP) for each of the C different classes. We use the notation m_i to refer to the value of one of the measures when applied to class c_i, $i = 1...C$. The macro-average measure is calculated as:

$$MacroAverageMeasure = \frac{1}{C}\sum_{i=1}^{C} m_i \tag{6.6}$$

that is, it simply finds the mean weighting equally the score obtained for each class. For the specific case that macro-averaging is applied to the AP measure, the resulting number corresponds to **Mean Average Precision (MAP)**.

So, for macro-averaging, the local score is derived based on a different confusion matrix for each class and then the mean is found. In contrast, in micro-averaging all confusion matrices are merged into a single table, where the values of each corresponding cell (TP, FP, TN, FN) of the local tables are summed to the new one. Then, the evaluation measure is computed in the same manner that is calculated for local measures, based on the new confusion matrix. The differences between these two averaging approaches become obvious if we consider that for more frequent classes the corresponding local confusion matrices will contain larger values. So, since macro-average takes into account a single number extracted from each of the C confusion matrices, it is said to be “category-pivoted”, as it gives the same importance to each class without considering their frequency. On the other hand, the larger numbers of frequent classes will contribute more to the sums of the new confusion matrix created for micro-averaging and that’s why this approach is said to to “instance-pivoted”, i.e it gives the same importance to each instance. Obviously, the second method mentioned reflects more the score obtained from the dominant classes [Berka et al., 2009].

Although both kinds of averages are computed from our evaluation code, the ques-

tion as to which is more appropriate to be used in comparisons remains. Answering the question "are all classes of the same importance?" will determine the method to be used. An argument in favor of using micro-averaging is that giving a benefit to frequent classes is desirable for real-world applications where the same model is trained once and used many times to classify unseen instances. This is because if for example a classifier is better at recognizing the frequent category "Person" then it is expected to succeed in most of the tasks (since most of the unseen instances are also expected to be in the category "Person"). However, we believe that for evaluation purposes all classes should be given the same importance, otherwise one could perform feature selection in order to find the optimal classifier for the most dominant class and achieve high micro-averaging results although the model might be useless for the rest of the categories. Moreover, macro-averaging "penalizes" the tendency of classifiers to assign too high probabilities to frequent classes. Thus, macro-averaging will be used in order to do model selection and comparison.

Another measure for summarizing the results over all classes is **Median Average Precision**. It is calculated by ordering the results for each class and returning the median or the mean of the two medians, if the class number is an even number, like in our case. We calculate it for the graph of figure 6.9 but the rest of our comparisons are based on Mean AP.

Having described the evaluation measures used, we can now summarize the experimental procedure followed, commenting about the implementation issues at the same time:

1. The LDA-based and the SVM models are trained with the optimal parameters (generally different for each feature-set) found during parameter estimation on the training set of each of the four feature-sets.
2. The models are applied to the four different, unseen test-sets obtaining precision and recall for each threshold in [0,1] with step 0.01, and AP is then calculated. We also compare:
   (a) AP by class
   (b) Samples of Precision-Recall curves
   (c) The best F-measures found for any threshold, potentially different for each model (although there is no way of knowing a priori the optimal threshold for each model for the specific dataset)
   (d) A plot of F-measure versus threshold for all classes (obtained from apply-

ing the models on the *validation sets*)

(e) Accuracy and F-measure for each class for the best threshold found in step 2d.

3. We compare the Mean Average Precision for each test-set (Macro average).

6.2 Results

In this section we will present the results obtained from the experiments, as described in the previous section. Since there are many different combinations of evaluation measures, models and feature sets, we have to present only a representative subset. The whole set of graphs that follow, encapsulates as much information as possible. As we will demonstrate, the best results for both models were obtained for the dataset dSIFT500, and that is why for most of the detailed comparisons are for this dataset, although graphs for more high-level comparison taking into account all of our datasets are also presented, so that we can demonstrate how the best dataset was found. Many conclusions can be drawn from the final comparison of MAP measures. However, the more detailed graphs with per-class scores are necessary not only for drawing additional conclusions but also for backing up the ones made from the table of MAP. Some more graphs can be found in the appendix.

Note that all of the evaluation results presented here are obtained from applying the model to the test sets, except the F-measure vs Threshold curve in figure 6.7 for which the validation set was used because the optimal threshold can be considered as a parameter of the models and must be learned in the validation phase.

It is worth keeping in mind that for every graph, even for subfigures under the same caption, the maximum value of Y-axis usually varies in order to present with greater detail only the area where the curves or charts lie. Finally, it is vital to repeat that for the evaluation made by the PASCAL organizers (for example, figures 6.2(d) and 6.9) the original test database was used which contained more images. However, as already mentioned in section 3.1, the gold-standard labels for this database is not distributed and so we had to create our own test database by splitting the validation set. Thus, we avoid performing extended comparisons with these results and we focus mainly on comparing our two models. On the contrary, these figures can still can be used as a simple reference and as a means of acquiring a global understanding of the results.

6.2.1 Summarizing for all thresholds

We present a sample of the precision - recall curves obtained. They correspond to classes “Person”, “Car” and “Bottle” (it is worth comparing these classes’ priors, shown in figure 3.2). Multiple observations can be made based on these figures. Firstly,

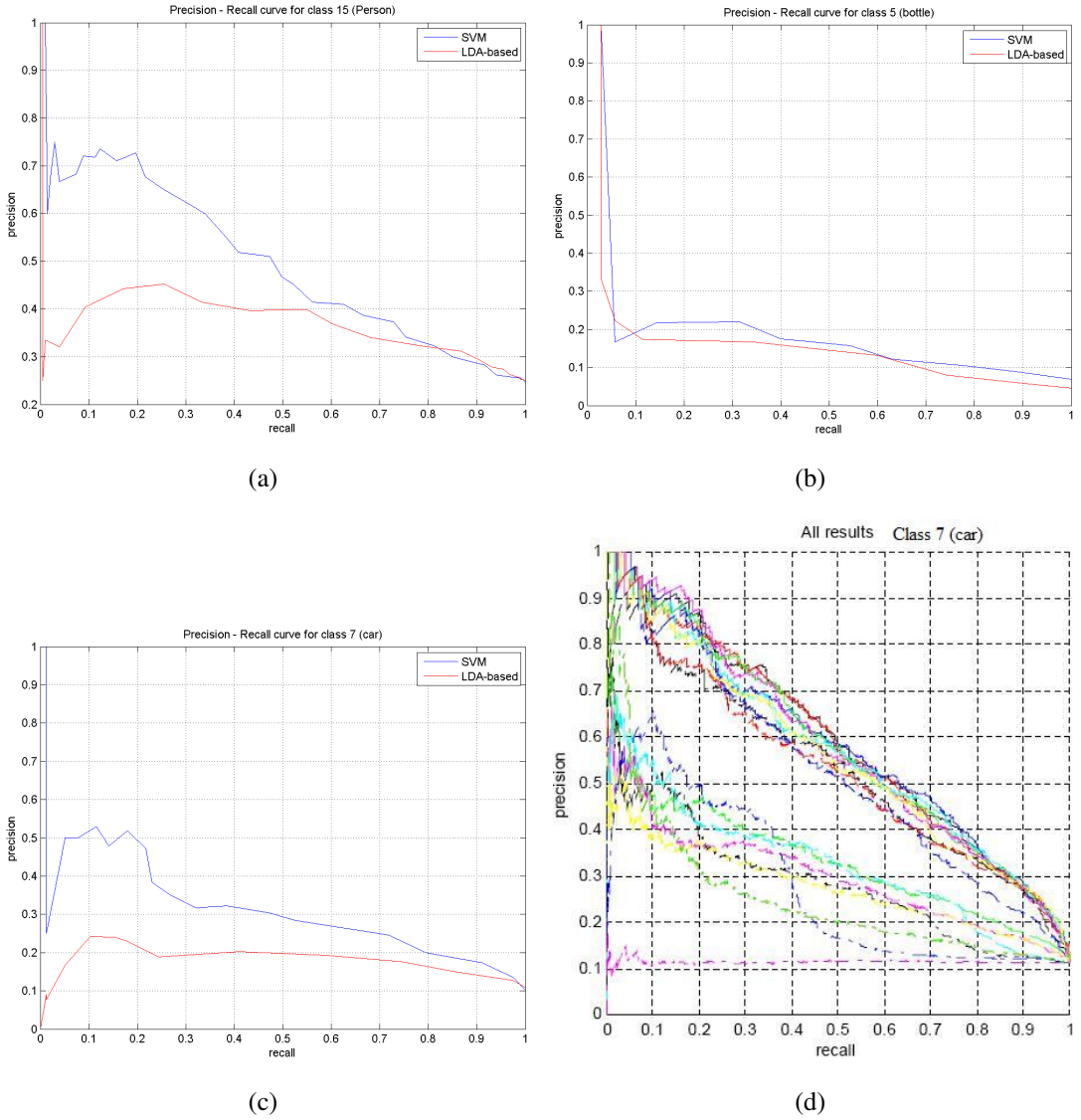
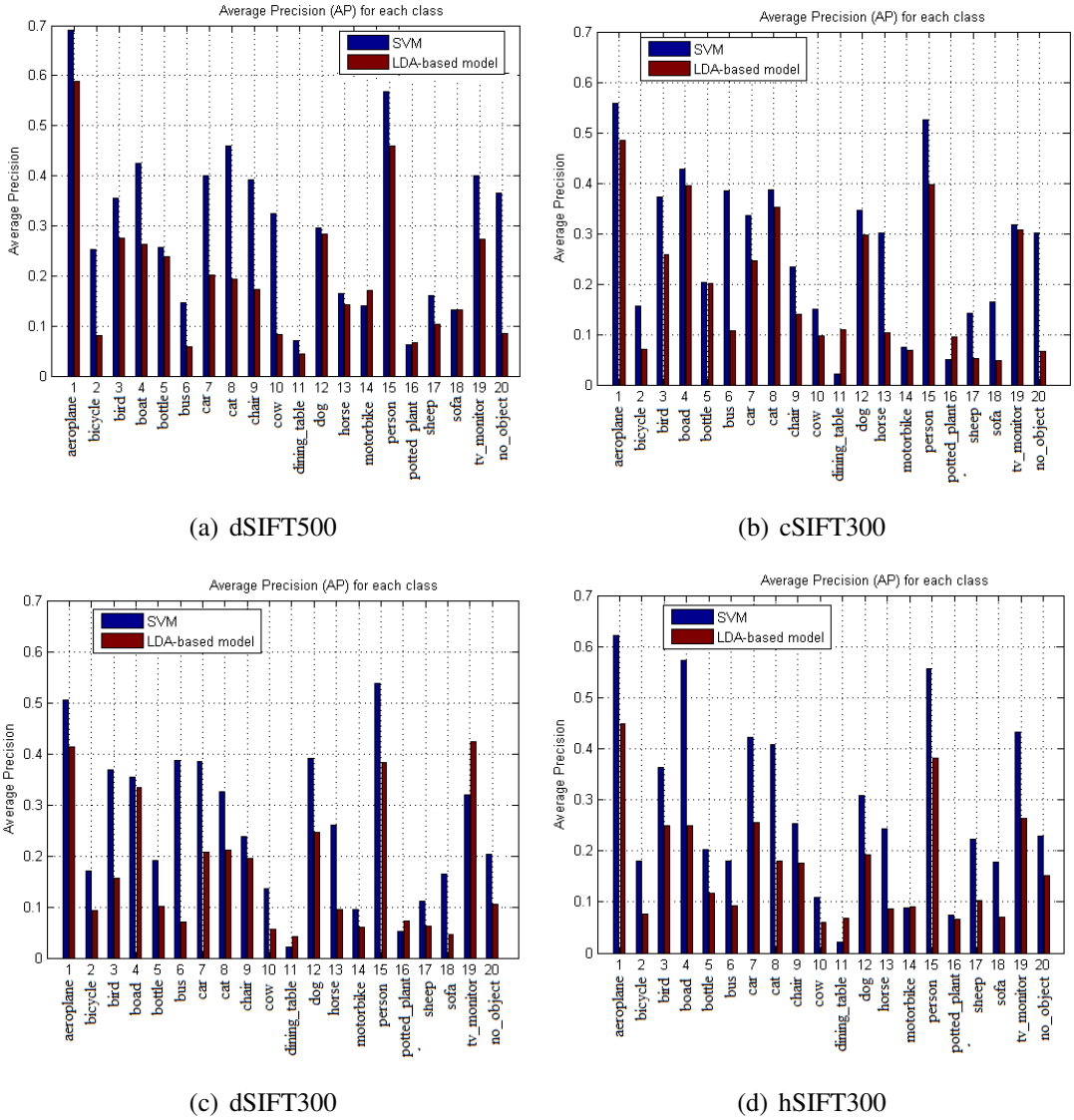


Figure 6.2: *Sample Precision-Recall curves for classes ’Person’, (a), ’Car’ (c) and ’Bottle’ (b) for the dataset dSIFT500 as well as the performance for class ’Car’ for the PASCAL challenge participants (b). Source of (b): [Everingham et al., 2008]*

by comparing with the prior of each class label in figure 3.2, we see that for both models **the performance gets better as this prior becomes larger**. This is a general phenomenon as it can be observed in all of the subsequent “per-class” graphs. Furthermore, it is evident how the SVM-model outperforms the LDA-based one in all cases. In fact, the greater the label-prior, the larger the gap in performance between SVM and

the LDA-based model. As for figure 6.2(d), one can compare with 6.9 and realize that the participants' algorithms can be grouped according to their achieved score into two categories, with a gap in performance between them. In the precision recall curves presented here, our algorithms are comparable with the group achieving the lower score, whereas for MAP, as it will be discussed next, the SVM's performance is somewhere in the middle whereas the LDA-based model's is in the lower bounds.

Next, the Average Precision (AP) is presented for each class separately. As we consider it to be the most important measure, we have included the corresponding graphs for all feature sets. Indeed, such a presentation reveals that there are certain classes for which the LDA-based model performs better.



(a) dSIFT500

(b) cSIFT300

(c) dSIFT300

(d) hSIFT300

Figure 6.3: *AP by class for dataset dSIFT500 (a), cSIFT300 (b), dSIFT300 (d), and hSIFT300 (c).*

By comparing with the figure of priors 3.2 one can confirm that such classes are those with a low prior, i.e "dining_table" and "potted_plant". It seems that, **although it is affected by class imbalances (as will be shown shortly), the LDA-based model still has a reasonable discriminative power.**

In order to compare the AP by class from a more general viewpoint, we summarize in two ways: Figure 6.4 summarizes the information shown in figure 6.3 by focusing on the different classes, i.e for each class the bar represents the *mean* of AP for all datasets. This is found by simply adding the four scores obtained and dividing with their count. On the other hand, figure 6.5 focuses on the *differences* between the score obtained for each class by the two kinds of models, for *each* dataset.

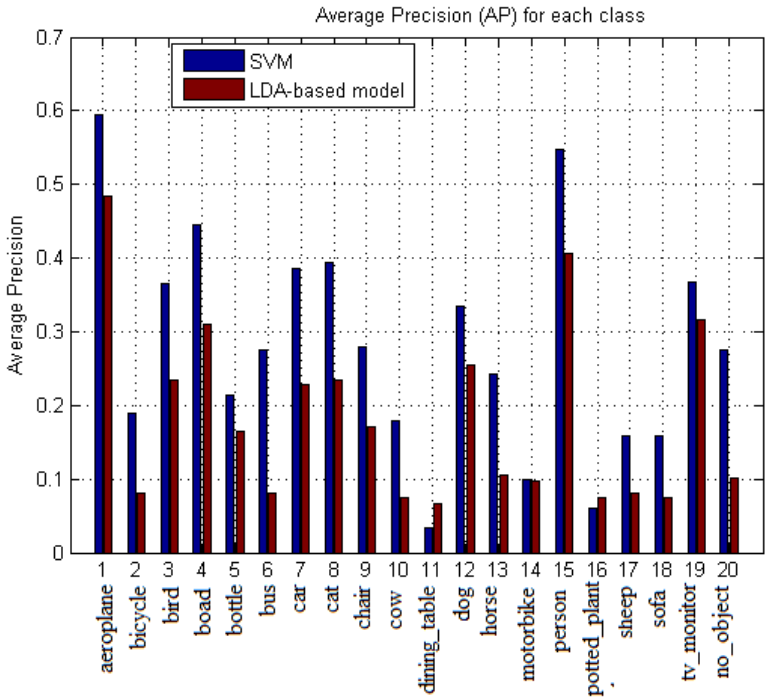


Figure 6.4: *The AP by class averaged over all datasets.*

Figure 6.4 provides a useful insight into various issues: Firstly, it can be seen that the LDA-based model is better for classes for which SVM is also better. For example, the best performance is recorded for classes "Aeroplane" and "Person" for both models. Secondly, large priors for a class label (figure 3.2) are connected to good performance, as has been commented for other graphs as well.

Figure 6.5, on the other hand, shows which of the models performed better in the particular dataset **and** class as well as how much better. This is done by setting the variable on the Y-axis to be $Y = AP(SVM)_c - AP(LDA)_c$, where $AP(SVM)_c$ and $AP(LDA)_c$ is the AP obtained from SVM and for LDA respectively for class c. This bar chart is

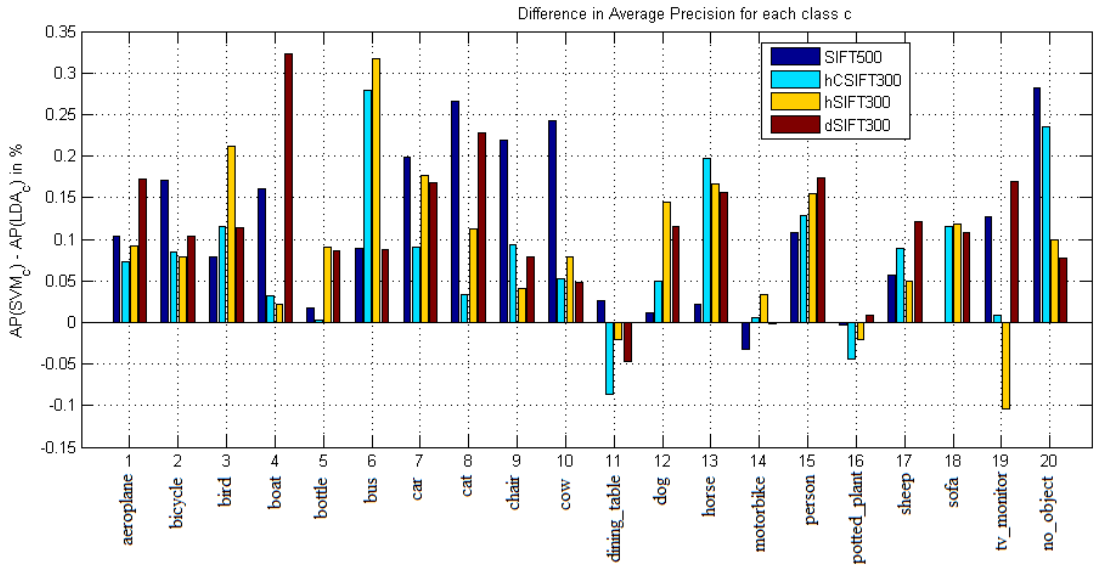


Figure 6.5: *The difference between SVM and LDA-based model in AP by class.Positive values mean that SVM performed better.*

useful for drawing conclusions either by examining the absolute values on the Y-axis or by examining the "disagreement" of the four bars above each class. For example, for the first kind of comparison one can see that dSIFT300 tends to take slightly larger values in the Y-axis, although there is not much difference (this is confirmed in table 6.1 as well). As for the second kind of comparison, one can see that for classes "Aeroplane" and "Person", almost the same difference value is computed for all four datasets. Unsurprisingly, these classes are also the ones where better performance is achieved. Apparently, this is because the classifiers learn better in these cases, either because they have a lot of samples to learn from (e.g "Person" class) or because the classes are easily being discriminated. For example, the "Aeroplane" class is easily distinguishable, because most of the pictures depicting aeroplanes contain nothing but the object itself and the sky, or a simple uncluttered background [1].

6.2.2 Using a fixed threshold

However, as already mentioned, in practical applications the models are required to be accompanied not only by a set of parameters but also by a **fixed threshold**. If the classification probability output for some class is larger than the threshold, then the class-label is emitted as true.

[1] Some example aeroplane images can be found in:
http://pascallin.ecs.soton.ac.uk/challenges/VOC/voc2008/examples/index.html#aeroplane

Figure 6.6 displays for every class and for every dataset the differences in the value of the best F-measure achieved for any threshold (potentially different for each model and dataset). A graph like this can help us understand the *potential* capabilities of the models in real applications, i.e their performance if the optimal threshold is selected. Of course, as it will be discussed shortly, it is not easy to find this threshold.

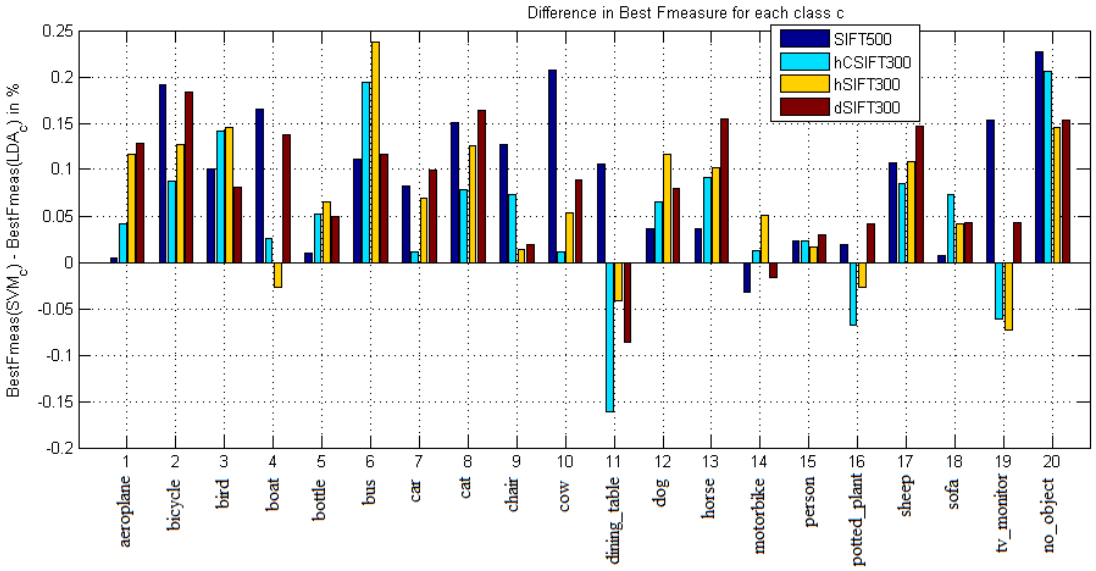
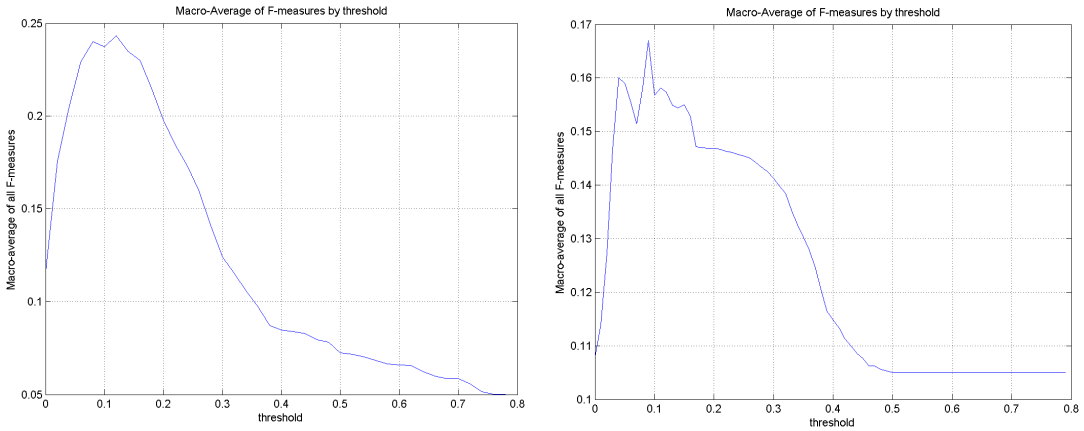


Figure 6.6: *The difference between SVM and LDA-based model in the best F-measure obtained for any threshold, by class. Positive values mean that SVM performed better.*

It is worth noting that precision-recall plots like 6.2, suggest that recall is the dimension of greatest variance and is allowed to reach greater values than precision. Thus, larger values of F-measure (calculated by equation (6.4)) can usually be calculated for larger values of recall. As can be seen from figure 6.2(a), the curve for the LDA-based model is above the one of SVM for large values of recall. Consequently, since the best F-measure is more likely to be found around this area, the value for the LDA-based model may be very close of even greater than the one for SVM. Figure 6.6 confirms this.

However, choosing a good threshold is a difficult task, because on the one hand the observations made for thresholds are not easily generalized and on the other hand it depends on the specific goals to be achieved with the object categorization task. In other words, it depends on how much we want to penalize *False Alarms* (False Positive rate) and *misses* (False Negative rate). Setting a high threshold will return more irrelevant class-labels along with the true ones, and so false alarms will be more but it is more likely for all of the true labels to be predicted correctly, so misses will be less.

In our case the task is to achieve as high performance as possible without considering whether this is achieved with a high number of misses or false alarms. Consequently, a simple way to find a good threshold would be to measure the performance on the validation set. This approach is proven to be a reasonable one, because the optimal threshold for the test set was also calculated and it was found to be about in the same point. Figure 6.7 is created by fixing the threshold to every discrete value in [0,1] with step 0.01, and calculating the macro-average of F-measures for each class.

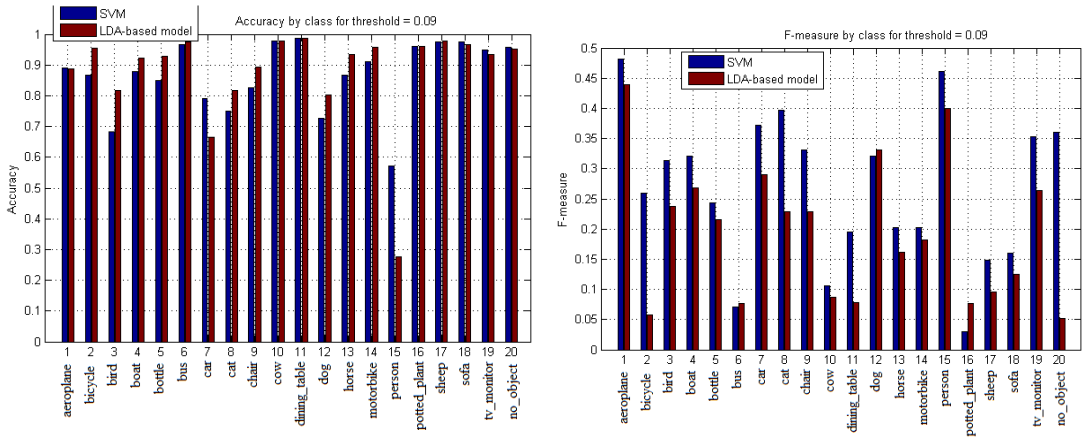


(a) SVM (b) LDA-based

Figure 6.7: *The Macro-average of F-measures for the validation set of dataset dSIFT500, plotted with threshold, for the SVM (a), and the LDA-based models (b). Obviously, threshold 0.09 is a good one for both models (for the test set).*

As can be seen, threshold 0.09 is a good one for both models, so we decided to use this for the graphs of figure 6.8. Subfigure (a) shows the accuracy obtained for each class if the threshold is fixed to 0.09 and subfigure (b) shows the F-measure.

As already explained, accuracy is not a reliable measure and figure 6.8(a) confirms this. For many classes the LDA-based model is found to perform better in accuracy, although all of the other graphs and measures suggest the opposite. Moreover, the only case where it is greatly outperformed by the SVM model is for the dominant class “Person”. Clearly, in our dataset the unbalanced data are the cause for the high accuracy recorded for rare classes. For example, the score for “dining_table” is almost 100%, although for the rest of the measures it was almost unacceptable. The reason for this, is that accuracy weights True Positives equally to True Negatives. In order to demonstrate the impact of this phenomenon, we evaluated the LDA-based model on the test set of dSIFT500 using only 20 iterations for inference and we paste a sample from the log file created, which contains the confusion matrices of each class:



(a) Dataset: dSIFT500 (b) Dataset: dSIFT500

Figure 6.8: *Threshold is fixed to 0.09. The figures show the accuracy (a) and F-measure (b) of both models by class for this threshold.*

```
class11 - dining_table:
                        Acc = 0.9868 (TP = 0 , TN = 749)
                        Recall = 0.0000
                        TN/N = 0.9987
class15 - person:
                        Acc = 0.2806 (TP = 202 , TN = 11)
                        Recall = 0.9951
                        TN/N = 0.0179
```

As can be seen, the classifier found none of the true positives whereas it found almost all true negatives ($TN/N \cong 1$) for the rare class "dining_table" whereas for the frequent class "Person" it found almost all true positives ($Recall \cong 1$) but very few true negatives. However, accuracy is very high in the first case, although the true negatives where found simply because the classifier cannot distinguish "dining_tables" in images and tends to predict "no dining_table" in all cases. From a quick calculation (based on figure 3.2 for the training set) it follows that there are only 17 labels of "dining_table" in the test set whereas there are 203 for "Person". As for the training set, the classifier has only 53 instances of "dining_table" to learn from versus 940 for "Person".

The fact that the LDA-based model is more prone to this phenomenon (as it has higher accuracy and lower F-measure and AP) means that it tends to assign higher probabilities to the most dominant classes. We conclude, thus, that ***the LDA-based model is affected more by imbalanced data***.

6.2.3 Mean Average Precision results

Finally, the results for all classes and thresholds will be summarized in a single number for each dataset: MAP measure. We can also go one step further and calculate the average for all datasets as well as the corresponding differences between the models. Table 6.1 summarizes all the above. The fact that dSIFT300 and hSIFT300 result in the same score for the LDA-based model, is just a coincidence. One can manually calculate these results from figures 6.3(d) and 6.3(c), which obviously show different scores by class which just happen to have the same mean.

Model	**dSIFT500**	**hCSIFT300**	**dSIFT300**	**hSIFT300**	**Average**
SVM	30.3%	27.3%	28.3%	26.1%	28%
LDA	19.6%	19.5%	16.9%	16.9%	18.26%
Difference	*10.7%*	*7.8%*	*11.4%*	*9.2%*	*9.74%*

Table 6.1: *The Mean Average Precision for the SVM and the LDA-based model for every feature-set.*

Figure 6.9 contains the results' chart for the PASCAL 2008 object classification competition [Everingham et al., 2008], modified in order to include the results of the models created for our project. It shows the *Median* AP achieved. We calculated and included the Median AP score obtained by the best dataset, as it is the same dataset which the experiments performed in the validation set suggested as being the best (in terms of Mean AP). As mentioned in the beginning of section 6.2, the PASCAL organizers used a different test set, so the comparison can only be made without attempting to draw detailed conclusions. In any case, though, it is obvious that both of our models had scores comparable to the ones achieved by the participants of the challenge, or at least to the group with the less impressive performance.

Table 6.1 forms the basis for some very important conclusions which regard not only the classifiers but the datasets as well:

1. The fact that **the SVM model performs better than the LDA-based one** is very obvious.
2. Both models perform the best on the dSIFT500 dataset. This suggests that **the dSIFT500 feature set is the best of all four**.

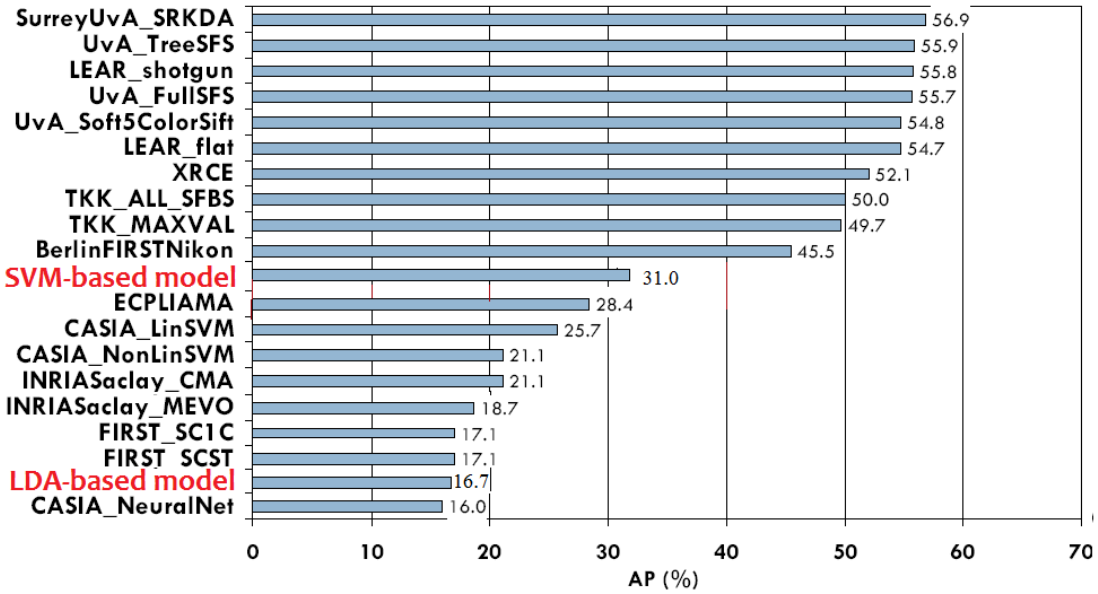


Figure 6.9: *The Median Average Precision of all participants of the PASCAL 2008 challenge, where we included our models as well. Note, however, that we used a different test set than the one used in the formal competition. source: [Everingham et al., 2008]*

3. Although the two classifiers are very different in nature, the difference in their performance for each dataset is similar, i.e in a small range of variation, which suggests that **we can easily generalize about the quality of the feature-set as well as about the capabilities of our LDA-based model**. For example, it is reasonable to believe that a third classifier is more likely to perform better in the dSIFT500 feature set as well. It is also reasonable to believe that in a different classification task, our LDA-based model would achieve scores in the same levels (when compared to SVM).
4. Comparing the scores for dSIFT500 and dSIFT300, **we conclude that the number of visual-words in the dataset is of great importance**. For the particular dataset, thus, **500 visual words are more appropriate than 300**. Of course, dSIFT500 was created with a less dense pixel-grid, but we consider that the number of attributes is by far a more important factor. Moreover, by comparing the scores on hSIFT300 and hCSIFT300, we also conclude that **color information in the descriptors results in better performance**. Finally, by comparing the scores on hSIFT300 and dSIFT300 we can see that **dense sampling is almost as good as a more sophisticated technique, i.e interest point sampling**. In fact, for SVM it is better.
5. Last but most important, we proved that **although our LDA-based model has**

not been extensively modified for the specific task of object categorization, it can perform in an acceptable level for this kind of tasks. More specifically, despite the assumptions and parametrization made for the specific object categorization task, the LDA-based model is still a model for automatic image annotation.

6.2.4 Reducing the amount of training data

The main focus of this project was not on discovering the best of the two models, because the SVM is the standard approach to the object categorization task whereas the LDA-based model is designed for a different purpose and it is not a surprise that it is outperfomed. However, through this comparison we do try to reveal interesting properties that the LDA-based model might have in contrast to SVM, so that they could potentially be used for a robust, combined model. For this reason, motivated by the work of [Quelhas et al., 2005] we designed a final experiment where the training set is reduced in two progressive steps and the models are re-evaluated. [Quelhas et al., 2005] showed that representing the data with the aid of PLSA results in algorithms which can classify scenes better even if the training set is reduced a lot.

Therefore, we trained our models with two subsets of the training set of dSIFT500: one with 1000 instances (47% of the original dataset) and one with 300 instances (14.3% of the original dataset). Notice that in the second case the number of training points is even less than the number of attributes. The test set was not changed. The results of this experiment can be seen in table 6.2. Note that for this kind of experiment we are not interested in measuring the exact performance, but we rather focus on the difference of the score in comparison with the corresponding one obtained when training on the full dataset. For this reason, we used a smaller number of Gibbs sampling iterations for the LDA-based model, and that is why the results for the full dataset do not match those of table 6.1. **Obviously, the LDA-based model is more robust to insufficient training data.** The decrease in its relative performance is 13.6% for the first subset and 25.4% for the smallest, whereas the corresponding numbers for SVM are 16.4% and 50.5%, almost reaching the LDA-based model's performance in the second case. Therefore, we confirm that this general property of algorithms based on topic models, mentioned by [Quelhas et al., 2005], who used a very different topic model and for a different purpose, is observed for our case too. The explanation given there is that by representing the data and their co-ocurrences in the topic space, we can

Data proportion	100%	47%	14.3%
Data size	2093	1000	300
SVM	30.3%	25.2% *(-16.4%)*	15.0% *(-50.5%)*
LDA-based	18.5%	16.0% *(-13.6%)*	13.8% *(-25.4%)*

Table 6.2: *The performance of the models in the test set of dSIFT500 when trained with less training data and their relative decrease in parentheses.*

capture more general information about visual co-ocurrences. This observation is very important, because it can be applied to cases where the training dataset is composed of *partially* label documents, a usual scenario for real-world applications.

Chapter 7

Conclusion and future work

7.1 Conclusion

The main objective of this project was to investigate the applicability of automatic image annotation algorithms based on topic models to a challenging object categorization task. The fact that this was carried out via a case study, allowed us to demonstrate how the combination of experiments and literature review can be used in order to take optimal decisions for this particular task in every step of the overall process, from dataset preparation to evaluation of the results. The main focus was the creation of the datasets with different feature extraction methods and, of course, building our LDA-based model by slightly modifying GM-LDA citemodellingAnnotatedData and making our own choices and assumptions to better suit the object categorization task. What is more, we proposed an efficient implementation by modifying the update equation of the Gibbs sampling algorithm which was originally proposed for performing inference for the traditional LDA.

The first set of experiments involved the parameter estimation for our models, and especially for the LDA-based one and for the specific task of object categorization. We showed that the number of topics K is related to the smoothing parameter β and that different combinations of these can result in similar performance. Unfortunately, we found no single rule which describes the relation between K and β or even between K and the total number of classes C and, thus, experiments for estimating optimal values are considered to be unavoidable. In effect, our experience from optimizing both models' parameters suggests that an SVM is more easily tuned.

The second set of experiments involved evaluating the performance of the LDA-based and the SVM model on four different feature sets. Although the LDA-based

model is clearly outperformed, its performance is still comparable to the one obtained from the SVM classifier. The fact that an algorithm taken from the image annotation domain had an acceptable performance to this different task without major modifications gives a clear answer to the question posed by our main objective, i.e if algorithms originally designed for image annotation can achieve reasonable scores in object categorization tasks. This makes us consider, thus, that our main goal was achieved. However, in order to generalize our conclusions it is vital to evaluate different LDA-based models on object categorization tasks, and this is one of the issues discussed in the next section.

Moreover, the evaluation results give some insight about the weaknesses and strengths of the LDA-based model. The fact that class labels are learned by indirectly counting co-occurrences of class labels with images via topics, in combination with the fact that each image is represented as a mixture of topics makes the LDA-based model to be more affected by imbalanced data. However, if imbalance is not an issue for a dataset, the LDA-based model was found to be more robust to other forms of "bad" datasets; firstly, because it is able to adapt better to the nature and characteristics of the image database whereas the SVM classifier relies more on choosing a good feature set. Secondly, because its performance deteriorates slower than the SVM's when the size of the training set is reduced. Finally, the fact that the training and inference phases of the LDA-based model are computationally expensive is a serious drawback, although our experiments show that when a much smaller number of Gibbs sampling iterations is used the results don't change significantly, so the trade-off is not large.

As far as the different feature extraction methods are concerned, we confirmed that the total number of visual words as well as the presence of rich color information in feature descriptors determine the overall quality of the dataset. As for feature detection, dense sampling performed as good as the Harris-Laplace feature detector.

7.2 Future work

As was already mentioned, our model is based on the GM-LDA one but we slightly modified it in order to better suit, hopefully, the object categorization task. The greatest assumption regards the type of distribution from which topics are sampled for visual words. We use a multinomial with the same β for the set of visterms and caption words as well, while most of the existing LDA-based models use a Gaussian with parameters which are learned independently. However, the fact that the performance

results are at least encouraging does not mean that these modifications worked towards the right direction. Thus, we consider that it is important to perform the same series of experiments with the original GM-LDA model. Another extension would be to allow different β parameters to be learned for the two kinds of terms (words and visual words).

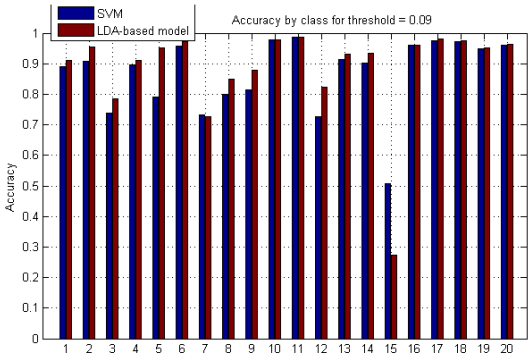
In addition, in order to learn K we designed a set of experiments based on some intuitive assumptions and previous research. Thus, although this setting is not found in an entirely heuristic way, it still relies on the results of a small number of experiments. Therefore, it would be interesting to perform parameter estimation using Bayesian methods, as described in [Griffiths and Steyvers, 2004]. The problem is that this approach needs β to be given, but using our already estimated β's would be a good starting point.

Although we focused on proving that the LDA-based model's performance is comparable to the one obtained by the SVM, it would be an interesting idea to use the more robust corr-LDA model [Blei and Jordan, 2003] in order to investigate if such models can even outperform the traditionally used SVM, as it does not suffer of the problems caused by the loose connection that GM-LDA imposes between words/topic and visual word/topic pairs. Of course, as was mentioned in section 5.1.1, this would require smoothing techniques to be used for corr-LDA so that we could get the most out of its capabilities.

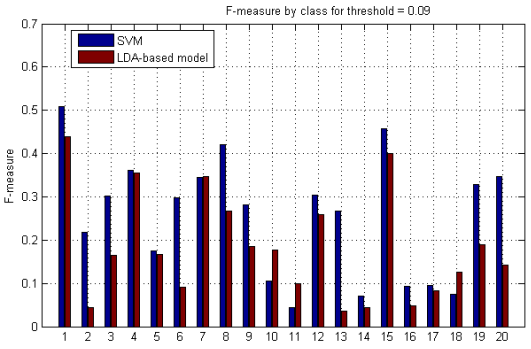
Finally, an even more interesting extension of this project would be to compare our LDA-based model, which is heavily based on one originally designed for image annotation purposes, with others that are also based on LDA but are designed especially for object categorization purposes. Comparing with the work by [Quelhas et al., 2005] and [Sivic et al., 2005] would be a reasonable start for this type of experiments.

Chapter 8

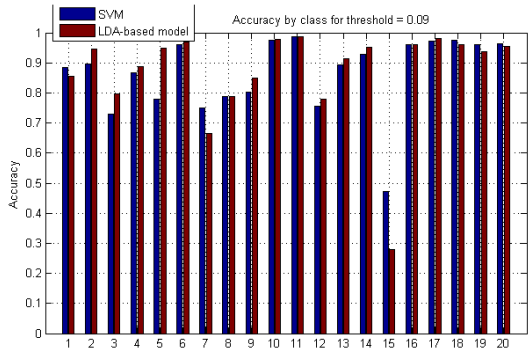
Appendix



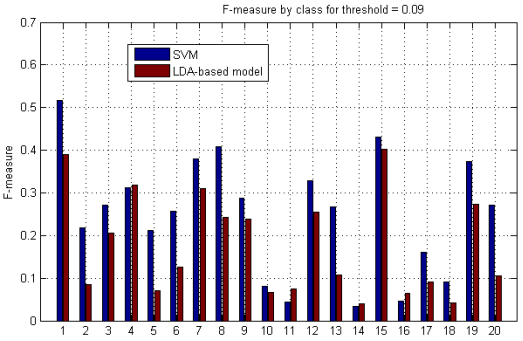
(a) Dataset: hCSIFT300



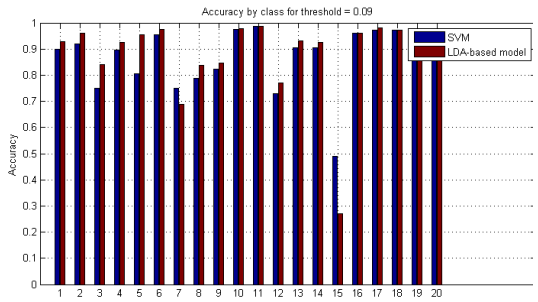
(b) Dataset: hCSIFT300



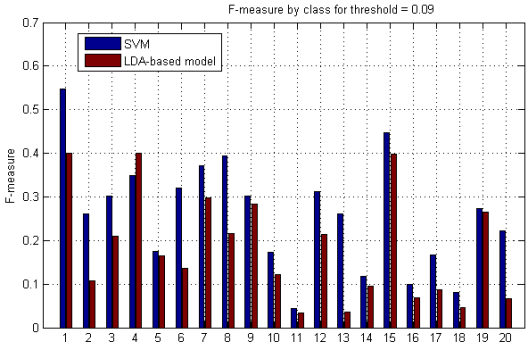
(c) Dataset: dSIFT300



(d) Dataset: dSIFT300

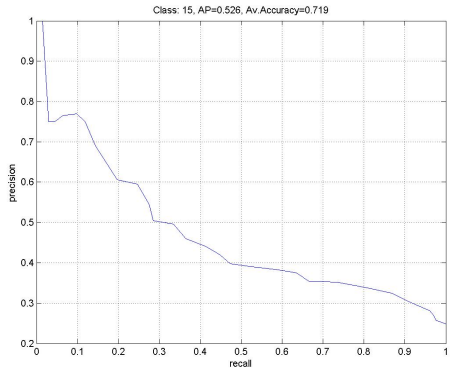


(e) Dataset: hSIFT300

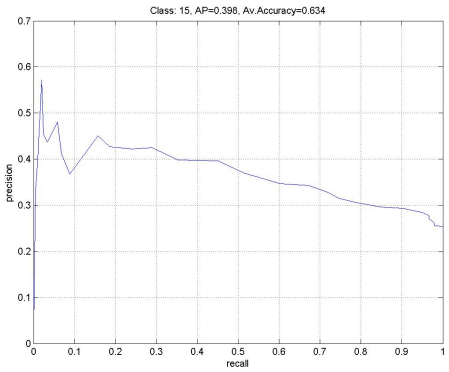


(f) Dataset: hSIFT300

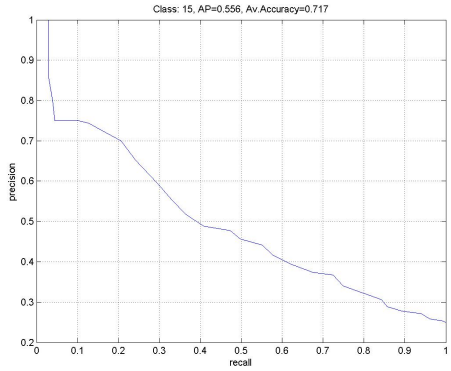
Figure 8.1: *Threshold is fixed to 0.09. The figures show the accuracy and F-measure of both models by class for this threshold.*



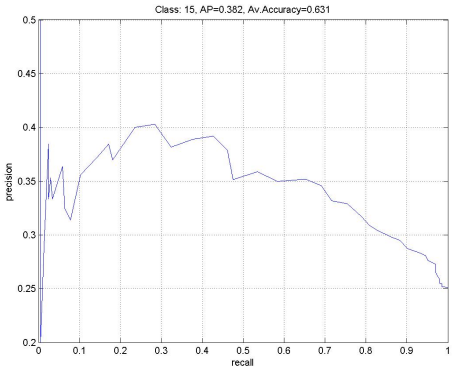
(a) SVM. Dataset: hCSIFT300



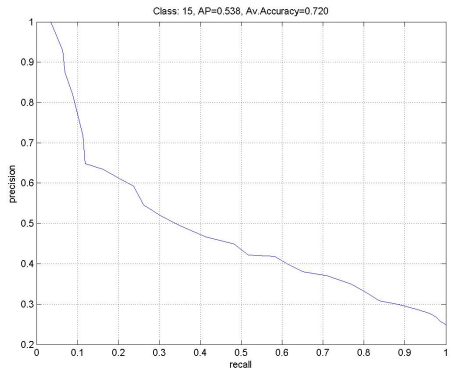
(b) LDA-based. Dataset: hCSIFT300



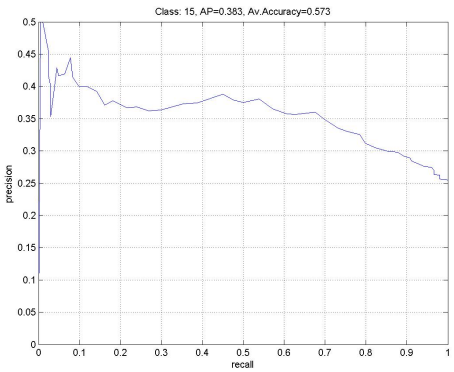
(c) SVM. Dataset: dSIFT300



(d) LDA-based. Dataset: dSIFT300



(e) SVM. Dataset: hSIFT300



(f) LDA-based. Dataset: hSIFT300

Figure 8.2: *Precision-recall figures for class 15. Note the difference in the maximum value of Y-axis.*

Bibliography

[Aizerman et al., 1964] Aizerman, M. A., Braverman, E. M., and Rozoner, L. I. (1964). Theoretical foundations of the potential function method in pattern recognition learning. *Automation and Remote Control*, 25:821–837.

[Anguita et al., 2000] Anguita, D., Boni, A., and Ridella, S. (2000). Evaluating the generalization ability of support vector machines through the bootstrap. *Neural Process. Lett.*, 11(1):51–58.

[Barnard et al., 2003] Barnard, K., Duygulu, P., Forsyth, D., Freitas, N. D., Blei, D. M., K, J., Hofmann, T., Poggio, T., and Shawe-taylor, J. (2003). Matching words and pictures. *Journal of Machine Learning Research*, 3:1107–1135.

[Baudat and Anouar, 2001] Baudat, G. and Anouar, F. (2001). Kernel-based methods and function approximation.

[Berka et al., 2009] Berka, P., Rauch, J., Zighed, D. A., Berka, P., Rauch, J., and Zighed, D. A. (2009). *Data Mining and Medical Knowledge Management: Cases and Applications*. Information Science Reference - Imprint of: IGI Publishing, Hershey, PA.

[Bishop, 2006] Bishop, C. M. (2006). *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA.

[Blei and Jordan, 2003] Blei, D. M. and Jordan, M. I. (2003). Modeling annotated data. In *SIGIR '03: Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*, pages 127–134, New York, NY, USA. ACM Press.

[Blei and Mcauliffe, 2007] Blei, D. M. and Mcauliffe, J. D. (2007). Supervised topic models.

[Blei et al., 2003] Blei, D. M., Ng, A. Y., Jordan, M. I., and Lafferty, J. (2003). Latent dirichlet allocation. *Journal of Machine Learning Research*, 3:2003.

[Bosch et al., 2006] Bosch, A., Zisserman, A., and Muñoz, X. (2006). Scene classification via plsa. In *In Proc. ECCV*, pages 517–530.

[Boser et al., 1992] Boser, B. E., Guyon, I. M., and Vapnik, V. N. (1992). A training algorithm for optimal margin classifiers. In *Proceedings of the 5th Annual ACM Workshop on Computational Learning Theory*, pages 144–152. ACM Press.

[Burges, 1998] Burges, C. J. (1998). A tutorial on support vector machines for pattern recognition.

[Chang and Lin, 2006] Chang, C.-C. and Lin, C.-J. (2006). Libsvm: a library for support vector machines.

[Cortes and Vapnik, 1995] Cortes, C. and Vapnik, V. (1995). Support-vector networks. In *Machine Learning*, pages 273–297.

[Csurka et al., 2004] Csurka, G., Dance, C. R., Fan, L., Willamowski, J., and Bray, C. (2004). Visual categorization with bags of keypoints. In *In Workshop on Statistical Learning in Computer Vision, ECCV*, pages 1–22.

[Das et al., 1998] Das, M., Manmatha, R., and Riseman, E. M. (1998). Indexing flowers by color names using domain knowledge-driven segmentation. In *In In the Proc. of the 4th IEEE Workshop on Applications of Computer Vision (WACV98), Princeton, NJ*, pages 94–99.

[Diaconis, 1977] Diaconis, P. (1977). Finite forms of de finetti's theorem on exchangeability. *Synthese*, 36(2):271–281.

[Duygulu et al., 2002] Duygulu, P., Barnard, K., de Freitas, N., Duygulu, P., Barnard, K., and Forsyth, D. (2002). Object recognition as machine translation: Learning a lexicon for a fixed image vocabulary.

[Eichhorn et al., 2004] Eichhorn, J., Eichhorn, J., Eichhorn, J., Chapelle, O., Chapelle, O., and Chapelle, O. (2004). Object categorization with svm: kernels for local features. Technical report, In Advances in Neural Information Processing Systems (NIPS.

[Everingham et al., 2008] Everingham, M., Van Gool, L., Williams, C. K. I., Winn, J., and Zisserman, A. (2008). The PASCAL Visual Object Classes Challenge 2008 (VOC2008) Results. http://www.pascal-network.org/challenges/VOC/voc2008/workshop/index.html.

[fan Wu et al., 2003] fan Wu, T., Lin, C.-J., and Weng, R. C. (2003). Probability estimates for multi-class classification by pairwise coupling. *Journal of Machine Learning Research*, 5:975–1005.

[Fei-Fei and Perona, 2005] Fei-Fei, L. and Perona, P. (2005). A bayesian hierarchical model for learning natural scene categories. volume 2.

[Foody and Mathur, 2004] Foody, G. M. and Mathur, A. (2004). A relative evaluation of multiclass image classification by support vector machines. *Geoscience and Remote Sensing, IEEE Transactions on*, 42(6):1335–1343.

[Friedman, 1996] Friedman, J. H. (1996). Another approach to polychotomous classification. Technical report, Department of Statistics, Stanford University.

[Gidudu et al., 2007] Gidudu, A., Hulley, G., and Marwala, T. (2007). Image classification using svms: One-against-one vs one-against-all. *CoRR*, abs/0711.2914.

[Grangier et al., 2006] Grangier, D., Monay, F., and Bengio, S. (2006). Learning to retrieve images from text queries with a discriminative model. In *Adaptive Multimedia Retrieval*, pages 42–56.

[Griffiths and Steyvers, 2004] Griffiths, T. L. and Steyvers, M. (2004). Finding scientific topics. *Proc Natl Acad Sci U S A*, 101 Suppl 1:5228–5235.

[Guyon and Elisseeff, 2003] Guyon, I. and Elisseeff, A. (2003). An introduction to variable and feature selection. *J. Mach. Learn. Res.*, 3:1157–1182.

[Heinrich, 2004] Heinrich, G. (2004). Parameter estimation for text analysis. Technical report.

[Hofmann, 1998] Hofmann, T. (1998). Unsupervised learning from dyadic data. pages 466–472. MIT Press.

[Hofmann, 1999] Hofmann, T. (1999). Probabilistic latent semantic analysis. In *In Proc. of Uncertainty in Artificial Intelligence, UAI99*, pages 289–296.

[Horster et al., 2008] Horster, E., Greif, T., Lienhart, R., and Slaney, M. (2008). Comparing local feature descriptors in plsa-based image models. In Rigoll, G., editor, *DAGM-Symposium*, volume 5096 of *Lecture Notes in Computer Science*, pages 446–455. Springer.

[Hsu et al., 2003] Hsu, C. W., Chang, C. C., and Lin, C. J. (2003). A practical guide to support vector classification. Technical report, Taipei.

[Jeon et al., 2003a] Jeon, J., Lavrenko, V., and Manmatha, R. (2003a). Automatic image annotation and retrieval using cross-media relevance models.

[Jeon et al., 2003b] Jeon, L. M., Lavrenko, V., Manmatha, R., and Jeon, J. (2003b). A model for learning the semantics of pictures. In *in NIPS*. MIT Press.

[Kahsay et al., 2005] Kahsay, L., Schwenker, F., and Palm, G. (2005). Comparison of multiclass svm decomposition schemes for visual object recognition. In *DAGM-Symposium*, pages 334–341.

[Karatzoglou et al., 2006] Karatzoglou, A., Meyer, D., and Hornik, K. (2006). Support vector machines in r. *Journal of Statistical Software*, 15(9):1–28.

[Lewis, 1998] Lewis, D. D. (1998). Naive (bayes) at forty: The independence assumption in information retrieval. pages 4–15. Springer Verlag.

[Lindeberg, 1996] Lindeberg, T. (1996). Edge detection and ridge detection with automatic scale selection. *International Journal of Computer Vision*, 30:465–470.

[Lowe, 1999] Lowe, D. (1999). Object recognition from local scale-invariant features. pages 1150–1157.

[Luenberger, 1984] Luenberger, D. G. (1984). *Linear and Nonlinear programming, second edition*. Addison-Wesley.

[McCallum and Nigam, 1998] McCallum, A. and Nigam, K. (1998). A comparison of event models for naive bayes text classification.

[Mikolajczyk and Schmid, 2001] Mikolajczyk, K. and Schmid, C. (2001). Indexing based on scale invariant interest points. In *In Proc. ICCV*, pages 525–531.

[Mikolajczyk and Schmid, 2004] Mikolajczyk, K. and Schmid, C. (2004). Scale & affine invariant interest point detectors. *Int. J. Comput. Vision*, 60(1):63–86.

[Mikolajczyk and Schmid, 2005] Mikolajczyk, K. and Schmid, C. (2005). A performance evaluation of local descriptors. *IEEE Trans. Pattern Anal. Mach. Intell.*, 27(10):1615–1630.

[Mikolajczyk et al., 2005] Mikolajczyk, K., Tuytelaars, T., Schmid, C., Zisserman, A., Matas, J., Schaffalitzky, F., Kadir, T., and Van Gool, L. (2005). A comparison of affine region detectors. *Int. J. Comput. Vision*, 65(1-2):43–72.

[Mori et al., 1999] Mori, Y., Takahashi, H., and Oka, R. (1999). Image-to-word transformation based on dividing and vector quantizing images with words.

[Nowak et al., 2006] Nowak, E., Jurie, F., and Triggs, B. (2006). Sampling strategies for bag-of-features image classification. In *In Proc. ECCV*, pages 490–503. Springer.

[Opelt et al., 2004] Opelt, A., Pinz, A., Fussenegger, M., and Auer, P. (2004). Generic object recognition with boosting. *PAMI*, 28:2006.

[Phan et al., 2008] Phan, X.-H., Nguyen, L.-M., and Horiguchi, S. (2008). Learning to classify short and sparse text & web with hidden topics from large-scale data collections. In *WWW '08: Proceeding of the 17th international conference on World Wide Web*, pages 91–100, New York, NY, USA. ACM.

[Philbin et al., 2008] Philbin, J., Sivic, J., and Zisserman, A. (2008). Geometric lda: A generative model for particular object discovery. In *Proceedings of the British Machine Vision Conference*.

[Platt and Platt, 1999] Platt, J. C. and Platt, J. C. (1999). Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. In *Advances in Large Margin Classifiers*, pages 61–74. MIT Press.

[Quelhas et al., 2005] Quelhas, P., Monay, F., m. Odobez, J., Gatica-perez, D., Tuytelaars, T., and Gool, L. V. (2005). Modeling scenes with local descriptors and latent aspects. In *In Proc. of IEEE Int. Conf. on Computer Vision*, pages 883–890.

[Schmid, 2006] Schmid, C. (2006). Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *In CVPR*, pages 2169–2178.

[Schmid et al., 2000] Schmid, C., Mohr, R., and Bauckhage, C. (2000). Evaluation of interest point detectors.

[Scholkopf et al., 2000] Scholkopf, B., Smola, A. J., Williamson, R. C., and Bartlett, P. L. (2000). New support vector algorithms.

[Shi and Malik, 1997] Shi, J. and Malik, J. (1997). Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22:888–905.

[Sivic et al., 2005] Sivic, J., Russell, B. C., Efros, A. A., Zisserman, A., and Freeman, W. T. (2005). Discovering object categories in image collections. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*.

[Smeulders et al., 2000] Smeulders, A. W. M., Worring, M., Santini, S., Gupta, A., and Jain, R. (2000). Content-based image retrieval at the end of the early years. *IEEE Trans. Pattern Anal. Mach. Intell.*, 22(12):1349–1380.

[Steyvers and Griffiths, 2006] Steyvers, M. and Griffiths, T. (2006). Probabilistic topic models. In Landauer, T., Mcnamara, D., Dennis, S., and Kintsch, W., editors, *Latent Semantic Analysis: A Road to Meaning*. Laurence Erlbaum.

[tien Lin et al., 2003] tien Lin, H., Lin, C.-J., and Weng, R. C. (2003). A note on platt's probabilistic outputs for support vector machines.

[Tsoumakas and Katakis, 2007] Tsoumakas, G. and Katakis, I. (2007). Multi label classification: An overview. *International Journal of Data Warehouse and Mining*, 3(3):1–13.

[Vailaya et al., 2001a] Vailaya, A., Member, A., Figueiredo, M. A. T., Jain, A. K., Zhang, H.-J., and Member, S. (2001a). Image classification for content-based indexing. *IEEE Transactions on Image Processing*, 10:117–130.

[Vailaya et al., 2001b] Vailaya, A., Member, A., Figueiredo, M. A. T., Jain, A. K., Zhang, H.-J., and Member, S. (2001b). Image classification for content-based indexing. *IEEE Transactions on Image Processing*, 10:117–130.

[van de Sande et al., 2008] van de Sande, K. E. A., Gevers, T., and Snoek, C. G. M. (2008). Evaluation of color descriptors for object and scene recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, Anchorage, Alaska, USA.

[Vapnik, 1999] Vapnik, V. N. (1999). *The Nature of Statistical Learning Theory (Information Science and Statistics)*. Springer.

[Wang et al., 2009] Wang, C., Blei, D., and Fei-Fei, L. (2009). Simultaneous image classi?cation and annotation. In *Proceedings of CVPR*.

[wei Hsu and Lin, 2001] wei Hsu, C. and Lin, C.-J. (2001). A comparison of methods for multi-class support vector machines.

[Ziou and Tabbone, 1998] Ziou, D. and Tabbone, S. (1998). Edge detection techniques - an overview. *International Journal of Pattern Recognition and Image Analysis*, 8:537–559.