Learning Aspect Models with Partially Labeled Data

THÈSE

présentée et soutenue publiquement le 30 Juin 2008

pour l'obtention du

Doctorat de l'Université Pierre et Marie Curie - Paris 6

Specialité: Informatique

par

Anastasia Krithara

Composition du jury :

Rapporteurs :	Yves Grandvalet	CNRS - Université de Technologie de Compiègne
	Marco Saerens	Université catholique de Louvain
Examinateurs :	Matthieu Cord	LIP6 - Université Pierre et Marie Curie
	Eric Gaussier	CLIPS - Université Joseph Fourier (Grenoble I)
Directeurs :	Massih-Reza Amini	LIP6 - Université Pierre et Marie Curie
	Patrick Gallinari	LIP6 - Université Pierre et Marie Curie
	Jean-Michel Renders	Xerox Research Centre Europe

Στους γονείς μου

Acknowledgments

First and foremost I want to thank my supervisors Massih-Reza Amini, Jean-Michel Renders and Patrick Gallinari. It has been an honor to work with them. I really appreciate all their contributions of time and ideas to make my Ph.D. experience productive and stimulating. Their help has been proved indispensable. I would like also to thank Cyril Goutte who has supervised me during the first year of my Ph.D. I appreciate the support and advices that he continued to offer me over the last two years, even from the other side of the Atlantic.

I would like to thank the two Labs where I did my work: Xerox Research Centre and LIP6 of the University Pierre et Marie Curie (Paris 6). I am grateful to all the people who have contributed immensely to my personal and professional time at Xerox Research Centre. They have been a source of friendships as well as good advice and collaboration. I am also grateful to the members of LIP6, and especially to the Ph.D and ex-Ph.D. students of MALIRE group, who have been really nice and helpful.

I wish to express my gratitude to the members of the jury, who undertook to evaluate my work: Yves Grandvalet and Marco Saerens for reviewing this thesis. Eric Gaussier et Matthieu Cord for being members of this jury.

My time in Grenoble was made enjoyable due to the many friends that became a part of my life. I am grateful for all the beautiful moments we spent together and for always being there for me.

Finally, special thanks are in order for my family for their love and encouragement through all these years.

Abstract

Machine learning techniques have been used for various information access tasks, such as categorization, clustering or information extraction. Acquiring the annotated data necessary to apply supervised learning techniques is a major challenge for these applications, especially in very large collections. Annotating the data usually requires humans who can read and understand them, and is therefore very costly, especially in technical domains.

Over the last years, two main approaches have been explored towards this direction, namely *semi-supervised* (SSL) and *active learning*. Both paradigms address the issue of annotation cost, but from two different perspectives. On the one hand, semi-supervised learning tries to learn by taking into account both labeled and unlabeled data. On the other hand, active learning tries to find the most informative examples to label, in order to minimize the number of labeled examples necessary for learning. Either methods try to reduce the human labeling effort.

In this thesis, we address the problem of reducing this annotation burden. In particular, we investigate extensions of aspect models for the classification task, where the training set is partially labelled. We propose two semi-supervised PLSA algorithms, which incorporate a mislabeling error model. We then combine these semi-supervised algorithms with two active learning algorithms. Our models are developped as extensions of the classification system previously developed in Xerox Research Centre Europe. We evaluate the proposed models in three well-known datasets and in one coming from a Business Group of Xerox.

Keywords: Aspect Models, Semi-Supervised Learning, Active Learning, Categorization

Résumé

L'apprentissage automatique a été utilisé pour diverses tâches d' accès à l' information, tels que la catégorisation, le clustering ou l' extraction d' information. Acquérir les données annotées nécessaires pour appliquer les techniques d' apprentissage supervisé est un défi majeur pour ces applications, en particulier pour les très grandes collections. L'annotation des données nécessite généralement l'effort humain et c'est donc très coûteux, en particulier dans les domaines techniques.

Au cours des dernières années, deux grandes approches ont été explorées dans ce sens, l'apprentissage semi-supervisé et l'apprentissage actif. Les deux paradigmes abordent la question du coût d'annotation, mais de deux points de vue différents. D'une part, apprentissage semi-supervisé essaie d'apprendre en tenant compte à la fois des données annotées et non-annotées. D'autre part, l'apprentissage actif tente de trouver les meilleurs exemples à annoter, afin de réduire au minimum le nombre d'exemples annotés necessaire. Chacune des méthodes tentent de réduire l'effort humain d'annotation.

Dans ce travail, nous abordons le problème de la réduction du coût annotation. En particulier, nous étudions des extensions de modèles d'aspect pour le tâche de la classification, où les données sont partiellement annotées. Nous proposons deux variants semi-supervisé de l'algorithme PLSA, qui incorporent un modéle d'erreur. Nous combinons ensuite ces algorithmes semi-supervisé avec deux algorithmes d'apprentissage actif. Nos modèles sont conçus comme des extensions de le système actuel pour la classification de Xerox. Nous évaluons les modèles proposés sur quatre bases de données, dont une en provenance d'un Business Group de Xerox.

Mots-clés: modéles d'aspect, apprentissage semi-supervisé, apprentissage actif, catégorisation

Contents

1 Introduction **1**

1.1 General 1

1.2 Contributions 3

1.3 Outline of this thesis 4

I Literature Review **5**

2 Learning with Partially Labeled and Mislabeled Training Data **7**

2 .1 Introduction 7

2 .2 Semi-Supervised Learning 9

2 .2.1 Transductive Learning 12

2 .2.1.1 Transductive Support Vector Machine . . 12

2 .2.1.2 Graph-based methods 16

Graph Mincuts. 17

Markov Random Walks. 18

Label Propagation. 19

Linear Neighborhood Propagation (LNP). . 20

Gaussian Fields and Harmonic functions. . 21

Spectral Graph Transducer. 22

Conditional Harmonic Mixing. 23

2 .2.1.3 Manifold methods 23

Linear dimensionality reduction. 24
Nonlinear dimensionality reduction. 25
2 .2.2 Inductive Learning 26
2 .2.2.1 Generative Methods 27
Semi-Supervised Naive Bayes. 27
Semi-supervised clustering with constraints. 29
2 .2.2.2 Discriminative Methods 30
Self-training. 30
Co-training. 31
Co-boosting. 34
2 .3 Mislabeling Error Models 37
2 .3.1 Semi-Supervised learning with mislabeled data . . 40
2 .4 Conclusion 44

3 Active Learning 45
3 .1 Introduction 45
3 .2 Active Learning Techniques 47
3 .2.1 Certainty-based sampling 47
3 .2.2 Query By Committee 48
3 .2.3 Expected error minimization 50
3 .3 Theoretical views of Active learning 51
3 .4 Combining SSL and Active Learning 53
3 .5 Conclusion 58

II Active and Semi-Supervised Aspect Models 59

4 Semi-Supervised Aspect Models 63
4 .1 Introduction 63
4 .2 Aspect Models for Document Classification 64
4 .3 Probabilistic Latent Semantic Analysis 65
4 .4 ssPLSA with a “missing values” model 68

4 .5 ssPLSA with a fake label model 70
4 .6 ssPLSA-mislabeling with hard clustering 73
4 .7 ssPLSA-mislabeling with soft clustering 78
4 .8 Conclusion 81

5 Active Semi-supervised Aspect Models 83
5 .1 Introduction 83
5 .2 Margin-Based Method 85
5 .3 Entropy-Based Method 86
5 .4 Conclusion 88

6 Evaluation 89
6 .1 Introduction 89
6 .2 Document Categorization 90
6 .2.1 Data Representation 91
6 .3 Datasets 93
6 .4 Evaluation Measures 96
6 .5 Experiments 96
6 .5.1 ssPLSA Results 101
6 .5.2 Active ssPLSA Results 106
6 .6 Conclusion 111

7 Conclusions 113
7 .1 Contributions 113
7 .2 Future Perspectives 114

Bibliography 116

List of Figures

2 .1 A simple example which demonstrate the usefulness of unlabeled examples (small dots), in different cases. The dotted line shows the correct decision border. The dark line is the estimated border, taking into account the available data in each case. When the cluster assumption holds, unlabeled data can help (b). But when it does not (c), they cannot provide any useful information. When the modelling assumption is incorrect, they can even degrade the performance (d) 11

2 .2 In TSVM, the unlabeled examples (small dots) put the decision boundary in low density regions 13

2 .3 A simple example which demonstrates the minimum cut in the graph, and annotation the unlabeled examples (white nodes) accordingly 18

2 .4 Self-Training algorithm 31

2 .5 Co-Training algorithm 32

2 .6 Filtering Techniques: They try to clean the dataset by removing the mislabeling data (the small dots and lines) and keeping only the correct labeled examples (big circles and lines) . . . 43

3 .1 Active learning: The classifier is trained with the labeled examples. Then, using a selection strategy, it chooses and demand the annotation of the most informative examples among the unlabeled ones. This procedure continues until a certain performance or a certain size of the labeled set 46

3 .2 Query By Committee. The active learner chooses the example with the biggest disagreement among the different classifiers . 49
3 .3 Combining semi-supervised and Active learning 57

4 .1 Graphical model representation of the PLSA model. Latent variables are double circled. 66
4 .2 Graphical model representation of the ssPLSA model with a fake label model.$z \in \{y + y_0\}$ 71
4 .3 Graphical model representation of the semi-supervised PLSA with a mislabeling error model, for labeled (left) and unlabeled (right) documents. 74

5 .1 Combining semi-supervised and Active learning 84

6 .1 The procedure of categorization: given some training examples already labeled (for example a set of documents or a set of images) and the specified categories, a classifier is trained. Then, the latter is able to classify new unlabeled instances to the respective categories. 90
6 .2 The documents are represented as a term-document matrix, where the frequency of a term within a document is given. . . 91
6 .3 The structure of the 20 Newsgroups dataset. 94
6 .4 F-Score (y-axis) versus, the percentage of labeled examples in the training set $|X_l|$, (x-axis) graphs for the various algorithms on `20Newsgroups` (left) and `WebKB` (right). 102
6 .5 F-Score (y-axis) versus, the percentage of labeled examples in the training set $|X_l|$, (x-axis) graphs for the various algorithms on `Reuters` dataset. 103
6 .6 F-Score (y-axis) versus, the percentage of labeled examples in the training set $|X_l|$, (x-axis) graphs for the various algorithms on `20Newsgroups`, `WebKB` and `Reuters`. 105
6 .7 F-Score (y-axis) versus, the percentage of labeled examples in the training set $|X_l|$, (x-axis) graphs for supervised and all semi-supervised variants of PLSA algorithm on `XLS` dataset. . 106

6 .8 F-Score (y-axis) versus, the number of labeled examples in the training set $|D_l|$, (x-axis) graphs for the combination of the two ssPLSA algorithms with active learning on `Reuters`, `WebKB` and `20Newsgroups` datasets 108
6 .9 Comparison of the three ssPLSA algorithms using the two different active learning algorithms, on `Reuters`, `WebKB` and `20Newsgroups` dataset 109
6 .10 Comparison of the two different active learning techniques and the Random selection, on the ssPLSA-Hard and ssPLSA-Soft algorithms, on `XLS` dataset 110
6 .11 Comparison of the three ssPLSA algorithms using the two different active learning algorithms, on `XLS` dataset 110

List of Algorithms

1 Transductive SVM (Joachims, 1999) 14
2 Graph mincuts (Blum and Chawla, 2001) 17
3 Label propagation (Zhu and Ghahramani, 2002) 20
4 Gaussian Fields and Harmonic Functions 22
5 Semi-supervised Naive-Bayes algorithm 29
6 Self Training 32
7 Co-Training (Blum and Mitchell, 1998) 34
8 Co-boosting (Collins and Singer, 1999) 36
9 The sequential steps in the Clustering-based Probabilistic Algorithm (CPA) 39
10 SSL with an explicit label-error model for misclassified data . . 42
11 Active Learning by expected Error Minimization (Roy and McCallum, 2001) 52
12 Combining active learning and semi-supervised learning using `EM` (McCallum and Nigam, 1998) 54
13 Combining active learning and semi-supervised learning using Gaussian fields and harmonic functions 56
14 Probabilistic Latent Semantic Analysis (PLSA) for Document Classification: Training 68
15 Probabilistic Latent Semantic Analysis (PLSA) for Document Classification: Testing 69
16 Semi-Supervised PLSA (ssPLSA) with fake labels 73
17 ssPLSA-mem hard 77
18 ssPLSA-mem soft 80

19 Combining ssPLSA and Active Learning 86
20 Combining ssPLSA and Active Learning 87

List of Tables

4 .1 Comparison of the different variants of the semi-supervised PLSA model. For the complexities $M = \#\{(w, x)|n(w, x) > 0\}$ 81

6 .1 Characteristics of the datasets 95

6 .2 Comparison of the F-score measures between the Naive Bayes and PLSA generative models as well as the SVM classifier on `20Newsgroups`, `WebKB` and `Reuters` test sets, where $|A|$ is the number of components. All classifiers are trained in a fully supervised way. 98

6 .3 Comparison of the F-score measures on `20Newsgroups` for the supervised PLSA. The first variant we have $|A|$ components, equally splited in the classes and the second one supposes we have one component per class plus a number of additional components 99

6 .4 F-score on `20Newsgroups` for the supervised PLSA. We start with one component per class. We calculate the heterogeneity of the components and we split them approprietily. The $|A|$ indicates the final number of components after different splits. In the second one we calculate the AIC and we choose the number for which the latter is bigger. 100

6 .5 F-score for varying proportions of labeled-unlabeled training data, for the three variants of the semi-supervised PLSA (ssPLSA-fake, ssPLSA-mem Hard, ssPLSA-mem Soft) and different numbers of the latent topics $|A|$. Bold indicates statistically better results, measured using a t-test at the 5% significance level. . 104

6 .6 Comparison of the two variants of ssPLSA with a mislabeling error model (*ssPLSA-mem Hard ssPLSA-mem Soft*) on `20Newsgroups`, `WebKB`, `Reuters` and `XLS` test sets, trained on different ratio of labeled-unlabeled data 107

1

Introduction

Contents

1.1 General 1

1.2 Contributions 3

1.3 Outline of this thesis 4

1.1 General

The explosion of available information during the last years has increased the interest of the Machine Learning (ML) community for different learning problems that have been raised in most of the information access applications. In this thesis we are interested in the study of two of these problems which are the ability of algorithms to handle partially labeled data and the capacity to model the generation of textual observations.

On the one hand probabilistic models (such as Naive Bayes) explaining the generation of observations based entirely on their classes have shown their limits in the sense that there are more and more textual documents which potentially cover different topics. New generative aspect models have recently been proposed which aim to take into account data with multiple facets. In this class of models, observations are generated by a mixture of aspects, or

topics, each of which being a distribution over the basic features of the observations (such as words in a document, or pixels in an image).

Aspect models have been succesfully used for various textual information access and image analysis tasks such as document clustering and categorization or scene segmentation. In many of these tasks, acquiring the annotated data necessary to apply supervised learning techniques is a major challenge, especially in very large data sets. These annotations require humans who can understand the scene or the text, and are therefore very costly, especially in technical domains.

To this end, the paradigm known as Semi-Supervised Learning, has emerged in the Machine Learning community in the late $90's$. Under this framework, the aim is to make a decision rule based on both labeled and unlabeled training examples. To achieve this goal, the decision rule is learned by simultaneously optimizing a supervised empirical learner on the labeled set, while respecting the underline structure of the unlabeled training data in the input space.

Different cluster, smoothness and manifold assumptions have been proposed to this end and have led to a number of semi-supervised algorithms, such as `EM`-based generative models, graph-based methods and transductive models.

In the same vein, Active Learning addresses also the issue of the annotation burden, but from a different perspective. Instead of using all the unlabeled data together with the labeled one, it tries to minimize the annotation cost by labeling as few examples as possible and focussing on the most useful examples. Different types of active learning methods have been introduced in the literature, such as uncertainty-based methods, expected error minimization methods and query by committee methods.

By combining semi-supervised and active learning, an attempt is made in order to benefit from both frameworks to address the annotation burden

problem. The semi-supervised learning component improves the classification rule and the measure of its confidence, while the active learning queries for labelling the most relevant and potentially useful examples. In this thesis, we move also towards this direction, that is the combination of semi-supervised aspect models with active learning.

In this thesis, we explore the possibility to learn aspect models with the help of a training set containing both labeled and unlabeled examples.

1.2 Contributions

In this thesis we address the problem of learning aspect models with partially labeled examples. We propose different algorithms which benefit from both semi-supervised and active learning frameworks. To the best of our knowledge, there has been little effort so far to extend aspect models to these frameworks. Our models are now in use in the context of a classification system developed previously in Xerox Research Centre Europe, namely the CategoriX/ClusteriX system. The motivation is to extend the latter under the semi-supervised and active frameworks, in order to take advantage of the huge amounts of available unlabeled datasets.

In particular we have elaborated:

- Two semi-supervised PLSA algorithms, which incorporate a mislabeling error model. The motivation is to reduce the annotation cost by taking advantage of aspect models properties.
- Combining two active learning techniques with the two semi-supervised PLSA methods above. The idea is to benefit from both the frameworks of Semi-Supervised and Active Learning, as they offer different advantages.
- Finally, an evaluation of the results in three widely used dataset and in

one coming from a Business Group of Xerox show the efficiency of our approach.

1.3 Outline of this thesis

The first part of this manuscript is composed of two chapters presenting a literature review of semi-supervised and active learning algorithms. The motivation is to give an global view of the different aspects of learning using partially labeled data. In chapter 2 we present the existing methods in semi-supervised and the mislabeling error models are discussed. In chapter 3 the active learning framework is presented.

In the second part of this thesis, we present our contributions. We are focusing on the task of document categorization and we present an extension of aspect models to the case of semi-supervised learning for this task. More precisely, in chapter 4 we present the semi-supervised PLSA models that we proposed. In chapter 5 we combine these methods with two different active learning techniques. Then, in chapter 6 the evaluation of all the above models is presented in four datasets: the three widely used collections of `20Newsgroups`, `Reuters` and `WebKB` and on the Xerox `XLS` dataset. Finally, in chapter 7, the conclusion and the future directions are given.

I

Literature Review

2

Learning with Partially Labeled and Mislabeled Training Data

Contents

2 .1 Introduction 7

2 .2 Semi-Supervised Learning 9

2 .2.1 Transductive Learning 12

2 .2.2 Inductive Learning 26

2 .3 Mislabeling Error Models 37

2 .3.1 Semi-Supervised learning with mislabeled data 40

2 .4 Conclusion 44

2 .1 Introduction

One of the major challenges in many Machine Learning (ML) tasks, such as textual Information Access (IA), Natural Language Processing (NLP) and image analysis applications, is the constitution of consistent databases, required in order to apply supervised learning techniques. Very often, skilled humans are needed in order to annotate the data, especially, in technical domains (e.g. biological data). In addition, the explosion of information during

the last years has led to a considerable increase of the cost and the difficulty of acquiring annotated data. The labeling process is so time-consuming that just a part of the available data could be labeled. On the other hand, huge amounts of unlabeled data are available and easy to obtain. The latter has stirred up the interest of the ML community to design new algorithms able to learn from partially labeled training sets. These algorithms, referred to as *Semi-Supervised Learning* (SSL) algorithms in the literature, rely on the assumption that unlabeled examples carry some useful information about the problem we try to solve.

A representative example is the information retrieval tasks in the World Wide Web. Due to the perpetual growth of the available web pages, it is impossible to have a sufficient and consistent labeled training set. On the contrary, billions of (unlabeled) web pages are available. In this case, semi-supervised learning could be of great practical value, as it could take advantage of the information contained in these data.

On the other side, the majority of the inductive methods take the quality of the training dataset for granted. Nevertheless, very often, noise is introduced in the labeling of the training set. Of course, the presence of noise can reduce the system performance in terms of classification accuracy. This led to several mislabeling learning models which have been introduced in the pattern recognition literature in the early $70's$. These studies aim at solving some practical applications such as remote-sensing, where the presence of noise is inevitable.

In this chapter, we start by presenting a synthesis of semi-supervised learning algorithms. We do not present an exhaustive list of all existing methods which have been presented in the literature. Instead, we refer to the different families of semi-supervised learning, their motivation, and the most representative methods in each of them. We start with a short discussion of the usefulness of unlabeled data. We distinguish transductive from inductive semi-supervised learning, and some transductive methods are presented.

Then, some methods coming from the two main families of semi-supervised learning models are detailled: the generative and the discriminative ones.

In the second section of this chapter, we present the problem of learning with the presence of noise in the training data. We review some existing techinques and we distinguish *random* from *non-random* imperfect supervision. We then present some work which combines mislabeling error models and semi-supervised learning.

2 .2 Semi-Supervised Learning

Semi-supervised learning can be placed in between supervised and unsupervised learning. As a result, it can be conceived from two different perspectives: either as a supervised task with some additional unlabeled data or as an unsupervised task with some additional constraints. The former is considered as semi-supervised classification, whether the latter as semi-supervised clustering.

A related family of methods is transductive learning. In this context, a partially labeled set of examples is available but, in contrast with semi-supervised learning which is inductive, the goal is to predict the labels only for the unlabeled examples in the given set, and not to derive a function. In other words, in the transductive setting, we do not have to possibility to classify any new data, but only the ones included in the training set.

Are unlabeled examples beneficial?

At this point, the question which arises is *if*, and *under which circumstances*, the amount of unlabeled data can be proved helpful. The research already conducted to answer this question has demonstrated, with theoretical and experimental results, that unlabeled examples could, under some assumptions, help and improve performance in the classification task. Nevertheless, there exists also some literature which has put some doubts about the beneficial

role of unlabeled data under certain circumstances.

From a theoretical point of view, the crucial issue to understand in what situations the unlabeled may be beneficial is still open. Some authors have tried to understand the role of unlabeled examples in the learning process. A first study was realized by (O'Neill, 1978), who considered the problem of estimating the Fisher linear discriminant using additional unlabeled data and concluded that *unclassified observations should certainly not be discarded.* (Castelli and Cover, 1995) showed that the classification error has an exponential convergence to the Bayes optimal solution, when the number of unlabeled examples grows to infinity. They generalized their finding to the situation where a finite number of labeled and unlabeled examples are available, the class-conditional densities are known, but the class priors are not (Castelli and Cover, 1996). The role of unlabeled data under the PAC framework was also analyzed by (Ratsaby and Venkatesh, 1995). Also, (Cozman et al., 2003) suggests that the unlabeled data can degrade the classification performance, when the modelling assumptions are incorrect, and it would be better if they are discarded. Finally, (Grandvalet and Bengio, 2005) proposed an estimation principle applicable to any probabilistic classifier, which benefits from the unlabeled data, especially when classes have small overlap.

Figure 2 .1 demonstrates a simple example, where we can easily notice that unlabeled examples can help (b), but sometimes not only they cannot (c), but they can even mislead the model (d), when the model assumptions are wrong.

Taking into account the above, it becomes apparent that some assumptions should hold, in order for the unlabeled examples to be meaningful. The most common assumptions are:

- *Smoothness assumption*: if two points are *close*, then they should be labeled similarly. In other words, data which belong to the same cluster (i.e. a high-density region) are likely to be in the same class. This assumption, does not imply that classes are formed from single compact

clusters. It only requires that objects from two distinct classes are not part of the same cluster.

- *Cluster assumption (a.k.a. Low density separation)*: the search of a decision boundary should take place in low-density regions. If we recall to our example in figure 2 .1 (b), where the cluster assumption holds, we can see that the decision boundary lies on the low-density region.

- *Manifold assumption*: the high-dimensional data lie on a low- dimensional manifold. In other words, the examples which belong to the same manifold, have the same class. It also does not imply that classes are formed from single compact clusters. This assumption is related to the cluster assumption, but it inspires different algorithms.

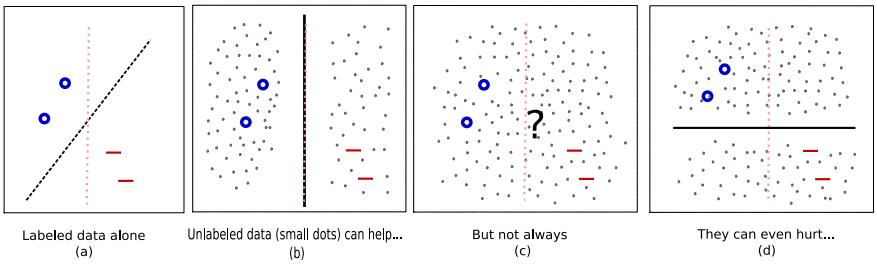

Figure 2 .1: A simple example which demonstrate the usefulness of unlabeled examples (small dots), in different cases. The dotted line shows the correct decision border. The dark line is the estimated border, taking into account the available data in each case. When the cluster assumption holds, unlabeled data can help (b). But when it does not (c), they cannot provide any useful information. When the modelling assumption is incorrect, they can even degrade the performance (d)

In the next sections, different techniques based on each of these assumptions are presented. At this point we have to mention that the importance of the unlabeled data also depends on the choice of features or, equivalently, the similarity metric we use, as the latter plays an important role on the clusters the data form.

To sum up, we can conclude that unlabeled data are helpful, as long as certain *assumptions* hold. So, before using semi-supervised learning, it would be wise to verify if some of the assumptions mentioned above hold.

Notation

Before presenting the existing methods on semi-supervised learning, some notation needs to be introduced.

We suppose that we have a collection of data $\mathcal{X} = \mathcal{X}_l \cup \mathcal{X}_u$, where $\mathcal{X}_l$ and $\mathcal{X}_u$ are respectively the set of labeled and unlabeled examples in $\mathcal{X}$. We also suppose that labeled data are sampled from the real joint distribution $p(x, y)$ and that unlabeled examples come from the marginal distribution $p(x)$. All examples from $\mathcal{X}_l$ have a class label $y \in \mathcal{C} = \{y_1, ..., y_k\}$, while for the examples from $\mathcal{X}_u$ the class label is unknown. Also, we suppose that $\mathcal{X}_{test}$ is the test set, which will be used for testing our learner, denoted as $f : X \to Y$. The test set is not available during the training.

2 .2.1 Transductive Learning

Transductive learning is closely related to semi-supervised learning. It was first introduced by (Vapnik, 1982, 1998). In transduction, in contrast with inductive learning, no general decision rule is inferred. The goal is just to annotate the unlabeled examples of the training set. In other words, it tries to find the labels $y \in \mathcal{C} = \{y_1, ..., y_K\}$ of the unlabeled examples X_u. Transductive learners cannot handle any unseen data (for example data from the test set $\mathcal{X}_{test}$). This approach is more often used for constrained clustering.

2 .2.1.1 Transductive Support Vector Machine

One of the most popular transductive methods, is the *Transductive Support Vector Machine*(TSVM) algorithm, which was first introduced by (Vapnik, 1998). TSVM is the extention of the standard SVM, where additional unlabeled data are available. It uses the information of these unlabeled samples

and predicts the optimal labels for them. The goal is to find a maximum margin hyperplane classifier based on the labeled training examples, but at the same time try to place this hyperplane away from the unlabeled data. TSVM follows the *low density separation*, as it tries to place the decision boundary in the less dense regions (e.g. figure 2 .2).

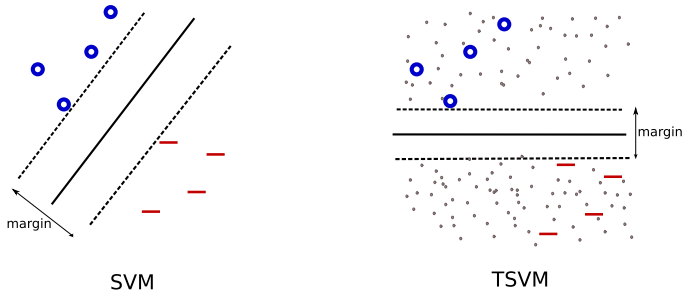

Figure 2 .2: In TSVM, the unlabeled examples (small dots) put the decision boundary in low density regions

Let us suppose that our hypothesis space H is a set of hyperplanes $h(x) = sign\{xw + b\}$. It tries to predict the labels $y_1^*, \ldots, y_n^*$ of the unlabeled data, and to find a hyperplane with parameters $\langle w, b\rangle$ which separates both labeled and unlabeled data with the maximum margin. In order to achieve the above criterion, we try to minimize the function

$$\frac{1}{2}\|w\|^2 + \underbrace{C\sum_{i=0}^{k}\xi_i}_{labeled} + \underbrace{C^*\sum_{i=0}^{n}\xi_i^*}_{unlabeled} \qquad (2\ .1)$$

over $(y_1^*, \ldots, y_n^*, w, b, \xi_1, \ldots, \xi_k, \xi_1^*, \ldots, \xi_n^*)$
and subject to

$$\begin{aligned}
\forall_{i=1}^{k} &: \; y_i\left[wx_i + b\right] \geqslant 1 - \xi_i \\
\forall_{i=1}^{n} &: \; y_i^*\left[wx_i^* + b\right] \geqslant 1 - \xi_i^* \\
\forall_{i=1}^{k} &: \; \xi_i > 0 \\
\forall_{i=1}^{n} &: \; \xi_i^* > 0
\end{aligned}$$

where ξ_i are slack variables and C, C_* are parameters, set by the user.

This function corresponds to the task of finding the exact solution of an transductive SVM and is considered as an NP-hard problem. This is the reason why much effort has been done in order to find some efficient approximation algorithms.

In this context, (Joachims, 1999) introduced a different formulation of the optimization of TSVM, and proposed his SVM^{light} software. The idea is to start by labeling the test data $\mathcal{X}_u$ based on the inductive SVM classification. Then, in order to increase the influence of the unlabeled data, we increase the values of the parameters C_-^*, C_+^* (which allow trading off margin size against misclassifying training examples or excluding test examples and which are initialized to some small number), until the value C^* defined by the user is reached. Then, we switch labels of test data in order to decrease the objective function. A description of this procedure is given in algorithm 1.

Algorithm 1: Transductive SVM (Joachims, 1999)

Input :

- A set of partially labeled data $\mathcal{X} = \mathcal{X}_l \cup \mathcal{X}_u$
- parameters C, C^*
- Initialize the cost factors C_-^*, C_+^* to some small numbers

Increment the cost factors C_-^*, C_+^* up to the user defined value C^*

repeat

- Locate two test examples for which changing the class labels leads to a decrease in the current objective function 2 .1
- If these two examples exist, switch them

until *Objective function 2 .1 doesn't decrease anymore* ;

Output : predicted labels of the test examples

(Chapelle and Zien, 2005) presented a different implementation which is

based on the optimization of the objective function using the gradient descent algorithm. The ∇TSVM, as it is known, directly optimizes the objective function according to the cluster assumption. The equation 2 .1 can be rewritten, without the need of constraints, as

$$\frac{1}{2}\|w\|^2 + C\sum_{i=1}^{k} L(y_i(w \cdot x_i + b)) + C^* \sum_{i=1}^{n} L(|w \cdot x_i + b|) \qquad (2\ .2)$$

with $L(t) = max(0, 1 - t)$.

Before performing a standard gradient descent in the above equation, as it is not differentiable, the expression is transformed in

$$\frac{1}{2}\|w\|^2 + C\sum_{i=1}^{k} L(y_i(w \cdot x_i + b)) + C^* \sum_{i=1}^{n} L^*(w \cdot x_i + b) \qquad (2\ .3)$$

with $L^*(t) = max(3t^2)$.

∇TSVM uses similar heuristics for the C^*, as TSVM of (Joachims, 1999) described above.

(De Bie and Cristianini, 2004) proposed a relaxation of the transductive SVM algorithm, using Semi-Definite programming (SDP). However, due to the high dimensionality of the feasible region of the relaxed parameters, the computation remains complex and, as a result, it cannot handle large datasets. They further proposed a spectral clustering method, which approximates the original SDP method, and shrinks the feasible region of the variables.

More recently, (Collobert et al., 2006) suggested an algorithm for TSVM, which uses the concave-convex procedure (CCCP) (Yuille and Rangarajan, 2002). CCCP iteratively optimizes non-convex cost functions that can be expressed as the sum of a convex function and a concave function. The optimization is carried out iteratively by solving a sequence of convex problems obtained by linearly approximating the concave function in the vicinity of the

solution of the previous convex problem. They report an important increase in the training speed using this method.

(Sindhwani et al., 2006) used a deterministic annealing approach in order to optimize the objective function. The motivation is to solve the problem of local minima in the TSVM optimization procedure. The idea is to start by minimizing a smoothed convex version of the objective function and gradually deform it into the TSVM one.

In the same vein, (Chapelle et al., 2006) proposed a continuation approach, which also starts by minimizing a convex objective function, and uses the solution as initialization of the next less smooth function. It iterates until it reaches the original objective function.

2 .2.1.2 Graph-based methods

Another family of transductive learning algorithms consists of *graph-based* methods. They rely on the idea of creating a graph $G = (V, E)$, where the set of nodes V represents the labeled $\mathcal{X}_l$ and unlabeled $\mathcal{X}_u$ data, and the set of edges E represents the similarities between the nodes. These similarities are defined by an adjacency (or weight) matrix W, where W_{ij} is the similarity between nodes x_i and x_j. The weights can be calculated in different ways. For example, using the k-nearest neighbor method, we can assign 1 for the k nearest neighbors of a node, and 0 for the others. Another, widely used, method of assigning the weights in a fully directed graph, is to use the Gaussian kernel:

$$W_{ij} = e^{\frac{\|x_i - x_j\|^2}{2\sigma^2}} \qquad (2\ .4)$$

These methods suppose that the smoothness assumption holds, in other words, they assume that nodes connected with heavy weighted edges, tend to have the same label. This section discusses some representative algorithms and their motivation.

At this point, we have to mention that the main drawnback of all graph-

based methods, is the construction of the graph. The latter is very important for the performance of the algorithms, even more important than the choice of the algorithm itself. Nevertheless, little work has been performed towards this direction. A discussion on this matter can be found in (Zhu, 2005).

Graph Mincuts. (Blum and Chawla, 2001) proposed a method based on graph cuts (known as s-t mincut). The idea is to try to find a minimum cut on the graph (that is the cut with the smallest sum of weights), such as to separate labeled examples of different classes. Assuming we have a binary classification problem, the algorithm tries to find a minimum cut ($cut(G^+, G^-)$) on the graph $G = (V, E)$, where G^+ and G^- are the set of examples (vertices) which include the labeled examples with labels $y_i = +1$ and $y_i = -1$ respectively. Then, it annotates as positive the unlabeled examples which belong to G^+ and as negative the ones which belong to G^-. A summary of this algorithm is shown in algorithm 2.

Algorithm 2: Graph mincuts (Blum and Chawla, 2001)

Input : A weighted graph $G = (V, E)$

- Find a minimun cut of the graph, such that G^+ contains the positive labeled examples and G^- the negative ones.
- Assign the positive label to the nodes (examples) which belong to G^+ and the negative label to the ones of G^-

Output : predicted labels of the unlabeled examples

One of the problems of the above algorithm is that the predictions are based on hard classification. This is why an extension of the mincut approach is presented in (Blum et al., 2004). The idea is to add some randomness to the graph. In particular, the algorithm creates different versions of the graph, by adding each time, some random noise to the edge weights. Then, the algorithm of mincut is applied to each of these graphs and their predictions is calculated. The final predictions of the labels is determined by majority vote. That way, a kind of confidence on the predictions is calculated.

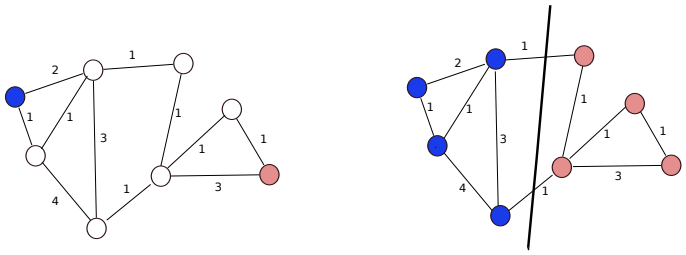

Figure 2 .3: A simple example which demonstrates the minimum cut in the graph, and annotation the unlabeled examples (white nodes) accordingly

Markov Random Walks. (Szummer and Jaakkola, 2002) proposed a graph-based algorithm, which uses Markov random walks on the graph. The idea is to start from a randomly chosen unlabeled node and walk on the graph, with transition probabilities between nodes i and j defined as

$$p_{ij} = \frac{W_{ij}}{\sum_k W_{ik}} \tag{2 .5}$$

($p_{ij} = 0$ if i and j are not connected).

We denote by $P_{t|0} = (j|i)$ the t-step transition probabilities, where t is a user defined parameter. Supposing that we have a transition matrix A, which contains the transition probabilities p_{ij} for all the nodes of the graph, we can then calculate the t-step transition probabilities as

$$P_{t|0}(j|i) = [A^t]_{ij} \tag{2 .6}$$

This is the probability that the Markov process starts from a given node i and ends in node j after t steps. These conditional probabilities $P_{0|t}(i|j)$ define our new representation for the examples. In other words, each point j is associated with a vector of conditional probabilities $P_{0|t}(i|j), i = 1, \ldots, N$. Using this representation, the points are close whenever they have nearly the same distribution over the states.

The classification model assumes that each data point has a label or a distribution $P(y|i)$ over the class labels. These distributions are unknown and represent the parameters to be estimated. Now, given a point j we interpret it as a sample for the t step Markov random walk. As labels are associated with the starting points, the posterior probabilisty of the label for point j is given by

$$P_{post}(y|j) = \sum_i P(y|i)P_{0|t}(i|j)$$

To classify the j-th point, we choose the class that maximizes the posterior:

$$c_j = \underset{c}{\operatorname{argmax}} P_{post}(y = c|j)$$

One of the problem of this algorithm, is the choice of the value of t (i.e. the length of the random walk), which is very important for the performance of the algorithm. If, for example, its value is very small, then the data are merged in small clusters. On the other hand, if it is very big, all nodes become indistinguishable. In general, the latter is calculated either by cross-validation or heuristics.

Label Propagation. In the literature, several transductive graph-based methods are based on label propagation. The idea is to start by the labeled nodes, propagate their labels to their neighbors, and iterate the process until convergence.

In this context, (Zhu and Ghahramani, 2002) presented such an algorithm. The idea is to combine random walks and clamping. The weights of the nodes and the transition probabilities are defined as in the Markov Random Walks algorithm (equations 2 .5 and 2 .6). The labels are propagated across the graph until convergence. The initial labels of the labeled examples are enforced to stay unchangeable through the iterations, in order not to loose the information they provide. This method is described in algorithm (3).

Algorithm 3: Label propagation (Zhu and Ghahramani, 2002)

Input :

- A weighted graph $G = (V, E)$, with weights W
- The diagonal matrix $D_{ii} = \sum_k W_{ik}$
- Initial labels $\widehat{y}^{(0)} = (\widehat{y}_l, \widehat{y}_u)$, with $\widehat{y}_l = y_l$ for the labeled examples and $\widehat{y}_u = 0$ for the unlabeled examples

repeat

- Propagate label $\widehat{y}^{(t+1)} \leftarrow D^{-1}W\widehat{y}^{(t)}$
- Row-normalize $\widehat{y}$
- Clamp the labeled data (i.e. $\widehat{y}_l = y_l$)

until *convergence of* $\widehat{y}$;

Output : predicted labels of the unlabeled examples

(Zhou et al., 2004) presented a similar method. It uses the normalized Laplacian $L \leftarrow I - D^{-1/2}WD^{-1/2}$, where D is the diagonal matrix and W the weight matrix. In each iteration, the labels are propagated on the graph taking into account the neighbors but also the initial value of each node. Supposing we have a parameter $\gamma \in [0, 1)$, the estimation of the labels is calculated as: $\widehat{y}^{(t+1)} \leftarrow \gamma L\widehat{y}^{(t)} + (1 - \gamma)\widehat{y}^{(0)}$, where $\widehat{y}^{(t)}$ are the estimated labels of the previous iteration and $\widehat{y}^{(0)}$ are the initial labels. The algorithm stops when $\widehat{y}$ converges. An extention of this method in directed graph is presented in (Zhou et al., 2005).

Linear Neighborhood Propagation (LNP). In the same vein, (Wang and Zhang, 2008) presented a method based on a linear neighborhood model, which assumes that each data point can be linearly reconstructed from its neighborhood. This algorithm can propagate the labels from the labeled points to the whole dataset using these linear neighborhoods with sufficient smoothness. It approximates the graph by a series of overlapped linear neigh-

borhood patches. It then aggregates the weights which are calculated for each of the above patches, in order to determine the final weights of the graph. They proved that the resulting Laplacian matrix is an approximation of the standard Laplacian matrix of a weighted undirected graph, and as a result, it can be considered as a smoothed version of the latter. The propagation of the labels is done using a similar technique as in (Zhou et al., 2004), taking into account the neighbors labels but also keeping some information of the initial labels.

Gaussian Fields and Harmonic functions. (Zhu et al., 2003a) formulates the problem in terms of a Gaussian random field on the graph. It can be seen as a nearest neighbor approach, where the neighbors are calculated using random walks on the graph. The Gaussian random field differs from the Markov random field on the fact that it is defined on a continuous state space. The goal is to find a real-valued function $f : V \rightarrow \mathbb{R}$ according to which we will assign labels to the unlabeled examples. For the labeled examples we assume that $f_l = y_l$ (their real values). We define the following energy function:

$$E(f) = \frac{1}{2} \sum_{i,j} w_{ij} \left(f(x_i) - f(x_j)\right)^2 \qquad (2.7)$$

As we can notice, according to the above function, low energy corresponds to slowly varying function over the graph. We assume the *Gaussian Random field* $p_\beta(f) = \frac{e^{-\beta E(f)}}{Z_\beta}$ where β is an "inverse temperature" parameter and $Z_\beta = \int_{f|f_l=y_L} e^{-\beta E(f)} \mathrm{d}f$ can be considered as a normalization over all functions, under the constraint that labeled examples keep their labels. It can be proved than the minimun energy function $f = argmin_{f|y_l=f_l} E(f)$ is *Harmonic*. In other terms, the value of the function f of each unlabeled example is averaged over the values of f on the neighboring points in the graph. Also, according to the maximun principle of harmonic functions (Doyle and Snell, 1984) f is unique and either satisfies the constraints $0 < f(x_j) < 1$ for X_u or is a constant. As a result, we can assign the example x_i to class 1 if $f(x_i) > 0.5$ and to class 0 differently. Algorithm 4 describes the solution using matrix methods.

Algorithm 4: Gaussian Fields and Harmonic Functions

Input :

- A weighted graph $G = (V, E)$, with weights W (equation 2 .4), and f_l the labels of the labeled examples
- The diagonal matrix $D_{ii} = \sum_k W_{ik}$
- The combinatorial Laplacian matrix $L = D - W$. We split the matrix according to labeled and unlabeled examples as: $L = \begin{bmatrix} L_{ll} & L_{lu} \\ L_{ul} & L_{uu} \end{bmatrix}$

Output :

- $f_u = -L_{uu}^{-1} * L_{ul} * f_l$

Spectral Graph Transducer. (Joachims, 2003) presented another transductive algorithm, which can been seen as the transductive version of the k nearest-neighbor (kNN) classifier. This algorithm has three main steps. First, it constructs a similarity-weighted k nearest neighbor graph G, where the weights are calculated as

$$W_{ij} = \begin{cases} \frac{sim(x_i, x_j)}{\sum_{x_k \in knn(x_i)} sim(x_i, x_k)} & \text{if } x_j \in knn(x_i) \\ 0 & \text{else} \end{cases} \quad (2 .8)$$

Then, it decomposes the G into spectrum. In order calculate the latter, it tries to minimize the normalized graph cut with constraints:

$$\min_y \frac{cut(G^+, G^-)}{|\{x_i : y_i = +1\}||\{x_i : y_i = -1\}|}$$

subject to

$$\begin{aligned} y_i &= +1, \text{ if } x_i \in \mathcal{C} \text{ and positive} \\ y_i &= -1, \text{ if } x_i \in \mathcal{C} \text{ and negative} \\ y &= \{+1, -1\}^n \end{aligned}$$

where the $cut(G^+, G^-)$ is the sum of the edges weights across the cut of the graph, and G^+ and G^- are the set of examples (vertices) with $y_i = +1$ and $y_i = -1$ respectively. In other words, it tries to minimize the average weight of the cut, instead of the sum of weights of the cut, as in $s - t$ mincuts algorithm described above. The motivation is to avoid unbalanced cuts. As the minimization is an NP-hard problem, Spectral Graph Transducer proposes an approximation to this problem, using a spectral graph method. The algorithm can be seen as an extention of the work of (Hagen and Kahng, 1992), who presented a method which uses spectral clustering for minimizing the ratio cut of a graph, but in the case of unsupervised learning. In the final step, the unlabeled examples are classified according to the subgraph (G^+ or G^-) they belong to.

Conditional Harmonic Mixing. (Burges and Platt, 2006) presented an algorithm applicable to directed graph. This method supposes that we have a directed graph and a conditional probability matrix associated to each link. The posterior class probability for each node is updated by minimizing the *Kullback-Leibler (KL) divergence* between the current distribution and the one predicted by its neighbors.

2 .2.1.3 Manifold methods

In the literature high interest has been shown for the knownledge of manifold learning. The motivation behind these methods is the fact that the structure of data can affect the answer, as it changes the notion of similarity. These methods are based on the *manifold assumption*, mentioned in the previous section: High dimensional data are distributed on some low dimensional man-

ifold. The goal is to find a low dimensional structure in high dimensional data. In other words, supposing that we have as input the training data $\mathcal{X} = \mathcal{X}_l \cup \mathcal{X}_u$ with $x_i \in \mathbb{R}^D$, we want to find their projection in the d-dimensional space, that is $\psi_i \in \mathbb{R}^d$ (where $d << D$). Originally, the existing methods addressed manifold learning in the context of unsupervised learning. The idea is to use the unlabeled data in order to estimate the geometry of the data. As such, they are very close to transductive framework and they are worth mentioning.

At this point we have to note that manifold methods do not perform classification, but they rather try to simplify the structure of the data. Nevertheless, they can use also unlabeled data in order to find a lower dimensional structure of the data, this is why there are worth mentioning.

Linear dimensionality reduction. The most known methods for linear dimensionality reduction, are the Principle Component Analysis (PCA) (Jolliffe, 1986) and the Multidimensional Scaling (MDS) (Cox and Cox, 1994). Both methods are used in the algorithms for nonlinear dimensionality reduction, described below.

The motivation of *Principle Component Analysis* is to try to preserve the covariance structure of the data set. In other words, it tries to find a d-dimensional projection of the input patterns $x_i \in \mathbb{R}^d$ in such way that distance of examples are presented as:

$$\epsilon_{PCA} = \sum_i \|x_i - \sum_{\tau=1}^{m} (x_i e_\tau) e_\tau\|^2$$

where vector e_τ, with $\tau = 1, \ldots, d$ represents a partial orthonormal basis of the input space.

The solution to this problem is the d eigenvectors having the highest eigenvalues of the centered covariance matrix ($C = \frac{1}{N_x} \sum_i x_i x_i^T$).

On the other hand, *Multidimensional Scaling*, initially designed to preserve the distance between pairs, tries to preserve the inner product between the input data. In other words, it aims at minimizing the function:

$$\epsilon_{MDS} = \sum_{ij}(x_i x_j - \psi_i \psi_j)^2$$

It starts by calculating the Gram matrix $G_{ij} = x_i x_j$. Supposing that v_τ and λ_τ are its eigenvectors and eigenvalues respectively, the outputs ψ are calculated as $\psi_\tau = \sqrt{\lambda_\tau} v_\tau$, with $\tau = 1, \ldots, d$.

Nonlinear dimensionality reduction. Spectral methods have played an important role for nonlinear dimensionality reduction and different methods have been proposed in this context. The general framework of all proposed methods is:

1. Create a k nearest neighbor graph
2. Derive a matrix from the graph weights
3. Yield low dimensional embedding from eigenvectors

At this point we have to mention that graph-based methods are nothing else than one-dimension spectral methods.

(Tenenbaum et al., 2000) proposed the *Isomap* algorithm. It can be seen as a variant of Multidimensional Scaling (MDS), where instead of Euclidean distances, it uses the geodesis ones. For the latter, it calculates the pairwise distances between all nodes along the shortest paths through the k nearest neighbor graph. It therefore uses Djikstra's algorithm. In step 3, it feeds MDS with the matrix containing the distances of the previous step.

In the same vein, *Maximum variance unfolding* has been proposed by (Weinberger and Saul, 2006; Sun et al., 2006). Like *Isomap*, it starts again with a k nearest neighbor graph, and it uses the top eigenvectors of the learned inner product matrix in order to calculate the low-dimensional embedding.

Nevertheless, it does not use the geodesic distances. Instead, it attempts to "unfold" the graph, with the help of *semidefinite programming* (SDP) (Vandenberghe and Boyd, 1996). Instead of learning the output vectors directly, the semidefinite programming aims to find an inner product matrix that maximizes the pairwise distances between any two inputs that are not connected in the neighborhood graph, that is $K_{ij} = \psi_i \cdot \psi_j$.

(Roweis and Saul, 2000; Saul and Roweis, 2003) presented an algorithm known as *Locally Linear Embedding (LLE)*. It starts also by creating a k nearest neighbor graph, but this time *directed*. It then creates a sparse matrix, which tries to capture the local geometric properties. The idea is to find a linear combination for each x_i and each neighbors, and then try to represent the same linear combination for ψ_i and its neighbors. This latter is expressed by the matrix $(I - W)^T(I - W)$, where the weight matrix W is computed by reconstructing each x_i from its neighbors. Finally, in order to calculate the d-dimensional embedding, it uses the d bottom eigenvectors of the above sparse matrix.

Laplacian eigenmaps (Belkin and Niyogi, 2003) as the Locally Linear Embedding, uses sparse matrix methods for the derivation of the matrix of the graph weights. The weight matrix W can be computed by the Gaussian kernel (equation 2 .4). Then it derives the matrix $L = I - D^{-1/2}WD^{-1/2}$, which is the normalized and symmetrized form of the Laplacian matrix. The idea is to preserve proximity relations between data. As in LLE, we choose the d bottom eigenvectors for yielding the low-dimensional structure.

2 .2.2 Inductive Learning

In semi-supervised learning, the idea is to learn a decision rule based on labeled and unlabeled data, in such a way that this decision rule can be used for the annotation of other unseen data. The semi-supervised algorithms can be separated in two main families: *Generative* and *Discriminative* methods.

2 .2.2.1 Generative Methods

In these algorithms, the goal is to start from a generative model and try to estimate the density $P(x)$. These methods are making assumptions on the nature of the data and their density.

Most generative SSL methods rely on **mixture models**. These approaches follow the *cluster assumption*. The mixture model is used to model both the input distribution and the labeling process. The labeled examples are used jointly with the unlabeled examples to estimate the mixture model, for example using the `EM` algorithm, and the labeled examples are used as a basis to assign labels to the mixture components (i.e. unlabeled data are considered as missing values in the EM algorithm (we calculate $P(x|y)$. Then we use the Bayes rule to calculate $P(y|x)$). As a consequence, the decision boundary falls in between clusters of data, and therefore in low density regions.

The introduction of the *Expectation-Maximization* (`EM`) algorithm for learning from incomplete data, was first proposed and formalized by (Dempster et al., 1977). The idea is quite simple. The method starts by initializing the model parameters using the labeled data. Then, the model is re-estimated based on unlabeled data using the `EM` algorithm. The process repeats until `EM` converges. The final model can be used to measure the performance on test data.

The idea of using `EM` algorithm to learn from labeled and unlabeled data has been brought to the attention of the Machine learning community by (Miller and Uyar, 1997), and has been applied to document classification task by (Nigam et al., 2000).

Semi-Supervised Naive Bayes. (Nigam et al., 2000) proposed a semi-supervised version of the *Naive Bayes classifier*[1] for document classification.

[1] A nice review of different variants of Naive Bayes classifier can be found in (Lewis,

The Naive Bayes classifier assumes that each example is generated by a mixture model, where each mixture component corresponds to a class $y \in \mathcal{C}$:

$$p(x, \Theta) = \sum_{k=1}^{K} p(y_k \mid \Theta) p(x \mid y_k, \Theta) \qquad (2\ .9)$$

In this model, each mixture component y_k may be selected with probability $p(y_k \mid \Theta)$, and document x is generated entirely from the selected mixture component, with probability $p(x \mid y_k, \Theta)$. For examples with no known label, the probability is given by the sum over all mixture components (equation 2 .9). In this case, each document is represented as a vector $x = \langle n(w, x) \rangle_{w \in W}$

The Naive Bayes assumption is that the features of a document are generated independently, without taking order into account. Under this assumption the probability of an example x given the class y_k is given by

$$p(x \mid y_k, \Theta) \propto \prod_{j=1}^{N_w} p_{jk}^{n(w_j, x)} \qquad (2\ .10)$$

Where, p_{jk} is the probability of generating feature $w_j \in \mathcal{W}$ in class y_k. Thus, the complete model parameters, Θ, is a set of class priors and a set of multinomial parameters:

$$\Theta = \{p(y_k) : y_k \in \mathcal{C}; p_{jk} : w_j \in \mathcal{W}, y_k \in \mathcal{C}\}.$$

Parameter estimation in a semi-supervised learning context is carried out using an `EM` algorithm, as detailed in algorithm 5. Parameters are first initialized using Maximum Likelihood estimates over the labelled data $\mathcal{X}_l \subset \mathcal{X}$ only. It then iteratively estimates the probability that each mixture component $y_k \in \mathcal{C}$ generates each example $x \in \mathcal{X}$ using the current parameters $\Theta^{(j)}$ (`E-step`), and updates the parameters $\Theta^{(j+1)}$ by maximizing the complete-data log-likelihood (`M-step`). During the `M-step`, a parameter $\lambda \in [0, 1]$ is introduced. The motivation is to weight the effect of unlabeled data. In other words, its

1998)

Algorithm 5: Semi-supervised Naive-Bayes algorithm

Input :

- A set of partially labeled data $\mathcal{X} = \mathcal{X}_l \cup \mathcal{X}_u$
- Initial model parameters $\Theta^{(0)}$ estimated over the labeled set $\mathcal{X}_l$.
- $j \leftarrow 0$

repeat

- `E-step`: Estimate the posterior class probability that each example $x \in \mathcal{X}$ belongs to each mixture component $y_k : \forall x \in \mathcal{X}, \forall y_k \in \mathcal{C}$,

$$p(y_k \mid x, \Theta^{(j)}) = \frac{p(y_k \mid \Theta^{(j)}) p(x \mid y_k, \Theta^{(j)})}{p(x \mid \Theta^{(j)})}$$

- `M-step`: Estimate the new parameters $\Theta^{(j+1)}$ which maximize the complete-data log-likelihood:

$$p(y_k \mid \Theta^{(j+1)}) = \frac{1 + \sum_{x \in \mathcal{X}} \delta(x) p(y_k \mid x, \Theta^{(j)})}{\mathcal{C} + |X_l| + \lambda |X_u|}$$

$$p_{jk}^{\Theta^{(j+1)}} = \frac{1 + \sum_{x \in \mathcal{X}} \delta(x) n(w_j, x) p(y_k \mid x, \Theta^{(j)})}{|\mathcal{W}| + \sum_{l=1}^{|\mathcal{W}|} \sum_{x \in \mathcal{X}} \delta(x) n(w_l, x) p(y_k \mid x, \Theta^{(j)})}$$

  where, $\delta(x) = 1$ if $x \in \mathcal{X}_l$ and $\delta(x) = \lambda$ if $x \in \mathcal{X}_u$

- $j \leftarrow j + 1$

until *convergence of the complete-data log-likelihood* ;

Output : A Naive Bayes classifier with parameters $\Theta^{(j)}$

goal is to control the influence of unlabeled data over labelled examples.

Semi-supervised clustering with constraints. At this point, it is worth mentioning another family of approaches, namely the *semi-supervised cluster-*

ing with constraints, which can be considered as semi-supervised generative methods. The idea is to perform clustering on the data in order to define the clusters they rely on, and use the labeled examples in order to define constraints which clusters should respect.

Different methods have been presented in the literature. They can be distinguished in two main families: the *constraint-based* and the *distance-based*. In the former the idea is to perform clustering by incorporating some kind of penalties for the violation of the constraints. Such methods have been proposed by (Demiriz et al., 1999; Wagstaff et al., 2001; Basu et al., 2002). In the distance-based methods, the idea is to perform clustering using a distance function which is parametrized using the labeled examples. Methods of this category include (Cohn et al., 2003; Xing et al., 2003).

As clustering is out of the scope of this thesis, for more details of the above methods, the reader can refer to (Basu et al., 2006).

2 .2.2.2 Discriminative Methods

Discriminative approaches focus on directly estimating the decision boundary between classes, that is the probability $P(y|x)$, without implementing the cluster assumption. Note that, although discriminative training is known to be asymptotically better than generative approaches, the latter may be preferable when the number of annotated data is limited. They make few assumptions on the nature of the data, and these hypotheses are generally weak.

Self-training. The probably earliest idea of SSL is based on the principle of self training. It has appeared early in the literature (Scudder, 1965; Spragins, 1966; Agrawala, 1970) and has been applied to different problems such as *adaptive signal processing* (Widrow and Stearns, 1985), *natural language processing* (Yarowsky, 1995), *object detection systems from images* (Rosenberg et al., 2005), *gene identification* (Lomsadze et al., 2005) and others.

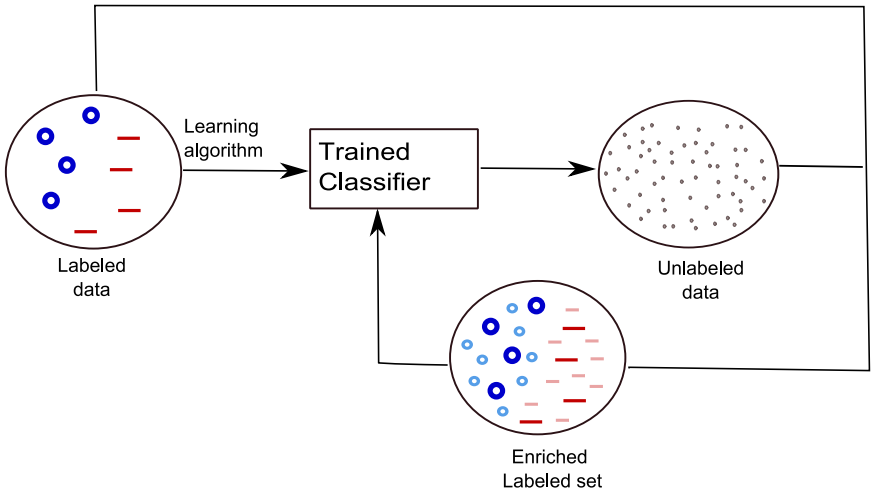

Figure 2 .4: Self-Training algorithm

The process of self-training begins with building a classifier that is trained with few labeled examples. The trained classifier is used to annotate the unlabeled examples. The ones among them with the highest confidence are added to the training set together with their predicted labels. The classifier is retrained and this procedure is repeated until there no unlabeled examples left. Another variant of self-training proposes to train the model until there is no changes in the label predictions, when a margin-based criterion is used label the unlabeled examples. This method finds the decision boundary following the *low density separation* assumption, as it tends to push the boundary far from the unlabeled data.

One of the drawbacks of self-training is the fact that it reinforces its classification errors.

Co-training. Based on the idea of self-supervised learning, (Blum and Mitchell, 1998) presented the co-training algorithm. This method supposes that we have two different modalities of the data set, under the assumption that each of them is rich enough to learn the parameters of a classifier. That is, each example x_i has two different views $x_{i,1}$ and $x_{i,2}$. It also supposes that

Algorithm 6: Self Training

Input : A partially labeled dataset $\mathcal{X} = \mathcal{X}_l \cup \mathcal{X}_u$

repeat

- Train the classifier with the labeled examples $\mathcal{X}_l^{(i)}$
- Annotate the unlabeled examples $\mathcal{X}_u^{(i)}$ using the trained classifier
- Add the most confident unlabeled examples $(\mathcal{X}_u^{'})$ with their predicted labels to the labeled set $(\mathcal{X}_l^{(i+1)} = \mathcal{X}_l^{(i)} \cup \mathcal{X}_u^{'}$ and $\mathcal{X}_u^{(i+1)} = \mathcal{X}_u^{(i)} \setminus \mathcal{X}_u^{'})$

until *all unlabeled data have been labeled* ;

Output : The model parameters

the two views of the data are consistent:

$$\exists h_1, h_2, x_i : h^{opt}(x_i) = sgn(h_1(x_{1,i})) = sgn(h_2(x_{2,i}))$$

It is also assumes that the two views are independent given the label:

$$\begin{aligned} p(x_{1,i}|x_{2,i}, y_i) &= p(x_{1,i}|y_i) \\ p(x_{2,i}|x_{1,i}, y_i) &= p(x_{2,i}|y_i) \end{aligned}$$

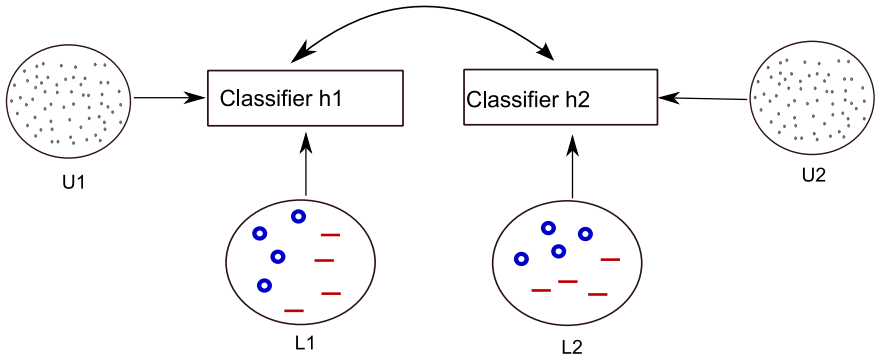

Figure 2 .5: Co-Training algorithm

Initially, two separate classifiers h_1 and h_2 are trained with the labeled set, each using one view. Then, they classify the unlabeled examples, and a subset of these examples classified with classifier h_1 are chosen randomly and used as input to classifier h_2, considering that the labels predicted by h_1 are the correct ones. Each classifier is retrained with the additional examples and their labels predicted by the other classifier. This process repeats for a given number of iterations. A more detailed description of the co-training method is given in algorithm 7.

(Nigam and Ghani, 2000) proposed a similar semi-supervised, multi-view algorithm (the algorithm **Co-EM**) which can be seen as a probabilistic version of the Co-training. The algorithm runs EM in each view and, before each iteration, it inter-changes the probabilistic labels generated in each view. The basic idea of both Co-EM and co-Training, is to use the knowledge learned in one view to train the other one. The difference between them is that Co-EM uses probabilistic labels for the labeled examples that may change from one iteration to the other.

The assumption of consistence and independence between the two views of the data that co-training (and Co-EM) is making is very strong, and it is difficult to be met in real-world applications. And this can even result a decrease in performance (Nigam and Ghani, 2000). This is why an effort has been made in order to relax these assumptions. (Goldman and Zhou, 2000) proposed a variant of co-training which does not suppose independence and consistency of the data views. Instead, it learns two different classifiers on the labeled set. Then, they annotate the unlabeled examples and they enrich the labeled sets of each other. The motivation is that the two classifiers will learn two different models which can eventually complement each other. A different way to relax this assumption is presented by (Balcan et al., 2004). In the latter, an expansion property on the underlying distribution of the data is proposed, in order to replace the conditional independence assumption of co-training.

The idea of using two classifiers has appeared earlier than co-training in the literature. For example, in the work of (De Sa, 1993, 1994) a similar algorithm is presented. The so-called *self-supervision* algorithm uses two different classifiers, which play alternatively the role of teacher and student. The output of the one is used as desired input for the other. This procedure continues until the convergence of the output.

Algorithm 7: Co-Training (Blum and Mitchell, 1998)

Input :

- A partially labeled dataset $\mathcal{X} = \mathcal{X}_l \cup \mathcal{X}_u$,
  where $\mathcal{X}_l = \{(x_{1,i}, x_{2,i}\}_{i=1,\ldots,m}$ and $\mathcal{X}_u = \{(x_{1,i}, x_{2,i}\}_{i=m+1,\ldots,m+n}$
- Two classifiers h_1 and h_2
- Create a set $\mathcal{X}'_u$, by choosing randomly N examples from $\mathcal{X}_u$

repeat

- Train classifier h_1 with the labeled examples $\mathcal{X}_l$
- Train classifier h_2 with the labeled examples $\mathcal{X}_l$
- Annotate p positive and n negative unlabeled examples from $\mathcal{X}'_u$ using the trained classifier h_1
- Annotate p positive and n negative unlabeled examples from $\mathcal{X}'_u$ using the trained classifier h_2
- Add the above examples with their predicted labels to the labeled set
- Add $2p + 2n$ examples from the $\mathcal{X}_u$ to $\mathcal{X}'_u$

until *a fixed number of iterations* ;

Output : The predicted labels and the classifiers h_1 and h_2

Co-boosting. (Collins and Singer, 1999) proposed the co-boosting algorithm, which is based on the algorithm of Adaboost (Freund and Schapire, 1997; Schapire and Singer, 1999). It builds two additive models in parallel,

with an objective function that bounds the rate of agreement. It can be considered as a variant of co-training.

We suppose that each example x_i is an instance pair $(x_{1,i}, x_{2,i})$, which represents the two modalities of the example. We also suppose that we have a partially labeled dataset of size $m+n$, where the first m pairs have labels y_i, for $i = 1, \ldots, m$ and the rest n are unlabeled. The algorithm makes the fairly strong assumption, that each of the two modalities of each example $x_{1,i}$ and $x_{2,i}$ is sufficient in order to determine its label.

Let us denote with g_1 and g_2 the two classifiers. For the labeled data we suppose that $sign(g_1(x_{1,i})) = sign(g_2(x_{2,i})) = y_i$. For the unlabeled data, we suppose that $sign(g_1(x_{1,i})) = sign(g_2(x_{2,i}))$. The two classifiers, are built during the iterations of the algorithm, by updating each time the equation:

$$\forall i, g_j^t(x_{j,i}) = g_j^{t-1}(x_{j,i}) + \alpha_t h_t^j(x_{j,i}) \quad (2.11)$$

The α_t corresponds to the confidence value and is calculated also during the iterations as follows:

$$\alpha_t = \frac{1}{2} ln \left(\frac{W_+ + \epsilon}{W_- + \epsilon} \right) \quad (2.12)$$

where W_+ and W_- are computed for each possible hypothesis $h_t(x_i)$ (see algorithm 8 for details).

The ϵ corresponds to a smoothing parameter, and it is introduced in order to avoid the extreme confidence values, which may appear when, for example, a feature is present in very few examples.

The algorithm tries to minimize the sum of the classification error on the labeled examples and the number of disagreements between the two classifiers on the unlabeled data. In other words, on each step, the algorithm searches

Algorithm 8: Co-boosting (Collins and Singer, 1999)

Input : $\{(x_{1,i}, \ldots, x_{2,i})\}_{i=1,\ldots,n+m}$ and $(y_i)_{i=1,\ldots,m}$

Initialize: $\forall i, j : g_j^{(0)}(x_i) = 0$

repeat

- Set pseudo-labels:

$$\tilde{y}_i = \begin{cases} y_i & \text{if } 1 \leq i \leq m \\ sign\left(g_{3-j}^{t-1}(x_{3-j,i})\right) & \text{if } m < i \leq n+m \end{cases}$$

- Set virtual distribution:

$$D_t^j(i) = \frac{1}{Z_t^j} e^{\left(-\tilde{y}_i g_j^{t-1}(x_{j,i})\right)}$$

  where Z_t^j is the normalization term, i.e. $Z_t^j = \sum_{i=1}^{n+m} e^{\left(-\tilde{y}_i g_j^{t-1}(x_{j,i})\right)}$

- Train the classifier h_t^j using the distribution D_t^j
  (i.e. each observation is weighted differently for different t)

- Choose the weights $\alpha_t \in \mathbb{R}$ of the obtained classifier $\alpha_t = \frac{1}{2} ln\left(\frac{W_+ + \epsilon}{W_- + \epsilon}\right)$,
  where $W_+ = \sum_{i:h_t(x_i)=y_i} D_t^j(i)$ and $W_- = \sum_{i:h_t(x_i)=-y_i} D_t^j(i)$

- Update the global classifier taking into account the classifier
  of step t $\forall i, g_j^t(x_{j,i}) = g_j^{t-1}(x_{j,i}) + \alpha_t h_t^j(x_{j,i})$

until *a fixed number of iterations t and for $j = 1, 2$;*

Output : Final hypothesis: $f(x) = sign\left(\sum_{j=1}^{2} g_j^T(x_j)\right)$

to minimize the function:

$$\begin{aligned} Z_{co} &\overset{def}{=} \sum_{i=1}^{m} \left[e^{(-y_i g_1(x_{1,i}))} + e^{(-y_i g_2(x_{2,i}))}\right] \\ &+ \sum_{i=m+1}^{m+n} \left[e^{(-sign(g_2(x_{2,i}))g_1(x_{1,i}))} + e^{(-sign(g_1(x_{1,i}))g_2(x_{2,i}))}\right] \end{aligned}$$

As we can see for the above function, small values of Z_{co} means that the two classifiers have low error rate on labeled examples and there is also low disagreement in the predictions of labels for unlabeled examples.

Each iteration of the algorithm is composed by two rounds: in each of them, one of the classifier is updated while the other remains fixed. This procedure continues for T iterations, by alternating the two classifiers.

2 .3 Mislabeling Error Models

As we mentioned in the introduction of this chapter, very often, noise can be introduced in the labeling of the training set which can reduce the system performance in terms of classification accuracy. Some proposed solutions try to capture the mislabelings within the learning algorithm. That is, they learn with the noisy data, by using mechanisms in order to capture the mislabelings. The existing studies distinguish between *random* and *no-random* imperfect supervisions: the probability of misclassification of an observation does depend on its feature vector while it does not for the former.

Random imperfect supervision. It can occur when the noise in the data depends on their feature vector. In the context of medical diagnosis for example, this could be the labeling of test blood results (Aitchison and Begg, 1976). (McLachlan, 1972) studied conditional error rates using their asymptotic expansions for the case where one group does not get mislabeled sample.

(Chittineni, 1980) considered the problem of learning from imperfectly labeled data. He used noisy data in order to analyse the Bayes classifier error and to calculate the error bounds on the performance of nearest neighbor classifiers. In the same vein, (Lugosi, 1992) investigated the asymptotic behavior of the error probability of two methods under very general conditions: the nearest neighbor algorithm and a method based on the maximization of the estimated a posteriori probabilities. In (Chhikara and McKeon, 1984), an analysis of the importance of mislabeled training data is done, and it is

proved that the training of classifiers by ignoring mislabeling in the training set can degrade classification performance. (Krishnan and Nandy, 1987) presented the derivation of the likelihood estimation of parameters for two group multivariate normal mixtures with a common covariance matrix using the maximum likelihood principle, for a binary classification problem. Following this work, Krishnan compared an imperfect and a perfect supervision scheme by measuring the *Asymptotic Relative Efficiency*, that is the number of samples needed in each of the schemes in order to achieve the same performance (Krishnan, 1988).

More recently, (Karmaker and Kwek, 2005) presented a boosting approach, namely the ORBoost (Outlier Removal Boosting). It is based on the well-known AdaBoost algorithm (Freund and Schapire, 1997; Schapire and Singer, 1999). The idea is to introduce a threshold which puts an upper bound on the weights of the noisy examples. During the iterations, the examples which are identified as outliers (that is, the examples with weights larger than the limit bound) are eliminated. As the iterations increase, the remained examples have hopefully the correct labels and the classifier have better performance.

In (Lawrence and Scholkopf, 2001), an algorithm for constructing a kernel Fisher discriminant from noisy training data is presented. The idea is to assign to each example a probability of its label being flipped. They use then the `EM` algorithm in order to update these probabilities. They assume that the class conditional densities are Gaussian distributions. (Li et al., 2006), based on the latter algorithm, presented two extensions to non-Gaussian datasets, namely the Clustering-based Probabilistic Algorithm (CPA) and the Probabilistic Kernel Fisher (PKF): the former applies the algorithm introduced by (Lawrence and Scholkopf, 2001) to a Mixture-of-Gaussians (MoG) in the input space (algorithm 9), while the latter gives a similar framework to their algorithm, but this time no distribution assumption is made.

In the first step of algorithm (9), the optimal number of mixture com-

Algorithm 9: The sequential steps in the Clustering-based Probabilistic Algorithm (CPA)

1. Estimate the number of mixture components K
2. Estimate the mixture density parameters and priors by the Mixture-of-Gaussians
3. Map clusters to classes
4. Optimize the mixture parameters by applying the modified algorithm of (Lawrence and Scholkopf, 2001) to each of the components
5. Map updated clusters to classes
6. Create a Bayes classifier

ponents must be calculated. In (Li et al., 2006) the latter is determined as the K value that produces the highest total log-likelihood on the test set. In their paper, more details on alternative techniques for estimating the number of mixtures can be found.

No-random imperfect supervision. The imperfect supervision can be *no-random* in the case where the noise is not uniform in the features space. In other words, the noise depends on the feature vector. In the context of medical diagnosis (if we want to compare with the example given in the previous section) a *no-random* imperfect supervision can occur when humans label a patient disease by its symptoms. As in the case of random imperfect supervision, different techniques have been proposed in order to deal with this kind of data. (Lachenbrunch, 1974) presented such a technique, by calculating the conditional error rates using Monte Carlo methods. Also, (Titterington, 1989) used an `EM` algorithm in order to estimate the parameters of a logistic-normal distribution.

(Ambroise and Govaert, 2000) proposed an `EM` algorithm which estimates the posterior distribution of the true label class with respect to the incomplete data. They are based on the concept of Maximum Likelihood Estimators (MLE) computed on observations which may be either labeled, unlabeled or partially labeled. The idea is to introduce a distribution which indicates the subset of classes an example could belong, including the true class. In other words, it tries to identically distribute its doubts about the label of an example in the other possibles classes. They compared this method with the transferable belief model (TBM), first introduced by (Denoeux, 1995): the latter is a non-probabilistic approach, based on the "Dempster-Shafer" theory (Smets, 1994), and can also handle mislabeling data.

2 .3.1 Semi-Supervised learning with mislabeled data

The methods we presented above were all proposed in the context of supervised learning. (Amini and Gallinari, 2003) introduced another method which takes into account the mislabelings, but in addition performs semi-supervised learning. Their method could be placed in the *random imperfect supervision* framework, as the mislabeling of an example does not depend on its feature vector. In contrast with the other methods of this framework, this method does not assume that the label errors come from the manual labeling of the data. Instead, it assumes that the mislabeling errors occur by the classification algorithm itself and it uses the label error model to correct them. Their algorithm is based on the Logistic-CEM first introduced in (Vittaut et al., 2002). The idea is to incorporate in the latter a mislabeling error model. The algorithm is first trained on the labeled part of the training set and it iteratively assigns class labels to unlabeled training examples. These newly labeled examples, together with the labeled part of the training set, are then used to re-train the classifier. At each iteration, the semi-supervised learning system is acting as an imperfect supervisor on unlabeled training examples.

In order to model the mislabeling errors, supposing there is a set of n labeled and a set of m unlabeled examples, they introduced the following

probability, where y_i and $\tilde{y}_i$ are the perfect (i.e. the real but unknown) and the imperfect (i.e. the predicted by the classifier) labels of the unlabeled example $x_i \in \mathcal{X}_u$:

$$\forall k, \forall h, \alpha_{kh} = p(\tilde{y} = k | y = h)$$

subject to the constraints:

$$\forall h, \sum_k \alpha_{kh} = 1$$

They assume that the density of an example, given its true label, does not depend on its imperfect label:

$$p(x_i | \tilde{y} = k, y = k) = p(x_i | y = h)$$

In order to train their model, they use the CEM[2] algorithm. The latter tries to maximize the following log-likelihood:

$$\begin{aligned} L_c &= \sum_{i=1}^{n} \sum_{k=1}^{c} t_{ki} \log P(y = k | x_i, \beta) \\ &+ \sum_{i=n+1}^{n+m} \sum_{k=1}^{c} \left[\tilde{t}_{ki} \log \left(\sum_{h=1}^{c} \alpha_{kh} P(\tilde{y} = h | x_i, \beta) \right) \right] \end{aligned} \quad (2\ .13)$$

where $t_i = \{t_{ki}\}_k$ is the indicator vector class associated with the labeled examples x_i:

$$\forall i \in \{1, \ldots, n\}, \forall y_i = k \Leftrightarrow t_{ki} = 1 \text{ and } \forall h \neq k, t_{hi} = 0$$

The $\tilde{t}_{kj}$ corresponds to the respective indicator vector class, based on the estimated labels $\tilde{y}_j$ for the unlabeled examples x_j. The parameters β are the parameters of the logistic classifier.

[2]CEM refers to *Classification EM*. The latter was introduced by (Symons, 1981) and was applied to semi-supervised learning by (McLachlan, 1992). The idea is to introduce an additional step (`C-step`) in the `EM` algorithm. During this step, each of the examples is assigned to the most likely class.

Algorithm 10: SSL with an explicit label-error model for misclassified data

Input : A partially labeled dataset $\mathcal{X} = \mathcal{X}_l \cup \mathcal{X}_u$
A logistic classifier f

Initialize:

- Train f with the labeled examples.
  We denote with $\beta^{(0)}$ the obtained parameters
- Initialize $\alpha^{(0)}$ by random
- $j \leftarrow 0$

repeat

- `C-step`: Estimate the imperfect class posterior probabilities using the output of the classifier, and get an imperfect label for each $x_i \in \mathcal{X}_u$:

$$\forall x_i \in \mathcal{X}_u,\ \tilde{y}_i^{(j+1)} = \underset{k}{\text{argmax}} \sum_{h=1}^{c} \alpha_{kh}^{(j)} p(\tilde{y}^{(j)} = h | x_i)$$

  Let $\pi^{(j+1)}$ be the new partition obtained from this classifier for the unlabeled data

- `M-step`: Estimate the new parameters $\beta^{(j+1)}$, $\alpha^{(j+1)}$ which maximize $L_c(\pi^{(j+1)}, \beta^{(j)}, \alpha^{(j)})$ (eq. 2 .13):
  - $\beta^{(j+1)} = \underset{\beta^{(j)}}{\text{argmax}} L_c(\pi^{(j+1)}, \beta^{(j)}, \alpha^{(j)})$
  - Find the parameters $\alpha^{(j+1)}$ which maximize $L_c(\pi^{(j+1)}, \beta^{(j+1)}, \alpha^{(j)})$,
    subject to constraints $\forall k, \forall h, \alpha_{kh}^{(j+1)} \in [0, 1]$
    and $\forall h, \sum_k \alpha_{kh}^{(j+1)} = 1$
- $j \leftarrow j + 1$

until *convergence of* L_c ;

Output : The labels of the examples $x \in \mathcal{X}_u$

In algorithm (10) the different steps of the method are described. The `E-step` does not appear explicitly in the algorithm, as it is trivial, since the posterior estimates are given by the classifier outputs directly.

Additional methods. It is worth mentioning that in the literature there exist some methods which try to solve the problem of noisy data from a different perspective. Istead of modeling the mislabeling errors, they try to "clean" the data, by finding and removing the mislabeled data. In other words, they employ some preprocessing mechanisms to handle noisy instances before a learner is formed. These filtering techniques (as they are known) usually result in a reduced training set. Such methods include (Brodley and Friedl, 1999) who used cross-validation over the training data to find mislabeled instances. Also, (John, 1995) tried removing the training instances that are pruned by the C4.5 algorithm (Quinlan, 1993). In each iteration the tree was rebuilt from the filtered set of training instances. This procedure was repeated until no further pruning could be done. (Van Hulse et al., 2007) introduced an approach (called Pairwise Attribute Noise Detection Algorithm (PANDA)) which tries to identify the most noisy examples. Due to the potential risk of data cleaning when noisy examples are retained while good examples are removed, in which cases the reduced training set can be much less accurate than the full training set, efforts have been taken to construct noise tolerant classifiers directly.

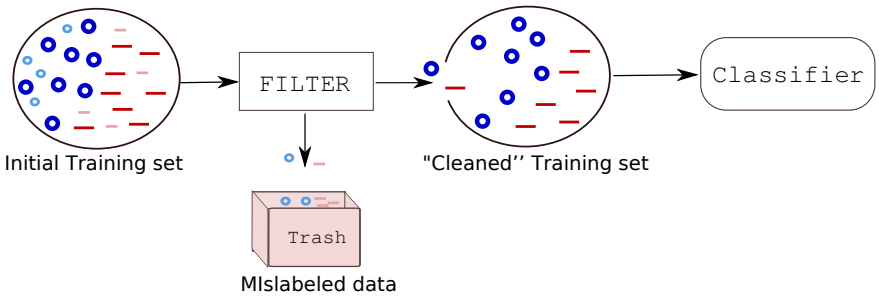

Figure 2 .6: Filtering Techniques: They try to clean the dataset by removing the mislabeling data (the small dots and lines) and keeping only the correct labeled examples (big circles and lines)

2 .4 Conclusion

In this chapter we have first introduced the idea of semi-supervised learning and the motivation behind this concept. The huge amount of available unlabeled data and the cost of labeling these examples have led to the design of algorithms which try to take advantage of both labeled the unlabeled data. Several of these methods have been inspired from techniques presented first in the statistical community. Before presenting the different families of semi-supervised learning techniques, we have discussed the assumptions in which they rely on. A distinction between transductive and semi-supervised learning has been made and a synthesis of different methods in each of the two approaches has been presented.

In the second part of this chapter, we discussed the presence of noise in a training dataset. Very often there are mislabelings in the training data and that can lead to a decrease of the performance. This is why different methods have been introduced in the literature, which try to deal with this problem.

The goal was not to present an exhaustive list of all existing methods. Instead, the concepts of semi-supervised learning and mislabeling error models have been presented. Some representative methods, together with their motivation, from each of the different frameworks have been discussed.

3

Active Learning

Contents

3 .1 Introduction 45

3 .2 Active Learning Techniques 47

3 .2.1 Certainty-based sampling 47

3 .2.2 Query By Committee 48

3 .2.3 Expected error minimization 50

3 .3 Theoretical views of Active learning 51

3 .4 Combining SSL and Active Learning 53

3 .5 Conclusion 58

3 .1 Introduction

Active learning (AL), as semi-supervised learning, addresses the issue of annotation cost. In contrast with semi-supervised learning which, as mentioned in the previous chapter, uses the unlabeled data in addition with the labeled ones, active learning suggests to choose the most informative examples among the unlabeled ones to annotate, in order to obtain better performance than, if unlabeled examples are not taken into account in the learning process or if they are labeled at random. This form of active learning is also known as

selective sampling (Cohn et al., 1994).

The typical active learning setting consists of a partially labeled dataset $\mathcal{X} = \mathcal{X}_l \cup \mathcal{X}_u$, a classifier f and a query module q. The classifier is initially trained with the labeled subset $\mathcal{X}_l$. Then, using the query module q, it chooses the unlabeled examples $\mathcal{X}'_u \subset \mathcal{X}_u$ which can bring more information in our classifier. These examples are labeled by the user and the classifier is retrained with the enriched labeled set $\mathcal{X}_l \cup \mathcal{X}'_u$ (figure 3 .1). The measure of efficiency for an active learner can be either the reduction of the labeled set size needed to achieve a certain performance, or the performance achieved for a certain size of the labeled set.

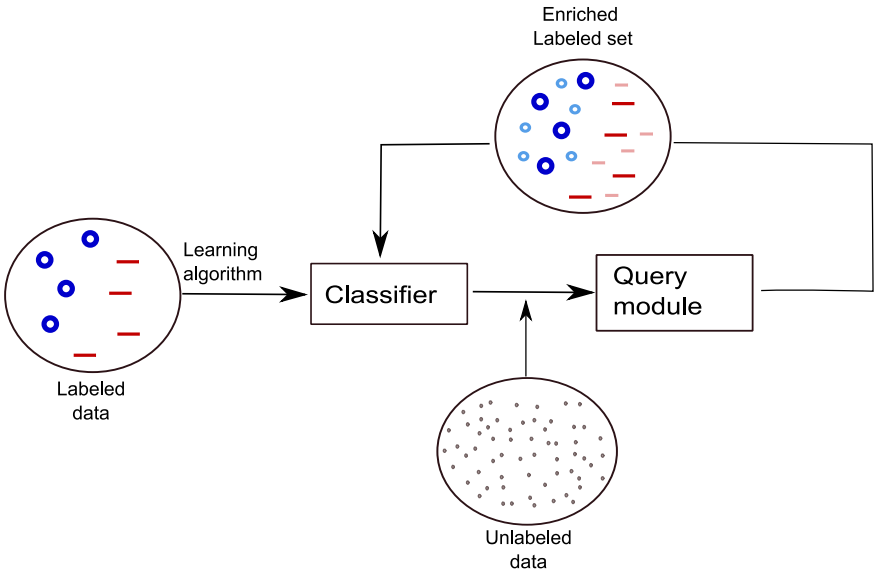

Figure 3 .1: Active learning: The classifier is trained with the labeled examples. Then, using a selection strategy, it chooses and demand the annotation of the most informative examples among the unlabeled ones. This procedure continues until a certain performance or a certain size of the labeled set

Active learning has been applied in various tasks. For example, (Schohn and Cohn, 2000) presented very good results for text classification using ac-

tive learning. (Vogiatzis and Tsapatsoulis, 2008) proposed an active learning method in the field of bioinformatics and they presented results on DNA microarray data sets. (Kuo et al., 2008) presented an adaptive learning framework for Phonetic Similarity Modeling (PSM) that supports the automatic construction of transliteration lexicons. (Cooper et al., 2007) used active learning to identify which motion sequence the user should perform next, in order to improve the quality and responsiveness of a kinematic character controller. (Hakkani-Tür et al., 2006) applied active learning in the task of spoken language understanding.(Chawla and Bowyer, 2007) proposed a learning framework for the face recognition task. They proposed to actively learn the face space in order to achieve a good performance using just a subset of the training set.

3 .2 Active Learning Techniques

The existing active learning algorithms can be placed in three main categories: the *certainty-based sampling*, the *query by committee* and the *expected error minimization*. In the following sections the motivation of these categories are discussed and some well known methods are presented.

3 .2.1 Certainty-based sampling

Certainty-based sampling is based on the confidence of the current classifier on unlabeled data. This method was first introduced by (Lewis and Gale, 1994), where a probabilistic classifier is used (Naive Bayes), which assigns all possible labels to the unlabeled data with certain probabilities. Then, based on these probabilities, the most ambiguous examples are chosen for annotation, i.e. the examples with the highest entropy (high entropy suggests high uncertainty for an example).

In (Tong and Koller, 2000) a similar idea is presented. In their method, the uncertainty of the unlabeled data is measured as the closeness to the decision boundary of an SVM classifier. Also, (Campbell et al., 2000) pro-

posed an algorithm for the training of support vector machines using instance selection. In each iteration the example which is the closest to the current hyperplane of the SVM algorithm is chosen.

In the same vein, (Ertekin et al., 2007) proposed an active learning method which selects informative examples from a randomly picked small pool of examples rather than making a full search in the entire training set. That way, the active learning method can be applicable to very large datasets. They used the SVM algorithm, but instead of using a traditional SVM solvers (e.g. SVM^{light} (Joachims, 1999)), they used an online SVM algorithm, LASVM (Bordes et al., 2005). LASVM works in an online setting, where its model is continually modified as it processes the training examples one by one. The proposed active learning method selects the examples closest to the margin as in (Tong and Koller, 2000; Campbell et al., 2000). They used the proposed active learning strategy in order to address the class imbalance problem, and they presented some encouraging results. The intuition is that we can achieve more balanced class distributions in the earlier steps of the learning, if we focus the learning on the examples around the classification boundary.

3 .2.2 Query By Committee

A second type of active learning which is met in the literature is the *query by committee* (QBC). It was first introduced by (Seung et al., 1992; Freund et al., 1997). The idea here is to measure the agreement among a committee of classifiers. The classifiers are trained with the labeled data and they classify then the unlabeled examples. The algorithm chooses the examples with the biggest disagreement among the classifiers. These examples are annotated by the user and they are incorporated in the labeled examples. The intuition behind this method is that if different classifiers disagree about the label of an example, it means that the later is difficult to label. Here we must note that in order this method to be efficient, the results of the classifiers should not be correlated.

(Muslea et al., 2000) presented the *co-Testing* algorithm which can be applied in multi-view tasks, that is the tasks where there are more than one signal to describe observations (like in co-training, described in the previous chapter). The idea is to use different classifiers for the different views of the data and to query the unlabeled examples on which the views predict different labels.

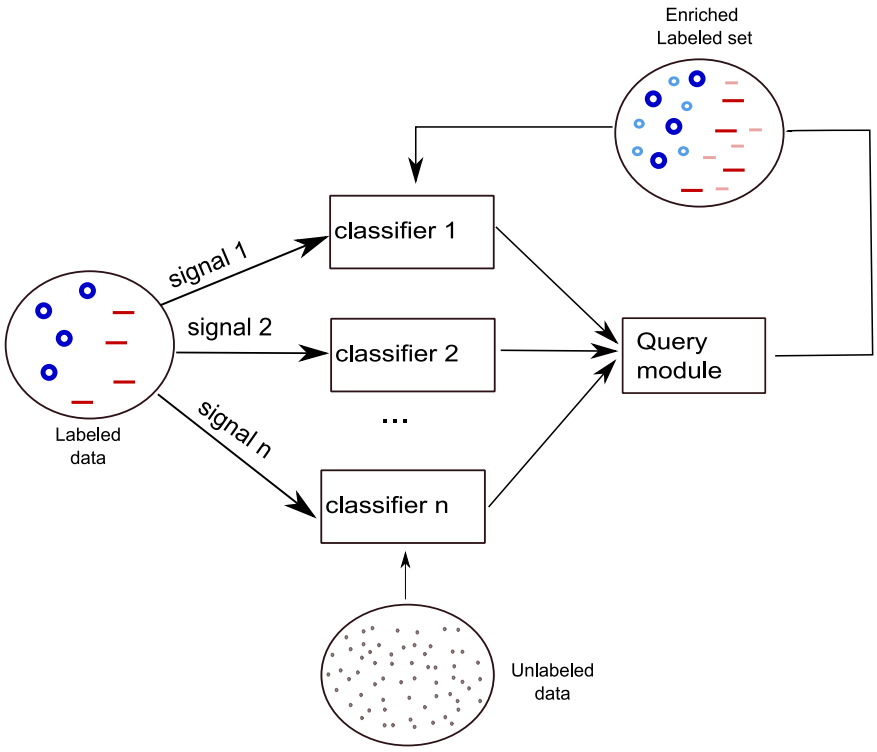

Figure 3 .2: Query By Committee. The active learner chooses the example with the biggest disagreement among the different classifiers

(Dagan and Engelson, 1995) presented a general committee-based active learning method for selective sampling, which is appicable to probabilistic classifiers. In their work, they focused on the task of tagging, where an example is a word sequence and each word w is labeled with a tag t by each committee member. In order to quantify the committee disagreement for a

word, they use the vote entropy defined as:

$$VE(w) = -\sum_{t} \frac{V(t,w)}{k} \log \frac{V(t,w)}{k}$$

where $V(t,w)$ is the number of committee members (out of k members) voting for tag t for the word w. The vote entropy can be seen as a measure of classification uncertainty based on the training data. They then measure the disagreement over an a word sequence by averaging the voting entropy of all words in the sequence. They applied their method to training Hidden Markov Models (HMM) (Rabiner, 1990).

(Davy and Luz, 2007) proposes the *History Kullback-Leibler Divergence (HKLD)* algorithm. The idea is to incorporate the predictions made in previous iteration of active learning into the selection of informative unlabelled examples. The past k predictions, of the previous k iterations, can be thought of as the output of a committee of size k. In this context we can measure uncertainty as the disagreement among committee members using *Kullback-Leibler divergence to the mean* (McCallum and Nigam, 1998). KL divergence to the mean is an average of the KL divergence between each distribution and the mean of all the distributions.

3 .2.3 Expected error minimization

The third type of active learning algorithms tries to *minimize the expected error* (for example (Iyengar et al., 2000)). According to this paradigm, the unlabeled data which reduce the expected classification error are chosen for annotation. This last type of active learning methods is the most sophisticated, as it is based on a statistically optimal solution. The idea is to consider each of the unlabeled examples as the next query. Then the reduction of the classification error is calculated. The unlabeled data with the largest estimated reduction is asked to be annotated by the system. For example, in (Roy and McCallum, 2001) a sample estimation method is used for the Naive

Bayes classifier. The idea is to train the classifier using the current labeled examples and then produce an estimated output distribution $\tilde{P}_D(y|x)$ for the unlabeled examples which are candidates (as the true output distribution is unknown). The best candidate is the one for which the knowledge of the true label will cause the largest reduction of the risk (expected loss). Using this estimated distribution, they calculate the expected loss for an candidate unlabeled example x^* by either a log loss (as the real labels are not known we use the estimated ones):

$$\tilde{E}_{\tilde{P}_{D^*}} = \frac{1}{|\mathcal{X}|} \sum_{x \in \mathcal{X}} \sum_{y \in Y} \tilde{P}_{D^*}(y|x) \log(\tilde{P}_{D^*}(y|x)) \quad (3.1)$$

or a 0/1 loss:

$$\tilde{E}_{\tilde{P}_{D^*}} = \frac{1}{|\mathcal{X}|} \sum_{x \in \mathcal{X}} \left(1 - \max_{y \in Y} \tilde{P}_{D^*}(y|x)\right) \quad (3.2)$$

where $D^* = D + (x^*, y^*)$. Of course, before making the query, the true label of x^* is also unknown. Again, the current learned classifier gives an estimate of the distribution $\tilde{P}_D(y|x^*)$ from which the estimated label of the x^* would be chosen. The latter is used in order to calculate the estimated error for each possible label $y \in Y$. Algorithm (11) presents the above method.

Also (Dönmez et al., 2007) proposed a similar approach (the so-called "dual") where the strategy selection parameters are adaptively updated based on estimated future residual error reduction after each actively sampled point.

3 .3 Theoretical views of Active learning

Except the different techniques of active learning presented in the previous sections, some effort has been made in a theoretical basis, and some interresting works have appeared in the literature.

In this context, (Cohn et al., 1995) considered the problem of actively selecing examples as a the statistically optimal manner. They studied two

Algorithm 11: Active Learning by expected Error Minimization (Roy and McCallum, 2001)

1. train a classifier using the current labeled examples $\mathcal{X}$
   - consider each unlabeled example x in the pool as a candidate for the next labeling request
     - consider each possible label y for x and add the pair (x, y) to the training set
     - re-train the classifier with the enlarged training set $\mathcal{X} + (x, y)$
     - estimate the resulting expected loss using equation (3 .1 or 3 .2)
   - Assign to x the average expected losses for each possible labeling y weighted according to the current classifier's posterior, $\tilde{P}_D(y|x)$
2. Select for labeling the unlabeled example x that generated the lowest expected error on all other examples.

well known statistical models, Mixtures of Gaussians and Locally Weighted Regression and they derived a greedy optimality criterion for the selection of examples.

More recently, (Castro and Nowak, 2007) tried to come up with some limits in active learning. Using minimax analysis techniques, they achieved some bounds under which one can expect significant gains through active learning.

(Hanneke, 2007b) studied the label complexity of pool-based active learning in the PAC model with noise. They derived upper and lower bounds on the label complexity in terms of generalizations of extended teaching dimension. They claimed that their bound is the first nontrivial general upper bound on label complexity in the presence of persistent classification noise.

(Balcan et al., 2006) presented an algorithm, the so-called *Agnostic Active learning* or A^2 *learning* (which is essentially the active learning algorithm of

(Cohn et al., 1994)), and they provided a label-complexity upper bound for learning linear separators under the uniform input distribution. (Hanneke, 2007a) extended this work by deriving a general bound on the number of label requests, applicable to any concept space and distribution. Also, in (Balcan et al., 2007a) presented a framework for margin based active learning of linear separators.

In (Balcan et al., 2007b) the problem is considered from a different angle and the asymptotic complexity of active learning is analyzed. They proved that in many interesting cases active learning does help asymptotically.

(Krause and Guestrin, 2007) came up with a theoretical bound on how much better a sequential algorithm can perform than an a priori design strategies. They considered Gaussian Processes (GPs) with unknown parameters and they presented some bounds which motivate the switch between exploration and exploitation approaches to active learning. They extended their algorithm to handle nonstationary Gaussian Processes, exploiting local structure in the model.

3 .4 Combining SSL and Active Learning

The idea of combining active and semi-supervised learning was first introduced by (McCallum and Nigam, 1998). The idea is to integrate an `EM` algorithm with unlabeled data into an active learning framework, and more particularly in a query by committee (QBC) method. The commitee members are created by sampling classifiers according to the distribution of classifier parameters specified by the training data (algorithm 12).

In (Muslea et al., 2002), *Co-EMT* is proposed. This algorithm combines Co-Testing and Co-`EM`. As opposed to Co-Testing algorithm, which learns hypotheses h_1 and h_2 based only on the labeled examples, Co-EMT learns the two hypotheses by running Co-`EM` on both labeled and unlabeled examples. Then, in the active learning step, it annotates the example on which the pre-

dictions of h_1 and h_2 are the most divergent, that is, the example for which h_1 and h_2 have an equally strong confidence at predicting a different label.

Algorithm 12: Combining active learning and semi-supervised learning using `EM` (McCallum and Nigam, 1998)

Input : The labeled and unlabeled training documents
while *more labeled data are required* **do**
  Build an initial estimate of the model parameters from the labeled documents only
  for *each of the k committee members (i.e. for each classifier)* **do**
    - Create a committee member by sampling a classifier for each class from the appropriate Dirichlet distribution
    - Starting with the sampled classifier apply `EM` with the unlabeled data:
    - **repeat**
      - Use the current classifier to probabilistically label the unlabeled documents
      - Recalculate the classifier parameters given the probabilistically-weighted labels

      until *parameters convergence* ;
    - Use the current classifier to probabilistically label all unlabeled documents

  end
  Calculate the disagreement for each unlabeled document, multiply by its density, and request the class label for the one with the highest score.
end
Output : The new labeled set and all the predicted labels

(Zhu et al., 2003b) also present a combination of semi-supervised and active learning using Gaussian fields and harmonic functions (the semi-supervised method is described analytically in the previous chapter 2 .2.1.2). In brief, under this semi-supervised framework, the expected generalization error after

querying a point is calculated, and the one which gives the largest reduction is chosen for annotation (algorithm 13). The estimated risk $\widehat{R}(f)$ can be calculated as follows (here, the f_i values are considered as "proxy" for the class probabilities:

$$\begin{aligned}\widehat{R}(f) &= \sum_{i=1}^{n} [sign(f_i - 0.5) \neq -1]\,(1 - f_i) + [sign(f_i - 0.5) \neq 1]\, f_i \\ &= \sum_{i=1}^{n} min(f_i, 1 - f_i)\end{aligned}$$

We then retrain the classifier on the new labeled training set (augmented by the annotated unlabeled example). If we denote by $f^{+(x_k,y_k)}$ the new harmonic function, the estimated risk becomes:

$$\widehat{R}(f^{+(x_k,y_k)}) = \sum_{i=1}^{n} min(f_i^{+(x_k,y_k)}, 1 - f_i^{+(x_k,y_k)})$$

As we do not know the value of y_k, in order to estimate the expected risk, after querying the example k, we use the following equation:

$$\widehat{R}(f^{+x_k}) = (1 - f_k)\widehat{R}(f^{+(x_k,-1)}) + f_k\widehat{R}(f^{+(x_k,1)})$$

In each iteration, we choose the next example k that minimizes the expected estimated risk:

$$k = argmin_{k'}\widehat{R}(f^{+x_{k'}}) \qquad (3\ .3)$$

(Zhou et al., 2006) presented the so-called method *Semi-Supervised Active Image Retrieval* (*SSAIR*) for a different task of relevance feedback. The method was inspired by co-training (Blum and Mitchell, 1998) and co-testing (Muslea et al., 2000), but instead of using two sufficient but redundant views of the dataset, it employs two different learners on the same data. Initially, the two learners are trained on the labeled data. Then, each of them ranks the unlabeled data by giving them a value between $\{-1, 1\}$, where negative and positive indicates whether the learner believes that the example is irrelevant or relevant resectively. The bigger the absolute value is, the more

Algorithm 13: Combining active learning and semi-supervised learning using Gaussian fields and harmonic functions

Input :

- A graph $G = (V, E)$ and weight matrix $\mathbf{W}$
- The labels of the labeled examples f_l
- The diagonal matrix $D_{ij} = \sum_k \mathbf{W}_{ik}$
- The combinatorial Laplacian matrix $L = D - \mathbf{W}$.
  We split the matrix according to labeled and unlabeled examples as: $L = \begin{bmatrix} L_{ll} & L_{lu} \\ L_{ul} & L_{uu} \end{bmatrix}$

while *more labeled data are required* **do**

- Compute harmonic function $f_u = -L_{uu}^{-1} * L_{ul} * f_l$
- Find best example to annotate, using equation (3 .3)
- Query point x_k, and receive answer y_k
- Add (x_k, y_k) in $\mathcal{X}_l$, and remove x_k from $\mathcal{X}_u$

end

Output :

- The new labeled set and the classifier f

confident the learner is about its decision. According to this ranking each learner passes the most relevant images to the other one. After re-training with the additional labeled data, the learners rank the data again and then their rankings are merged by summation, which gives the final ranking for the unlabeled data. The examples for which the learners are confident to be relevant are returned as the retrieval result. The ones which have low confidence are placed into the pool which is used in the next round of relevance feedback.

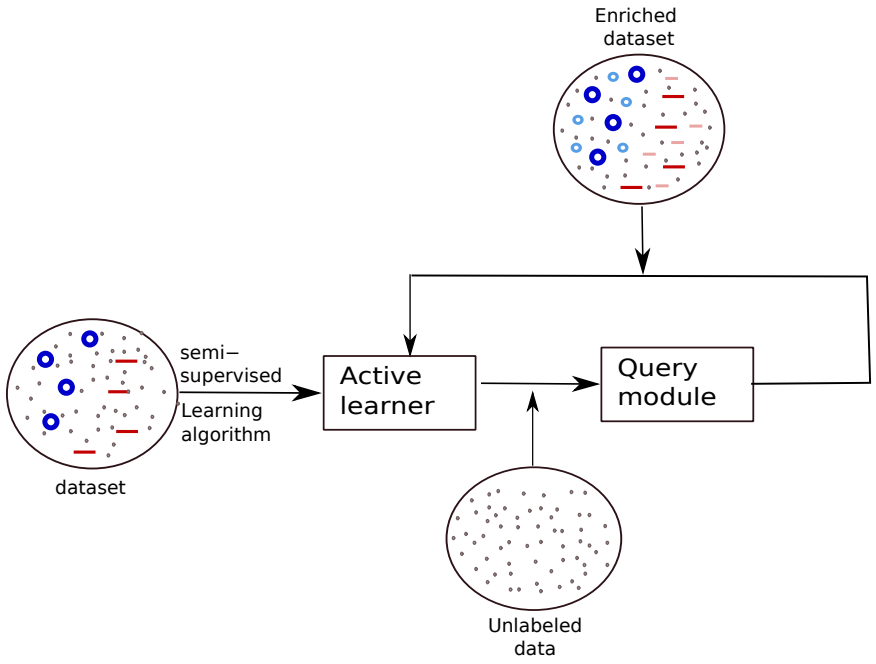

Figure 3 .3: Combining semi-supervised and Active learning

In the context of multi-view active learning, (Probst and Ghani, 2007) proposed a method which combines semi-supervised and active learning. The first step uses co-EM with naive Bayes as the semi-supervised algorithm. They present an approximation to co-EM with naive Bayes that can incorporate user feedback almost instantly and can use any sample-selection strategy for active learning.

Why the combination should work? Intuitively, the combination of both semi-supervised and active learning appears to be particularly beneficial in reducing the annotation burden. Semi-supervised learning is more focused on *exploitation*, while active learning is more dedicated to *exploration* of the data. As a result, used alone, it can lead to poor performance in certain cases, as semi-supervised strongly suffers from poorly represented classes, while being very sensitive to noise. On the other hand, active learning could be too slow, as it does not really exploit the information given by unlabeled

data. In the same vein, semi-supervised learning tends to over-weight easy-to-classify examples that will dominate the process, while active learning has the opposite strategy, resulting in exploring more deeply the hard-to-classify examples. Also, active learning based on the confidence scores calculated on the whole dataset and not only on the labeled examples, can be expected to be more efficient. The reason is that the confidence score will be more accurate based on both label and unlabeled data.

3 .5 Conclusion

In this chapter, we presented the existing methods in active learning. The latter, as semi-supervised learning, tries to reduce the annotation burden. The general idea of the latter is to annotate actively the most informative examples in order to ameliorate the performance of the classifier. We have presented the main types of active learners and representative algorithms in each of them. In the last section, the combination with semi-supervised learning has been discussed. The reasons that the combination is interesting and promising have been demonstrated, together with some works towards this direction.

II

Active and Semi-Supervised Aspect Models

In the first part of this thesis, we have presented three frameworks that aim to reduce the annotation burden and to model possible mislabelings in a training set. In particular, we first presented semi-supervised paradigm and mislabeling error models. Then, we focused on the effort that has been made to combine these two frameworks. Finally, we presented the paradigm of active learning and the combination of the latter with semi-supervised learning.

In this second part, we combine these three frameworks. We are focusing on the task of text categorization and we present an extension of the aspect models to the case of semi-supervised learning for this task. This study is motivated by the cost of labeling document collections and the ability of aspect models to explain the generation of textual observations. In this part, we propose two semi-supervised variants of aspect models, especially of the PLSA algorithm, which incorporate a mislabeling error model. We further extend these semi-supervised models by combining them with two different active learning strategies.

4

Semi-Supervised Aspect Models

Contents

4 .1 Introduction 63
4 .2 Aspect Models for Document Classification . 64
4 .3 Probabilistic Latent Semantic Analysis 65
4 .4 ssPLSA with a "missing values" model 68
4 .5 ssPLSA with a fake label model 70
4 .6 ssPLSA-mislabeling with hard clustering . . . 73
4 .7 ssPLSA-mislabeling with soft clustering . . . 78
4 .8 Conclusion 81

4 .1 Introduction

In this chapter, we start by presenting the framework of *aspect models*. Then, we describe the Probabilistic Latent Semantic Analysis (PLSA) algorithm introduced by (Hofmann, 2001). We continue by introducing our three semi-supervised variants of the PLSA model.

The motivation of the first variant is to try to handle the uncertainty posed by the unlabeled data clusters. In the second and third variants, we

try to capture the possible mislabeled data which occur during the training of our model. The idea is to iteratively assign class labels to unlabeled examples and estimate the probabilities of the mislabeling errors. These probabilities are taken into account in the estimation of the new model parameters before the next round. In the third variant, as opposed to the second one, we perform soft clustering on the unlabeled data.

4 .2 Aspect Models for Document Classification

As we saw in chapter 2, semi-supervised methods relying on a generative model usually implement a local independance assumption (similar to the *Naive Bayes* assumption), which is unlikely to be met in practice. In addition, some simple models (such as the Naive Bayes *model*) assume that an observation is generated in its entirety from the class it belongs to. This makes it inconvenient to model data that may comprise several aspects, for example textual documents which potentially cover different topics. This has led to the development of *Aspect Models* (Hofmann, 2001), which can take into account such data with multiple facets. The aspect models differ in the statistical assumptions they impose on the model: They are based on the assumption that examples cover one or more different topics. In other words, an example can be modeled as a mixture of topics. They specify a simple probabilistic procedure by which theses examples can be generated. In this way, examples are now characterized in terms of topics instead of simple features. Observations are generated by a mixture of aspects, or topics, each of which being a distribution over the basic features of the observations (such as words in a document, or pixels in an image etc). Interestingly, these models allow to capture interesting application-dependent phenomena. When modeling textual content, for example, they take into account linguistics properties[3] such as synonymy (different terms with the same meaning) and polysemy (different meanings of the same term). Both may have a crucial influence on the modelling of the relationship between documents.

[3]Further details on these linguistics properties are given in the evaluation chapter, where the representation of the data is discussed

4 .3 Probabilistic Latent Semantic Analysis

One of the first aspect models introduced in the literature, is the *Probabilistic Latent Semantic Analysis* (PLSA), proposed by Hofmann (Hofmann, 2001). The latter has been presented in terms of document classification, but it can be applicable to different tasks, such as image classification. It has been presented as a probabilistic version of the Latent Semantic Analysis (LSA) method (Deerwester et al., 1990).

PLSA is a probabilistic model which characterizes each word in a document as a sample from a mixture model, where mixture components are conditionally-independent multinomial distributions. This model, also known as the aspect model (Saul and Pereira, 1997), associates an unobserved latent variable (called aspect, topic or component) $\alpha \in \{\alpha_1, ..., \alpha_A\}$ to each observation corresponding to the occurrence of a word $w \in \mathcal{W}$ within a document $x \in \mathcal{X}$. One component or topic can coincide with one class or, in another setting, a class can be associated to more than one component. Although originally proposed in an unsupervised setting, this latent variable model is easily extended to classification with the following underlying generation process:

- Pick a document x with probability $P(x)$,
- Choose a latent variable α according to its conditional probability $P(\alpha \mid x)$
- Generate a feature w with probability $P(w \mid \alpha)$
- Generate the example's class y according to the probability $P(y \mid \alpha)$.

The probability $P(y \mid \alpha)$ is fixed, by forcing to zero the component α that do not belong to a certain class y, i.e. $P(y|\alpha) = \begin{cases} 1, \text{ if } \alpha \in y \\ 0, \text{ otherwise} \end{cases}$ (as we know how many components per class we have).

Hence, the model parameters are

$$\Xi = \{P(\alpha \mid x), P(w \mid \alpha) : \alpha \in A, x \in \mathcal{X}, w \in \mathcal{W}\}$$

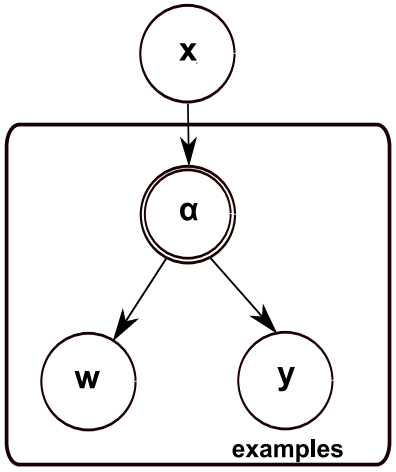

Figure 4 .1: Graphical model representation of the PLSA model. Latent variables are double circled.

The generation of a feature w within an example x can then be translated by the following joint probability model:

$$P(w, x) = P(x) \sum_{\alpha \in A} P(w \mid \alpha) P(\alpha \mid x) \tag{4 .1}$$

So, the log-likelihood of the model can be estimated as:

$$\mathcal{L} = \sum_{w \in \mathcal{W}} \sum_{x \in \mathcal{X}} n(x, w) \log P(x, w) \tag{4 .2}$$

where $n(w, x)$ denotes the frequency of the word w in instance x. At this point we have to note that y appears in the log-likelihood indirectly, through α.

In algorithm (14) the training of this model is described. The idea is to perform one clustering per class by fixing the number of components per class. The latter is done during the initialization of the model ($\Xi^{(0)}$), where we force to zero the $P(\alpha \mid x)$ for an example x which does not belong to a particular topic α (that is the labeled training examples). The algorithm used ensure

to maintain the forced zeros during the iterations. The $P(w \mid \alpha)$ is initialized by giving random values for all w and α.

After the training of the model, we run the PLSA for the test set,(algorithm (15)) using the calculated model (i.e. $P(w|\alpha)$), in order to learn the $P(\alpha|x)$ and we classify the examples of the test set with the maximum posterior probability using chain rule:

$$P(y|x) \propto \sum_{\alpha} P(\alpha|x)P(y|\alpha)$$

We choose as label for each example, the one with the highest probability.

In chapter 2, we presented (together with its semi-supervised variant) the Naive Bayes model (Lewis, 1998). In the latter, some simplifying assumptions are considered, which PLSA overcomes in two important ways. First, it relaxes the assumption that a class y is associated to a single topic. In PLSA, the number of topics $|A|$ may be larger than the number of classes K. The second and crucial difference is that in Naive Bayes, all features must be generated from the same topic. This requires the use of clever smoothing strategies to counter the fact that some features that are unrelated to a topic may appear by coincidence in an example from that topic. On the other hand, in PLSA, a topic is drawn independently from $P(\alpha \mid x)$ each time a new feature is generated in an example. This provides a much more natural way to handle unusual features or multi-topicality.

The complexity of PLSA is $O(|A| \times M)$, where $M = \#\{(w, x)|n(w, x) > 0\}$, is the number of pairs (w, x) co-occuring at least once in the collection (or, in other words, the number of non-zero entries in the example-feature matrix).

Algorithm 14: Probabilistic Latent Semantic Analysis (PLSA) for Document Classification: Training

Input :

- A labeled set $\mathcal{X} = X_l$,
- Random initial model parameters $\Xi^{(0)}$.
- $j \leftarrow 0$

repeat

- `E-step`: Estimate the latent class posteriors: $\forall x \in \mathcal{X}, \forall w \in \mathcal{W}, \forall \alpha \in A$ of the class $y(x)$

$$P(\alpha|w,x) = \pi_\alpha^{(j)}(w,x) = \frac{P^{(j)}(\alpha|x)P^{(j)}(w|\alpha)}{\sum_{\alpha' \in A} P^{(j)}(\alpha'|x)P^{(j)}(w|\alpha')}$$

- `M-step`: Estimate the new model parameters $\Xi^{(j+1)}$ by maximizing the complete-data log-likelihood:

$$\begin{aligned} P^{(j+1)}(w|\alpha) &\propto \sum_x n(w,x)\pi_\alpha^{(j)}(w,x) \\ P^{(j+1)}(\alpha|x) &\propto \sum_w n(w,x)\pi_\alpha^{(j)}(w,x) \end{aligned}$$

- $j \leftarrow j+1$

until *convergence of the complete-data log-likelihood* ;

Output : A generative classifier with parameters $\Xi^{(j)}$

4 .4 Semi-Supervised PLSA with a "missing values" model

The most straight forward semi-supervised variant of the PLSA algorithm is to treat it as a "missing values" model. The latter can be seen as a clustering with constraints. In this case, our model will be identical with the one presented in the previous section and in algorithm (14). The difference lies

Algorithm 15: Probabilistic Latent Semantic Analysis (PLSA) for Document Classification: Testing

Input :

- A test set $\mathcal{X}'$
- the learned model $P(w|\alpha)$

repeat

- `E-step`: Estimate the latent class posteriors: $\forall x \in \mathcal{X}', \forall w \in \mathcal{W}, \forall \alpha \in A$

$$P(\alpha|w,x) = \pi_{\alpha}^{(j)}(w,x) = \frac{P^{(j)}(\alpha|x)P(w|\alpha)}{\sum_{\alpha' \in A} P^{(j)}(\alpha'|x)P(w|\alpha')}$$

- `M-step`:

$$P^{(j+1)}(\alpha|x) \propto \sum_{w} n(w,x)\pi_{\alpha}^{(j)}(w,x)$$

until *convergence of the complete-data log-likelihood* ;
Calculate the labels of the test set:

$$P(y|x) \propto \sum_{\alpha} P^{(j)}(\alpha|x)P(y|\alpha)$$

Output : The labels of the test set

to the initialization process: instead of forcing to zero the $P(\alpha \mid x)$ for all examples x of the training set which do not belong to a particular topic α, this time we force $P(\alpha \mid x)$ only for the labeled examples and we give random values for all the unlabeled examples of the training set, under the constraint to sum up to 1. This happens because we do not know the labels of the unlabeled examples, and as a result, we cannot find in which topic they belong (by $P(y|\alpha)$), as we do with the labeled data.

This "missing values" model is the simplest way to perform semi-supervised learning with the PLSA model. The drawback is that it does not capture any

of the problems created because of the very low ratio of labeled and unlabeled examples. This is why we will focus on the three semi-supervised variants described in the next sections, which are more sophisticated. The first one (with the "fake label" model) solves the problem of the unlabeled components, while the other two variants (with the mislabeling error model) capture and modelize the mislabelings produced by the classifier during the iterations.

4 .5 Semi-Supervised PLSA with a fake label model

As the aspect PLSA model characterizes the generation of the co-occurrence between a feature w and an example x, for learning the semi-supervised models we have to form two labeled $\mathcal{Q}_l$ and unlabeled $\mathcal{Q}_u$ training sets from X_l and X_u. We consider now each observation as a pair $q = (w, x)$ such that observations in $\mathcal{Q}_l$ are assigned to the same class label as the example x they contain.

We recall that we still characterize the data using a mixture model with $|A|$ latent topic variables α, under the graphical assumption of aspect models (that x and w are independent conditionally to a latent topic variable α). (Krithara et al., 2006), following the work of (Gaussier and Goutte, 2005), presented a semi-supervised variant of PLSA, where additional "fake" labels were introduced for the unlabeled data (namely the *ssPLSA-fake*). The motivation for the latter was to try to solve the problem of the unlabeled components (which contain only unlabeled examples and for which a class assignation is risky). Indeed, the lack of labeled examples in these components can lead to arbitrary class probabilities, and as a result, to arbitrary classification decisions. So all labeled examples in $\mathcal{Q}_l$ (where $\mathcal{Q}_l$ are the co-occurrences for the labeled data) are kept with their real class labels and all unlabeled examples in $\mathcal{Q}_u$ (the co-occurrences for the unlabeled data) are assigned a new *fake* label $y = y_0$.

In this case the model parameters are

$$\Lambda = \{P(\alpha \mid x), P(z = y_0 \mid \alpha), P(w \mid \alpha) : \alpha \in A, x \in \mathcal{X}, w \in \mathcal{W}, z \in \{y + y_0\}\}$$

The above model parameters Λ are obtained by maximizing the complete data log-likelihood

$$\mathcal{L}_1 = \sum_{x \in \mathcal{Q}_l, w \in \mathcal{W}} \log P(w, x, z) \tag{4 .3}$$

using the Expectation-Maximization EM algorithm. In algorithm 16, the training of this model is summarized.

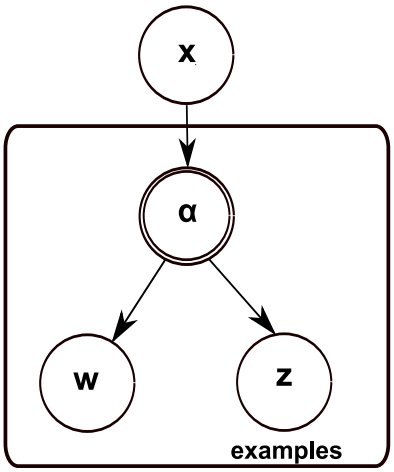

Figure 4 .2: Graphical model representation of the ssPLSA model with a fake label model.$z \in \{y + y_0\}$

At this point we have to note that the values of $P(z|\alpha)$ depend on the value of latent topic variable α. The cardinal of α is given, and in addition, the repartition of the α in different classes is also known, that is, the number of latent topic variables per class. So, in order to initialize, for the true classes (i.e. for all $z \neq y_0$) we force to zero the $P(z|\alpha)$ for the latent topic variables α which do not belong to the particular class z, and for $z = y_0$ we give random values for all α.

The complexity of this algorithm is $O(|A| \times M)$ where, as before, $M = \#\{(w,x)|n(w,x) > 0\}$.

Once the model parameters are obtained, if we want to assign the unlabeled data used in the learning stage to a class (i.e. transductive learning), for each example x, we distribute the probability obtained for the "fake" label $z = y_0$, on the "true" labels using the following equation:

$$P(y|x) \propto \sum_{\alpha} P(\alpha|x) \left[P(y|\alpha) + \lambda P(z{=}y_0|\alpha)\right] \qquad (4\ .4)$$

where $\lambda \in \left[0, \frac{1}{K}\right]$ (where $\lambda = \frac{1}{K}$ corresponds to a uniform repartition of uncertain "fake" label on the other labels) and $y = 1, \ldots, K$.

The *ssPLSA-fake*, can be seen as a confidence measure for each unlabeled example to belong to a given class. That is, after training, unlabeled examples for which the model is more confident, are assigned to one of the real classes. On the other hand, examples for which the classifier has no confidence will keep their fake labels and, from the above equation, their influence will be downweighted.

A new example x is assigned to the class with maximum posterior probability using the same rule as before (equation 4 .4), using the $P(z = y_0|\alpha)$ estimated during the training of the model.

If we want to test our model on new data, after having trained the model, we assign a new example x to the class with maximum posterior probability using the equation (4 .4) above (as in the case of the unlabeled examples in the training set).

Algorithm 16: Semi-Supervised PLSA (ssPLSA) with fake labels

Input :

- A set of partially labeled data $\mathcal{X} = X_l \cup X_u$,
- Random initial model parameters $\Lambda^{(0)}$.
- $j \leftarrow 0$

repeat

- `E-step`: Estimate the latent class posteriors:

$$\forall x \in \mathcal{X}, \forall z \in \tilde{\mathcal{C}} = \mathcal{C} + y_0 \quad , \quad \forall w \in \mathcal{W}, \forall \alpha \in A$$
$$\pi_\alpha^{(j)}(w,x,z) = \frac{P^{(j)}(\alpha|x)P^{(j)}(w|\alpha)P^{(j)}(z|\alpha)}{P^{(j)}(w,z|x)}$$

where $P^{(j)}(w,z|x) = \sum_{\alpha \in A} P^{(j)}(\alpha|x)P^{(j)}(w|\alpha)P^{(j)}(z|\alpha)$

- `M-step`: Estimate the new model parameters $\Lambda^{(j+1)}$ by maximizing the complete-data log-likelihood:

$$P^{(j+1)}(w|\alpha) \propto \sum_x n(w,x)\pi_\alpha^{(j)}(w,x,z(x))$$
$$P^{(j+1)}(\alpha|x) \propto \sum_w n(w,x)\pi_\alpha^{(j)}(w,x,z(x))$$
$$P^{(j+1)}(z|\alpha) \propto \sum_w \sum_{x,z} n(w,x)\pi_\alpha^{(j)}(w,x,z)$$

- $j \leftarrow j+1$

until *convergence of the complete-data log-likelihood* ;

Output : A generative classifier with parameters $\Lambda^{(j)}$

4 .6 Semi-Supervised PLSA with a mislabeling error model - hard clustering

In this section we introduce a semi-supervised variant of the PLSA model in which a misclassification error is incorporated (namely the *ssPLSA-mem*

hard). We assume that the labeling errors made by the generative model for unlabeled data come from a stochastic process and that these errors are inherent to semi-supervised learning algorithms. The idea here is to characterize this stochastic process in order to reduce the labeling errors computed by the classifier for unlabeled data in the training set.

We assume that for each unlabeled example $x \in X_u$, there exists a perfect, true label y, and an imperfect label $\tilde{y}$, estimated by the classifier. Assuming also that the estimated label is dependent on the true one, we can model these labels by the following probabilities:

$$\forall (k,h) \in \mathcal{C} \times \mathcal{C}, \beta_{kh} = P(\tilde{y} = k | y = h) \qquad (4\ .5)$$

subject to the constraint that $\forall h, \sum_k \beta_{kh} = 1$.

In figure 4 .3 below the graphical representation of this model for both labeled and unlabeled data is given.

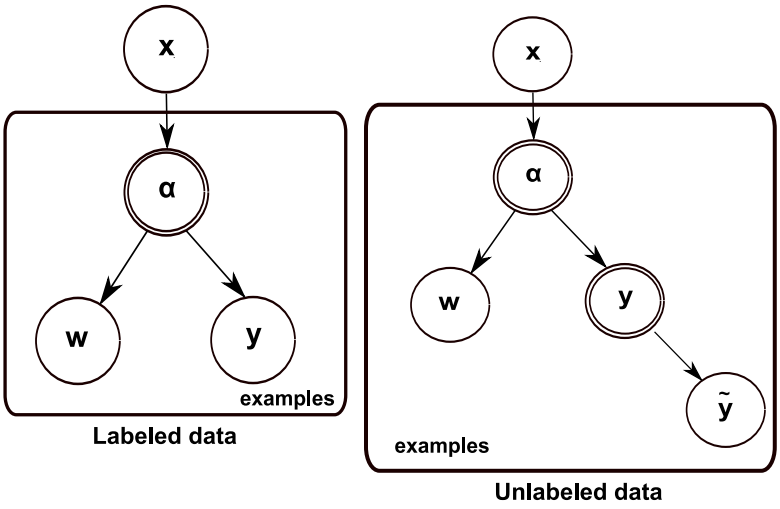

Figure 4 .3: Graphical model representation of the semi-supervised PLSA with a mislabeling error model, for labeled (left) and unlabeled (right) documents.

The underlying generation process associated to this latent variable model for unlabeled data is:

- Pick an example x with probability $P(x)$,
- Choose a latent variable α according to its conditional probability $P(\alpha \mid x)$
- Generate a feature w with probability $P(w \mid \alpha)$
- Generate the *latent* document class y according to the probability $P(y \mid \alpha)$
- The imperfect class label $\tilde{y}$ is generated with probability $\beta_{\tilde{y}|y} = P(\tilde{y} \mid y)$

As in the *ssPLSA-fake* presented in the previous section, the values of $P(y|\alpha)$ depend on the value of the latent topic variable α. The cardinal of α is given (as is considered as a hyper-parameter). The repartition of the α in the classes is also known for both labeled and unlabeled examples. We initialize by forcing to zero the $P(y|\alpha)$ for the latent topic variables α which do not belong to the particular class y. These values remain fixed. In other words, we perform *hard clustering*. We have to note that the hard clustering is done in terms of classes, as an example can be a mix of several components, as far as these components are related to the same class (y).

With this new graphical model, the joint probability between an unlabeled example $q \in \mathcal{Q}_u$ and its imperfect class label estimated by the classifier can be expressed as

$$\forall q \in \mathcal{Q}_u, P(w, x, \tilde{y}) = \sum_{\alpha \in A} P(w|\alpha) P(\alpha|x) \sum_{y \in \mathcal{C}} \beta_{\tilde{y}|y} P(y|\alpha)$$

The model parameters are in this case:

$$\Phi = \{P(\alpha \mid x), P(w \mid \alpha), \beta_{\tilde{y}|y} : x \in \mathcal{X}, w \in \mathcal{W}, \alpha \in A, y \in \mathcal{C}, \tilde{y} \in \mathcal{C}\}$$

and they are estimated by maximizing the complete data log-likelihood:

$$\begin{aligned}\mathcal{L}_2 = &\sum_{x\in X_l}\sum_{w} n(w,x)\log\sum_{\alpha} P(x)P(w|\alpha)P(\alpha|x)P(y|\alpha) \\ &+ \sum_{x\in X_u}\sum_{w} n(w,x)\log\sum_{\alpha} P(x)P(w|\alpha)P(\alpha|x)\sum_{y}\beta_{\tilde{y}|y}P(y|\alpha)\end{aligned} \tag{4 .6}$$

The Maximum likelihood estimates of model parameters are:

$$P^{(j+1)}(w|\alpha) \propto \sum_{x\in X_l} n(w,x)\pi_{\alpha}^{(j)}(w,x,y(x)) + \sum_{x\in X_u} n(w,x)\tilde{\pi}_{\alpha}^{(j)}(w,x,\tilde{y}(x)) \tag{4 .7}$$

where

$$\pi_{\alpha}(w,x,y) = \frac{P(\alpha|x)P(w|\alpha)P(y|\alpha)}{\sum_{\alpha} P(\alpha|x)P(w|\alpha)P(y|\alpha)} \tag{4 .8}$$

and

$$\tilde{\pi}_{\alpha}(w,x,\tilde{y}) = \frac{P(\alpha|x)P(w|\alpha)\sum_{y} P(y|\alpha)\beta_{\tilde{y}|y}}{\sum_{\alpha} P(\alpha|x)P(w|\alpha)\sum_{y} P(y|\alpha)\beta_{\tilde{y}|y}} \tag{4 .9}$$

are the latent topic posteriors for respectively the labeled and unlabeled data.

$$P^{(j+1)}(\alpha|x) \propto \sum_{w} n(w,x) \times \begin{cases} \pi_{\alpha}^{(j)}(w,x,y(x)), \text{ for } x \in X_l \\ \\ \tilde{\pi}_{\alpha}^{(j)}(w,x,\tilde{y}(x)), \text{ for } x \in X_u \end{cases} \tag{4 .10}$$

The mislabeling probabilities are estimated over the unlabeled training set:

$$\beta_{\tilde{y}|y}^{(j+1)} \propto \sum_{w}\sum_{x\in X_u} n(w,x) \sum_{\alpha|\alpha\in y} \tilde{\pi}_{\alpha}^{(j)}(w,x,\tilde{y}) \tag{4 .11}$$

In algorithm 17 the estimation of model parameters Φ is described. This algorithm is also an `EM`-like algorithm as the previous semi-supervised model. For the initialization of the model parameters $\Phi^{(0)}$ are assigned random values by respecting the constraints. Then, at each iteration j during the `E-step` , latent topic posteriors are estimated for labeled and unlabeled data using the current parameters $\Phi^{(j)}$. During the `M-step`, new parameters $\Phi^{(j+1)}$ are es-

timated by maximizing the complete data log-likelihood (equation 4 .6). We alternate these two step until the convergence of the complete data likelihood to a local maximum.

Algorithm 17: ssPLSA-mem hard

Input :

- A set of partially labeled data $\mathcal{X} = X_l \cup X_u$,
- Random initial model parameters $\Phi^{(0)}$.
- $j \leftarrow 0$
- Run a simple PLSA algorithm for the estimation of the initial $\tilde{y}$

repeat

`E-step`: Estimate the latent class posteriors

$$\begin{aligned}
\pi_\alpha(w,x,y) &= \frac{P(\alpha|x)P(w|\alpha)P(y|\alpha)}{\sum_\alpha P(\alpha|x)P(w|\alpha)P(y|\alpha)}, \text{ if } x \in \mathcal{X}_l \\
\tilde{\pi}_\alpha(w,x,\tilde{y}) &= \frac{P(\alpha|x)P(w|\alpha)\sum_y P(y|\alpha)\beta_{\tilde{y}|y}}{\sum_\alpha P(\alpha|x)P(w|\alpha)\sum_y P(y|\alpha)\beta_{\tilde{y}|y}}, \text{ if } x \in \mathcal{X}_u
\end{aligned}$$

`M-step`: Estimate the new model parameters $\Phi^{(j+1)}$
by maximizing the complete-data log-likelihood

$$\begin{aligned}
P^{(j+1)}(w|\alpha) &\propto \sum_{x\in X_l} n(w,x)\pi_\alpha^{(j)}(w,x,y(x)) + \sum_{x\in X_u} n(w,x)\tilde{\pi}_\alpha^{(j)}(w,x,\tilde{y}(x)) \\
P^{(j+1)}(\alpha|x) &\propto \sum_w n(w,x) \times \begin{cases} \pi_\alpha^{(j)}(w,x,y(x)), \text{ for } x \in X_l \\ \tilde{\pi}_\alpha^{(j)}(w,x,\tilde{y}(x)), \text{ for } x \in X_u \end{cases} \\
\beta_{\tilde{y}|y}^{(j+1)} &\propto \sum_w \sum_{x\in X_u} n(w,x) \sum_{\alpha|\alpha\in y} \tilde{\pi}_\alpha^{(j)}(w,x,\tilde{y})
\end{aligned}$$

$j \leftarrow j+1$

until *convergence of the complete-data log-likelihood* ;

Output : A generative classifier with parameters $\Phi^{(j)}$

The complexity of this algorithm is $O(|A| \times M \times \mathcal{C})$, which is comparable to the one of PLSA and *ssPLSA-fake*.

4 .7 Semi-Supervised PLSA with a mislabeling error model - Soft clustering

In this section we present an extention of the previous model, namely the *ssPLSA-mem soft*. We mentioned that in *ssPLSA-mem hard* we perform hard clustering by fixing the values of $P(y|\alpha)$. The idea here is to perform soft clustering for the unlabeled data. In other words, the repartition for the unlabeled data is not fixed. We denote by $\tilde{P}(y|\alpha)$ the values for the unlabeled data, which are obtained during the training of the model. For the labeled examples, we initialize, as before, by forcing to zero the $P(y|\alpha)$ for the latent topic variables α which do not belong to the particular class y.

We decided to perform hard clustering for the labeled examples, because, as discussed in (Gaussier and Goutte, 2005), the soft clustering potentially faces the problem of *cluster impurity*: all components contain examples from several classes instead of "specialising" to one or few classes. As a consequence, even if we use the unlabelled data to better model these components, this will not help to discriminate the different classes. On the other hand, by allowing soft clustering only on the unlabeled examples, we do not face this problem and in addition we give the possibility to the unlabeled examples to be distributed over all components.

Hence, in this case the joint probability between an unlabeled example $q \in \mathcal{Q}_u$ and its imperfect class label estimated by the classifier can be expressed as

$$P(w, x, \tilde{y}) = P(x) \sum_{\alpha \in A} P(w|\alpha) P(\alpha|x) \sum_{y \in \mathcal{C}} \beta_{\tilde{y}|y} \tilde{P}(y|\alpha)$$

The model parameters

$$\Psi = \{P(\alpha \mid x), P(w \mid \alpha), \beta_{\tilde{y}|y}, \tilde{P}(y|\alpha) : x \in \mathcal{X}, w \in \mathcal{W}, \alpha \in A, y \in \mathcal{C}, \tilde{y} \in \mathcal{C}\}$$

are estimated by maximizing the complete data log-likelihood:

$$\begin{aligned}\mathcal{L}_3 = &\sum_{x\in X_l}\sum_{w} n(w,x)\log\sum_{\alpha} P(x)P(w|\alpha)P(\alpha|x)P(y|\alpha) \\ &+ \sum_{x\in X_u}\sum_{w} n(w,x)\log\sum_{\alpha} p(x)p(w|\alpha)p(\alpha|x)\sum_{y}\beta_{\tilde{y}|y}\tilde{P}(y|\alpha)\end{aligned} \tag{4 .12}$$

As we can notice, the difference with the ssPLSA-hard strategy, relies on the introduction of $\tilde{P}(y|\alpha)$ for unlabeled data, which is not fixed, but is estimated during the `EM` algorithm. In order to initialize this parameter, we do not force to zero any of its values, but nevertheless, we favorize the components for which the $P(y|\alpha)$ of the labeled examples is not zero, by giving them bigger values. In other words, we initialize this parameter in such a way so that it is not very far from the $P(y|\alpha)$ (but on the other hand we do not forced to zero any value, as we want to perform soft clustering). We decided to initialize that way in order to avoid identifiability problems which can occur. For the training of the current model, we use again the equations (4 .7), (4 .10), (4 .11) and (4 .8), but this time the latent topic posterior $\tilde{\pi}_\alpha(w,x,\tilde{y})$ for the unlabeled data is defined as follows:

$$\tilde{\pi}_\alpha(w,x,\tilde{y}) = \frac{P(\alpha|x)P(w|\alpha)\sum_y \tilde{P}(y|\alpha)\beta_{\tilde{y}|y}}{\sum_\alpha P(\alpha|x)P(w|\alpha)\sum_y \tilde{P}(y|\alpha)\beta_{\tilde{y}|y}} \tag{4 .13}$$

In addition, the $\tilde{P}(y|\alpha)$ is estimated over the unlabeled training set:

$$\tilde{P}^{(j+1)}(y|\alpha) = \tilde{P}^{(j)}(y|\alpha)\sum_w\sum_{x\in X_u} n(w,x)\frac{P^{(j)}(\alpha|x)P^{(j)}(w|\alpha)\beta^{(j)}_{\tilde{y}(x)|y}}{\sum_\alpha P^{(j)}(\alpha|x)P^{(j)}(w|\alpha)\sum_y \tilde{P}^{(j)}(y|\alpha)\beta^{(j)}_{\tilde{y}|y}} \tag{4 .14}$$

The procedure for estimating model parameters Ψ is described in algorithm 18.

The complexity of this algorithm is $O(|A|\times M\times \mathcal{C})$, which is comparable with the one of the *semi-supervised Naive Bayes* (as presented in chapter 2) and *ssPLSA-fake* algorithms.

Algorithm 18: ssPLSA-mem soft

Input :

- A set of partially labeled data $\mathcal{X} = X_l \cup X_u$,
- Random initial model parameters $\Psi^{(0)}$.
- $j \leftarrow 0$
- Run a simple PLSA algorithm for the estimation of the initial $\tilde{y}$

repeat

`E-step`: Estimate the latent class posteriors

$$\pi_\alpha(w,x,y) = \frac{P(\alpha|x)P(w|\alpha)P(y|\alpha)}{\sum_\alpha P(\alpha|x)P(w|\alpha)P(y|\alpha)}, \text{ if } x \in \mathcal{X}_l$$

$$\tilde{\pi}_\alpha(w,x,\tilde{y}) = \frac{P(\alpha|x)P(w|\alpha)\sum_y \tilde{P}(y|\alpha)\beta_{\tilde{y}|y}}{\sum_\alpha P(\alpha|x)P(w|\alpha)\sum_y \tilde{P}(y|\alpha)\beta_{\tilde{y}|y}}, \text{ if } x \in \mathcal{X}_u$$

`M-step`: Estimate the new model parameters $\Psi^{(j+1)}$
by maximizing the complete-data log-likelihood

$$P^{(j+1)}(w|\alpha) \propto \sum_{x\in X_l} n(w,x)\pi_\alpha^{(j)}(w,x,y(x)) + \sum_{x\in X_u} n(w,x)\tilde{\pi}_\alpha^{(j)}(w,x,\tilde{y}(x))$$

$$P^{(j+1)}(\alpha|x) \propto \sum_w n(w,x) \times \begin{cases} \pi_\alpha^{(j)}(w,x,y(x)), \text{ for } x \in X_l \\ \tilde{\pi}_\alpha^{(j)}(w,x,\tilde{y}(x)), \text{ for } x \in X_u \end{cases}$$

$$\beta_{\tilde{y}|y}^{(j+1)} \propto \sum_w \sum_{x\in X_u} n(w,x) \sum_{\alpha|\alpha\in y} \tilde{\pi}_\alpha^{(j)}(w,x,\tilde{y})$$

$$\tilde{P}^{(j+1)}(y|\alpha) = \tilde{P}^{(j)}(y|\alpha)\sum_w \sum_{x\in X_u} n(w,x)\frac{P^{(j)}(\alpha|x)P^{(j)}(w|\alpha)\beta_{\tilde{y}(x)|y}^{(j)}}{\sum_\alpha P^{(j)}(\alpha|x)P^{(j)}(w|\alpha)\sum_y \tilde{P}^{(j)}(y|\alpha)\beta_{\tilde{y}|y}^{(j)}}$$

$j \leftarrow j+1$

until *convergence of the complete-data log-likelihood* ;

Output : A generative classifier with parameters $\Psi^{(j)}$

The experiments in chapter 6 will prove that the soft clustering in the un-

labeled training data is really beneficial, especially when the ratio of labeled-unlabeled data is very low.

Remarks The following matrix sums up the models presented in this chapter, comparing the parameters they control and their complexity. As we can see the complexity of all four is comparable. In chapter 6, we test these methods and discuss how the different parameters can affect their performances.

All the above models can be performed directly in multiclass classification tasks, without any modification. And this can be proved a great advantage with respect to binary classification semi-supervised models, as in may real world classification problems are multiclass, and many of the existing methods cannot handle multiclass problem easily.

	Parameters				**Complexity**
Models	$P(w\|\alpha)$	$P(\alpha\|x)$	$P(\tilde{y}\|y)$	$P(y\|\alpha)$	$O()$
PLSA	$\surd$	$\surd$			$O(\|A\| \times M)$
ssPLSA-fake	$\surd$	$\surd$		$\surd$	$O(\|A\| \times M)$
ssPLSA-mem hard	$\surd$	$\surd$	$\surd$		$O(\mathcal{C} \times \|A\| \times M)$
ssPLSA-mem soft	$\surd$	$\surd$	$\surd$	$\surd$	$O(\mathcal{C} \times \|A\| \times M)$

Table 4 .1: Comparison of the different variants of the semi-supervised PLSA model. For the complexities $M = \#\{(w,x)|n(w,x) > 0\}$

4 .8 Conclusion

In this chapter we presented three semi-supervised variants of the Probabilistic Latent Semantic Analysis. These aspect models use both label and unlabeled data and at the same time, they model the possible mislabeling errors. First a variant (*ssPLSA-fake*) which uses fake labels is presented and then two slightly different models were proposed (*ssPLSA-mem hard* and *ssPLSA-mem soft*). In the next chapter, we extend these models by combining them with two different active learning techniques.

5

Active Semi-supervised Aspect Models

Contents

5 .1 Introduction 83
5 .2 Margin-Based Method 85
5 .3 Entropy-Based Method 86
5 .4 Conclusion 88

5 .1 Introduction

In this chapter, we extend the presented semi-supervised models, by combining them with two active learning methods. The motivation is to try to take advantage of the characteristics of both frameworks.

As discussed in chapter 3, the combination of both semi-supervised and active learning appears to be particularly beneficial in reducing the annotation burden for the following reasons:

1. It constitutes an efficient way of solving the exploitation/exploration problem: semi-supervised learning is more focused on exploitation, while

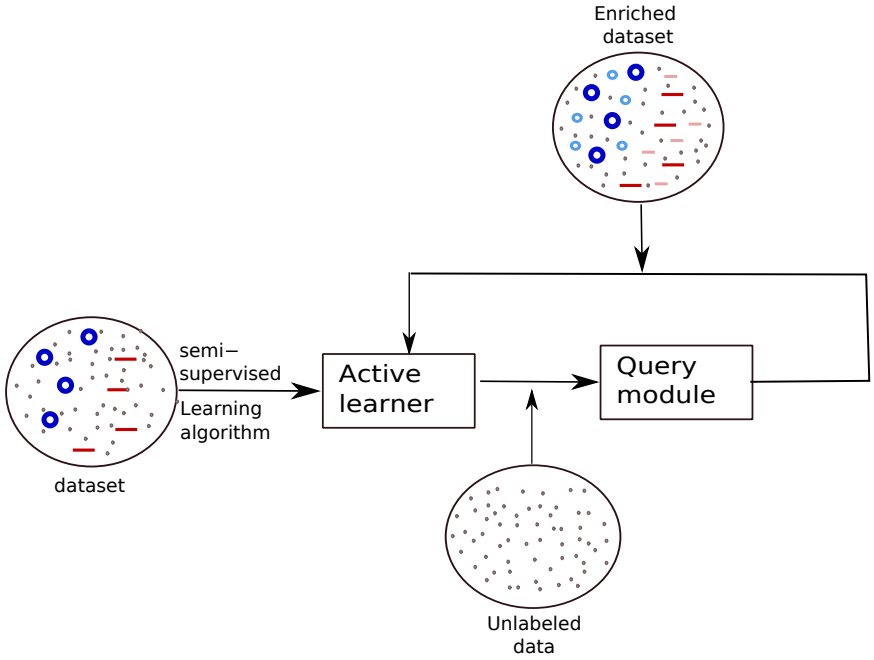

Figure 5 .1: Combining semi-supervised and Active learning

active learning is more dedicated to exploration. Semi-supervised learning alone may lead to poor performance in the case of very scarce initial annotation. It then strongly suffers from poorly represented classes, while being very sensitive to noise and potentially instability. On the other hand, active learning alone may spend too much time querying useless examples, as it can not exploit the information given by the unlabeled data.

2. In the same vein, it may alleviate the data imbalance problem due to each method separately. Semi-supervised learning tends to over-weight easy-to-classify examples that will dominate the process, while active learning has the opposite strategy, resulting in exploring more deeply the hard-to-classify examples (Tür et al., 2005).

3. Semi-supervised learning is able to provide a more motivated estimation of the confidence score associated to the class prediction for each

example, taking into account the whole data set, including the unlabelled data. As a consequence, active learning based on these better confidence scores can be expected to be more efficient.

In the next two sections, we present two different active learning methods which can be performed on the top of the semi-supervised models presented in the previous chapter. These model can be also used with any other semi-supervised probabilistic model. In both methods, we choose to annotate the less confident example. Their difference lies on the measure of confidence they use.

5 .2 Margin-Based Method

The first active learning method (the so-called margin method) chooses to annotate the example which is closer to the classes' boundaries (Krithara et al., 2006). The latter gives us a notion of confidence the classifier has on the classification of these examples. In order to measure this confidence we use the following class-entropy measure for each unlabeled example:

$$B(x) = -\sum_{y} P(y|x) \log P(y|x), \text{ where } x \in \mathcal{X}_u \qquad (5\ .1)$$

The bigger the B is, the less confident the classifier is about the labeling of the example. After having selected an example, we annotate it and we add it to the initial labeled set $\mathcal{X}_l$. More than one examples can be selected at each iteration. The reason is that, especially for classification problems with a big amount of examples and many classes, the annotation of only one example at a time, can be proved time-consuming, as a respectful amount of labeled examples will be needed in order to achieve good performances. If we choose to do the latter, it is not wise to choose examples that are next to each other, as they cannot give us more information than each of them does. As a result, it is better to choose, for instance, examples with big class-entropy which have been given different labels. That way the classifier can get information about different classes and not only one.

Algorithm 19 gives us the general framework under which the above active learning method can be combined with any semi-supervised variant of the PLSA model.

Algorithm 19: Combining ssPLSA and Active Learning

Input : A set of partially labeled documents $\mathcal{X} = \mathcal{X}_l \cup \mathcal{X}_u$

repeat

- Run the ssPLSA algorithm (and calculate the $P(y|x)$)
- Estimate the confidence of the classifier on the unlabeled examples: $\forall x \in \mathcal{X}_u$, $B(x) = -\sum_y P(y|x) \log P(y|x)$
- Choose the example(s) with low confidence, i.e. higher value of B (if we choose more than one example to label, we choose examples with have been classified into different classes) and add them in the labeled dataset $\mathcal{X}_l$

until *a given number of queries or a certain performance* ;

Output : A generative classifier

5 .3 Entropy-Based Method

In this section, we present another active learning method, which can be combined with the semi-supervised framework. Based on the method presented by (Dagan and Engelson, 1995), we calculate the entropy of the annotation of the unlabeled data, during the iterations of the algorithm. This method can be seen as a query by committee approach, where, in contrast to the method of (Dagan and Engelson, 1995), the committees here are the different iterations of the same model.

In contrast to the margin based method presented previously, the current one does not use the probabilities $P(y|x)$ of an example x to be assigned the label y but, instead, is uses the deterministic votes of the classifier during the different iterations. We denote by $V(y, x)$ the number of times that the label

y was assigned in the example x during the previous iterations.

Then, we denote as *Vote Entropy* of an example x as:

$$VE(x) = -\sum_{y} \frac{V(y,x)}{iters} log \frac{V(y,x)}{iters} \quad (5\ .2)$$

where *iters* refers to the number of iterations.

The examples to be labeled are chosen using equation (5 .2), that is, examples with higher entropies are selected. As long as we add new examples during the iterations, the labeling of some examples will change as, new information will be given to the classifier. Thus, the strategy chooses the examples for which the classifier changes its decision more often during the iterations. We have to note, that during the first 2-3 iterations, we do not have enough information in order to choose the best examples to label, but very quickly the active learner manage to identify these examples. The intuition behind this model is that examples which tend to change labels are those for which the classifier seems more undecided. In algorithm (20) the combination of this method with the semi-supervised PLSA is described.

Algorithm 20: Combining ssPLSA and Active Learning

Input : A set of partially labeled documents $\mathcal{X} = \mathcal{X}_l \cup \mathcal{X}_u$

repeat

- Run the ssPLSA algorithm
- Update the VE for each of the examples, according to the decision of the classifier in the current iteration
- Choose the example(s) with the highest entropy and add them in the labeled dataset $\mathcal{X}_l$

until *a certain number of queries or a certain performance* ;

Output : A generative classifier

5 .4 Conclusion

In this chapter, we propose the combination of the semi-supervised PLSA models presented in the previous chapter with two active learning methods. Both try to measure the confidence of the classifier by using two different strategies. The less confident examples are chosen for annotation and the classifier is retrained with the updated training set. In the next chapter we discuss the performance of these models, and the benefits they can offer to the semi-supervised learning.

6

Evaluation

Contents

6 .1 Introduction 89
6 .2 Document Categorization 90
6 .2.1 Data Representation 91
6 .3 Datasets 93
6 .4 Evaluation Measures 96
6 .5 Experiments 96
6 .5.1 ssPLSA Results 101
6 .5.2 Active ssPLSA Results 106
6 .6 Conclusion 111

6 .1 Introduction

In the two previous chapters, different methods for semi-supervised and active learning have been presented. In the current chapter, we try to evaluate all these models and compare their performance with some well known state-of-the-art techniques. We are focusing on document classification, that we start to describe. Then, we present the datasets we used for the compqrisons. Then, an evaluation with a real-world dataset, provided by a Xerox Business Group, is performed.

6 .2 Document Categorization

Document Categorization (or *classification*) refers to the task of assigning categories, to a given set of documents. The automated categorization framework dates from the early 60's when *knowledge engineering techniques* were used in order to built manually classifiers, by defining a set of rules encoding expert knowledge (for example (Hayes et al., 1990)). But it was only in the beginning of the 90's that document categorization has started to draw the attention of the Machine Learning community. The motivation was both the applicative interests, and the increasing amounts of available data (Sebastiani, 2002).

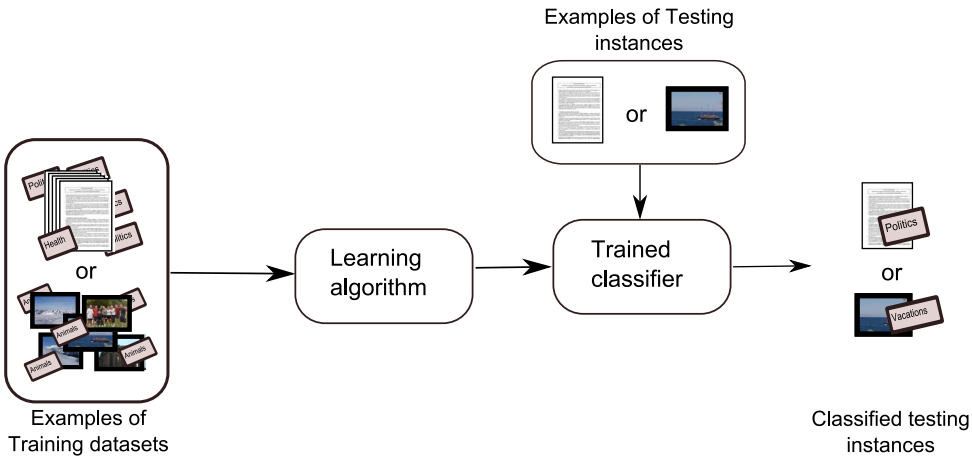

Figure 6 .1: The procedure of categorization: given some training examples already labeled (for example a set of documents or a set of images) and the specified categories, a classifier is trained. Then, the latter is able to classify new unlabeled instances to the respective categories.

Categorizing data into thematic categories usually follows the supervised learning paradigm: we train a learner using some already classified instances and then this trained learner is used to classify new unclassified instances (figure 6 .1).

6 .2.1 Data Representation

In order for a learning algorithm to interpret a dataset, the latter has to be processed to a form which the algorithm can process. One of the most widely used representations is a multidimensional feature vector. The intuition behind this representation is simple: examples which contain the same features probably belong to the same topics.

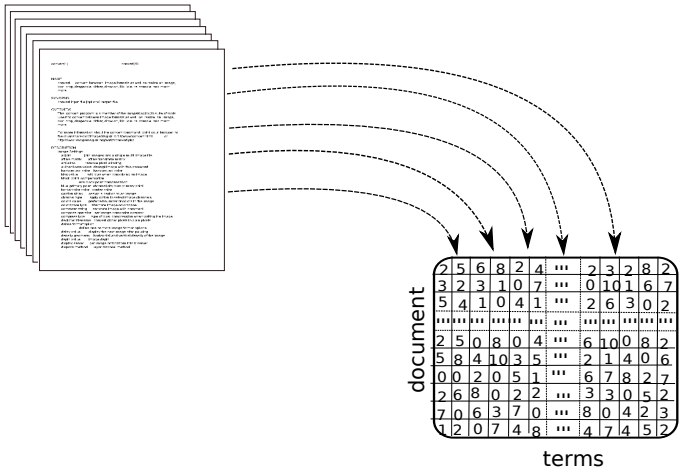

Figure 6 .2: The documents are represented as a term-document matrix, where the frequency of a term within a document is given.

In the case of document classification, this representation corresponds to the so-called *bag-of-words*. In the latter, features are words occuring in the documents. It is common to do some additional pre-processing before creating the multidimensional feature vector. That includes stemming, removal of very common words and collapsing of multiple occurrences of words into one. That way we are able to decrease the dimensionality of the data and, as a result, we can gain in terms of time and performance.

After determining the features which are going to be used (i.e. the words remaining after the pre-processing), we calculate the matrix of the feature frequencies, that is, the number of times a given word appears in each docu-

ment in the collection (figure 6 .2). In other terms, we denote each document $x \in \mathcal{X}$ as a vector $x =< n(w,x) >_{w \in \mathcal{W}}$, where w indicates the features from the set of features $\mathcal{W} = \{w_1, \ldots, w_{N_w}\}$, and $n(w_i, x)$ the number of times the word w_i occurs in the document x. This count is sometimes normalized to prevent a bias towards longer documents and results in values in the range between 0 and 1.

As a particular case, some authors (Lewis and Ringuette, 1994; Koller and Sahami, 1997; Schapire and Singer, 2000) have used a binary matrix to represent their datasets, where 1 denotes the presence of a word in a document and 0 its absence. In general, such a representation is used when the applied algorithm can only handle symbolic or non-numeric values.

Another very common technique is the weighting of the features, by using the $tf \cdot idf$ score (for example (Salton and Buckley, 1988)). The latter refers to the *"term frequency-inverse document frequency"* weighting function. In other words, instead of just counting the words in the documents, we weight the features' frequencies by using the following transformation:

$$tfidf(w_j, x_i) = n(w_j, x_i) \log \frac{|\mathcal{X}|}{\#\mathcal{X}(w_j)} \qquad (6\ .1)$$

where $|\mathcal{X}|$ denotes the total number of documents in the collection and $\#\mathcal{X}(w_j)$ the number of documents in which w_j occurs at least once (also known as *document frequency* of term w_j).

Different variants of the *bag-of-words* representation can be used, according to the needs of the learning algorithm used. For example, in graphical methods, an adjacency (weight) matrix is usually used. The weights represent the similarity between the examples and can be calculated using different functions (e.g. the Gaussian kernel).

Nevertheless, the use of the *bag-of-words* representation has some drawbacks. In the case of document classification for example, problems can be

caused by the properties of the human languages: *polysemy* and *synonymy*. According to the former, there exist words which have more than one meaning. As a result, if two documents contain such a word, they will be categorized in a similar way, even if they talk about completely different topics. For example, let us consider two documents which contain the word "apple": using the *bag-of-words* representation they will be classified in the same way, even if the one talks about fruits and the other about mac computers. The second property, *synonymy*, can lead to the opposite situation: if two documents contain two different words with the same meaning (for example, "car" and "automobile") they will not be classified in the same category even if they talk about the same topic. As we described in chapter 4, we have chosen to use *Aspect models*, in order to overcome these problems.

6 .3 Datasets

In our experiments we used four different datasets: two collections from the CMU World Wide Knowledge Base project - `WebKB` and `20Newsgroups`, the widely used text collection of `Reuters` (`Reuters` − 21578) and a real-world dataset from Xerox. As mentioned before, we are concentrated in document classification; nevertheless, the algorithms described in the previous chapters can be also used for different applications in which there is a relation of co-occrence between objects and variables such as image classification.

The `20Newsgroups` dataset[4] is a state-of-the-art document collection for text classification. The data set is a collection of approximately 20000 newsgroup documents, organized into 20 different newsgroups, each corresponding to a different topic. Some of the newsgroups are very closely related to each other (e.g. comp.sys.ibm.pc.hardware / comp.sys.mac.hardware), while others are highly unrelated (e.g misc.forsale / soc.religion.christian). In figure 6 .3 all the different 20 newsgroups are presented, partitioned (more or less) according to subject matter.

[4] http://people.csail.mit.edu/jrennie/20Newsgroups/

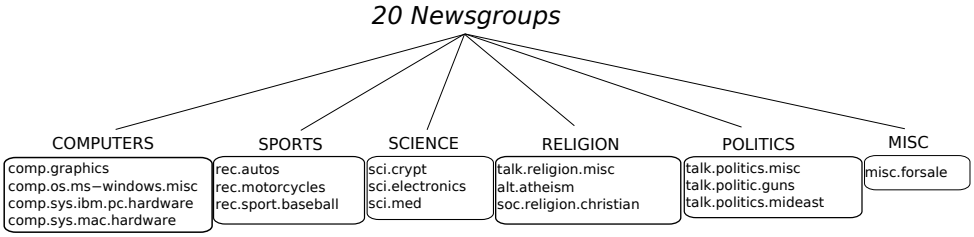

Figure 6 .3: The structure of the 20 Newsgroups dataset.

The `WebKB` dataset[5] (Craven et al., 1998) contains web pages gathered from 4 different university computer science departments (Cornell, Texas, Washington and Wisconsin). The pages are divided into seven categories. In our evaluation, we use the four most used entity-representing categories in the literature (Nigam et al., 2000): *student*, *faculty*, *course* and *project*, all together containing 4196 pages.

The `Reuters` dataset[6] consists of 21578 articles and 90 topic categories from the Reuters newswire. We selected the documents which belong only to one class, and in addition we only kept the classes which contain at least 100 documents. This gave as a base of 4381 documents classified in 7 different classes: ACQ (1084), EARN (2052), CRUDE (296), GRAIN (286), INTEREST (106), MONEY (377) and TRADE (207).

These three datasets were pre-processed as follows:

- Email tags as well as other non-alpha numeric terms were removed
- All documents were tokenized on white space and punctuation
- tokens in less than 5 documents in each test collection were discarded
- A total of 608 stopwords from the `CACM` stoplist[7] were removed from each document.

[5]http://www.cs.cmu.edu/afs/cs.cmu.edu/project/theo-20/www/data/
[6]http://www.daviddlewis.com/resources/testcollections/reuters21578/
[7]http://ir.dcs.gla.ac.uk/resources/test_collections/cacm/

No other form of pre-processing (stemming, multi-word recognition etc.) was used on the documents. Table 6 .1 summarizes the characteristics of these datasets.

Dataset	20Newsgroups	WebKB	Reuters
Collection size	20000	4196	4381
# of classes, K	20	4	7
Vocabulary size, $\|\mathcal{W}\|$	38300	9400	4749
Training set size, $\|X_l \cup X_u\|$	16000	3257	3504
Test set size	4000	839	876

Table 6 .1: Characteristics of the datasets

XLS dataset

Apart from the datasets above which, as we mentioned, are widely used for evaluation of different classification algorithms in the Machine Learning community, we used a real world dataset (called XLS) which was provided by a Xerox Business Group (XLS). As we mentioned in the introduction of this thesis, the developpement of our models was done in the framework of research project conducted in Xerox Research Centre Europe. The motivation was to extend a previously developed classification system by adding the aspects of semi-supervised and active learning that we developped here. As a result, we compared our models, with the previous classification system (CategoriX), in order to evaluate the amelioration they could offer.

The Xerox Litigation Services (XLS) business group is looking to improve their operations by integrating the Xerox CategoriX Technology into their online portal. XLS technology is intended to provide customers an efficient and secure way to collaborate with law firms, partners and government agencies on litigation and regulatory compliance matters. It provides the identification, filtering, production and storage of relevant data in the form of paper and electronic documents. Examples of electronic documents include: email, text files, memos, databases, presentations and spreadsheets. XLS have provided a sample of documents as well as a list of categories on which we performed

our experiments.

This dataset contains 20000 documents in the training set and 34770 in the test set. The documents consist of approximately 40% emails, 20% Microsoft Word documents; 20% Microsoft Excel documents, 10% Microsoft Power point documents and 10% PDF and other documents. We want to classify the documents as *Responsive* and *Non-Responsive* to a particular given case. The two categories are balanced (50%/50%). We compared our results, with the ones of the current version of CategoriX.

6 .4 Evaluation Measures

In order to evaluate the performance of the models, we used the microaverage F-score measure for all experiments.

For each classifier, $\mathcal{G}_f$, we first compute its microaverage precision P and recall R by summing over all the individual decisions it made on the test set:

$$R(\mathcal{G}_f) = \frac{\sum_{k=1}^{K} \theta(k, \mathcal{G}_f)}{\sum_{k=1}^{K} (\theta(k, \mathcal{G}_f) + \psi(k, \mathcal{G}_f))}$$

$$P(\mathcal{G}_f) = \frac{\sum_{k=1}^{K} \theta(k, \mathcal{G}_f)}{\sum_{k=1}^{K} (\theta(k, \mathcal{G}_f) + \phi(k, \mathcal{G}_f))}$$

Where, $\theta(k, \mathcal{G}_f)$, $\phi(k, \mathcal{G}_f)$ and $\psi(k, \mathcal{G}_f)$ respectively denote the true positive, false positive and false negative documents in class k found by $\mathcal{G}_f$. The F-score measure is then defined as (Lewis and Ringuette, 1994):

$$F(\mathcal{G}_f) = \frac{2P(\mathcal{G}_f)R(\mathcal{G}_f)}{P(\mathcal{G}_f) + R(\mathcal{G}_f)}$$

6 .5 Experiments

We compared the performance of the models on the 4 datasets by varying the percentage of labeled examples in the training set and using 10-fold cross

validation. We performed 10 runs for each of the folds and we calculated the average F-score (as we initialize some of the variables by random, we wanted to ensure that the obtained results do not depend on this random initialization). In order to evaluate the significance of the obtained results, we perfomed a t-test at the 5% significance level.

For the three well-known datasets (`WebKB`, `Reuters` and `20Newsgroups`), we compared our models with two state-of-the-art methods in text classification: the semi-supervised Naive Bayes classifier (Nigam et al., 2000) and the transductive SVM classifier (SVM-light package (Joachims, 1999)). For the latter, we used a linear kernel and we optimized the cost parameter, using a nested cross-validation. We performed the one vs. all strategy (we fusion the results by choosing, for each example, the class with the maximum score). We could pssibly obtain better results from TSVM if we had used a non-linear kernel, but the latter was computationaly intractable, comparing with the computational time of the other models.

For the `XLS` dataset, we compared our models with the supervised PLSA model, as used for the CategoriX system.

For all four datasets, we fixed the value of $\lambda = 0.01$ of the ssPLA-fake algorithm (we have tried different values of *lambda*, but 0.01 gave us the best results).

In order to have an upper bound on the performance of the compared classifiers,we first compare the systems in a fully supervised way, that is when 100% of the documents in the training set have their true labels and are used for training the classifiers. Table (6 .2) sums up these results. As we can notice, all PLSA models behave identically, which is expected, if we consider that there are no unlabeled training documents, and as a result, no fakes or mislabeling errors to characterize.

In order to evaluate empirically the effect of unlabeled documents for

	20Newsgroups	WebKB	Reuters
System	F-score (%)	F-score (%)	F-score (%)
Naive Bayes	88.23	84.32	93.89
PLSA	$\|A\| = 40$	$\|A\| = 16$	$\|A\| = 14$
	89.72	**85.54**	**94.29**
SVM	88.98	85.15	89.50

Table 6 .2: Comparison of the F-score measures between the Naive Bayes and PLSA generative models as well as the SVM classifier on 20Newsgroups, WebKB and Reuters test sets, where $|A|$ is the number of components. All classifiers are trained in a fully supervised way.

training the models we have also trained the PLSA model in a purely supervised way (with the corresponding percentage of randomly selected labeled documents). We used the supervised PLSA model described in chapter 4.

Number of Components. As we mentioned in the description of the models, the number of latent variables $|A|$ (i.e. the number of components) must be defined by the user, during the initialization. The latter depends mostly on the dataset and its distribution. As a result, the issue of how to choose this number occurs, as this parameter is quite important for the performance of the models (especially in the semi-supervised framework). We tried different techniques, in order to find which is the most convenient one.

We firstly tried to find the best number of components empirically, i.e. by doing cross-validation using different number of components and comparing the results. In this framework, we perfomed some experiments under two different scenarios: In the first, we considered that all classes have an equal number of components (Method 1). In the second, we consider that we have at least one component per class, and in addition, we have a number of components that we do not know in which class they belong, i.e. we let the algorithm to assign them in the different classes during the training process (Method 2). Table (6 .3) presents some representative results obtained for the 20Newsgroups dataset for different numbers of components and for 10-

fold cross-validation.

20Newsgroups			
Method 1		**Method** 2	
$\|A\|$	F-score (%)	$\|A\|$	F-score (%)
20 (1 per class)	88.93 ± 0.51	20 (0 additional)	88.93 ± 0.51
40 (2 per class)	$\mathbf{89.72 \pm 0.46}$	25 (5 additional)	88.52 ± 0.63
60 (3 per class)	89.21 ± 0.30	30 (10 additional)	89.13 ± 0.18
80 (4 per class)	88.77 ± 0.21	40 (20 additional)	$\mathbf{89.32 \pm 0.36}$
100 (5 per class)	87.34 ± 0.49	50 (30 additional)	88.83 ± 0.43
120 (6 per class)	87.73 ± 0.65	60 (40 additional)	88.49 ± 0.21

Table 6 .3: Comparison of the F-score measures on 20Newsgroups for the supervised PLSA. The first variant we have $|A|$ components, equally splited in the classes and the second one supposes we have one component per class plus a number of additional components

We also tried to define the number of components using a more sophisticated way: the idea was to start with one component per class, and then iteratively find and split the most *heterogeneous* components. In order to calculate the heterogeneity of a component we used the following equation:

$$volume(\alpha) = \frac{\sum_x P(\alpha|x) KL(P(w|x), P(w|\alpha))}{\sum_x P(\alpha|x)}$$

where $KL()$ refers to the *Kullback-Leibler divergence* (Kullback and Leibler, 1951). The above equation is actually measuring the average distance of the examples and the component they belong to: the closer the examples are in the profile of the component, the more homogeneous the component is. The results of this method are presented in table (6 .4).

As we can see from both tables (6 .3,6 .4), this method do not seem to obtain better results than the previous one, and in addition is more complex in terms of computation. The results we obtained for the other datasets indicate the same conclusion. By simply fixing the number of components per class we obtain good performance for PLSA. We have performed the same

20Newsgroups			
Heterogeneity			
$\|A\|$	F-score (%)	$\|A\|$	F-score (%)
20 (0 splits)	88.93 ± 0.51	50 (30 splits)	89.11 ± 0.35
25 (5 splits)	88.72 ± 0.29	60 (40 splits)	88.21 ± 0.72
30 (10 splits)	89.03 ± 0.11	100 (60 splits)	88.27 ± 0.51
40 (20 splits)	$\mathbf{89.73 \pm 0.67}$	120 (100 splits)	87.86 ± 0.55

Table 6 .4: F-score on 20Newsgroups for the supervised PLSA. We start with one component per class. We calculate the heterogeneity of the components and we split them appropriatily. The $|A|$ indicates the final number of components after different splits. In the second one we calculate the AIC and we choose the number for which the latter is bigger.

kind of experiments for the semi-supervised case (especially when the number of labeled examples is really small). The results we obtained by performing the above methods indicated also that the cross-validation seems to be most efficient for the choice of $|A|$. The infuence of $|A|$ in the semi-supervised case is discussed in more details in the next section, as the ssPLSA is more sensitive to the initialization.

For our datasets we obtained the best results by using $|A| = 40$ for the 20Newsgroups, $|A| = 16$ for the WebKB, $|A| = 14$ for the Reuters, and $|A| = 4$ for the XLS.

All these experiments, were performed in a non nested cross validation. That is, we performed the methods above in a supervised way, in order to find the best number of components for each dataset, and then we performed the experiments with our methods. We have chosen this procedure, for one main reason: in the semi-supervised setting, and especially when the ratio of labeled and unlabeled examples is very low, we do not have enough data to determine the best number of componets.

To sum up, we can conclude that by simply supposing that the classes have an equal number of components and calculating it using cross-validation,

we can obtain tquite good performaces, at least for the datasets we used in our evaluation. As a result, we decided to use this method instead of a more sophisticated one.

6 .5.1 ssPLSA Results

In this section, we present the results obtained for the semi-supervised models alone. Figures (6 .4) and (6 .5) show the F-score measure over the test sets on the three data collections (`20Newsgroups`, `WebKB` and `Reuters`) for semi-supervised learning for different ratio of labeled-unlabeled documents in the training set. 5% in the x-axis means that 5% of the labeled documents ($|X_l|$) in the training sets were used for training, the 95% remaining being used as unlabeled training documents ($|X_u|$). We compared the three semi-supervised variants of PLSA, as presented in chapter 4, with the TSVM and semi-supervised Naive Bayes. The *ssPLSA-mem soft* uniformly outperforms the other models on these datasets. This is particularly clear for `20Newsgroups` which is a more complex classification problem. With only 5% of labeled documents in the training set, the F-score of the ssPLSA with mislabeling algorithm is about 8% over the ssPLSA with fake labels on `20Newsgroups`. Using only 10% allows to reach 80% of the maximum F-score on `20Newsgroups` while the 90% remaining labeled documents allows to reach the maximum performance level. The semi-supervised Naive Bayes model outperforms on the other hand the *ssPLSA-fake* labels on all datasets. This might be due to the fact that the *ssPLSA-fake* algorithm tries to measure the confidence of the results, rather than directly ameliorate the performance.

As we can notice, the results of the TSVM are bad in these experiments. This can be explained by the fact that the model was initially designed for 2-class classification problems and the one vs. all strategy does not give adequate recognition of classes.

In order to evaluate empirically the effect of unlabeled documents for training the models we have also trained the PLSA model in a supervised

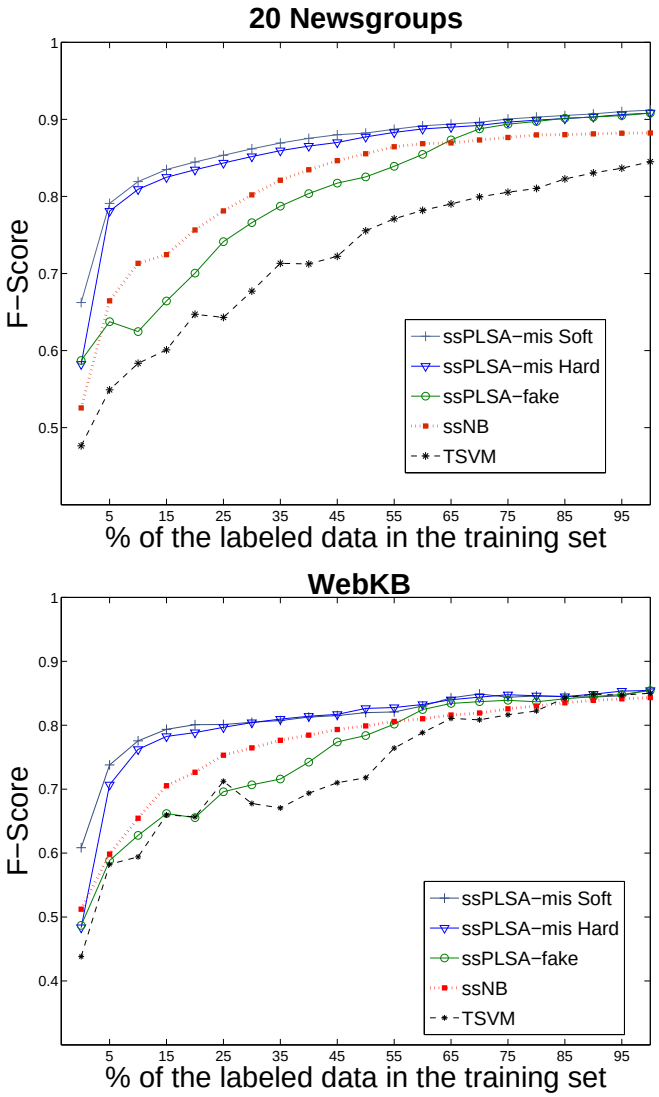

Figure 6 .4: F-Score (y-axis) versus, the percentage of labeled examples in the training set $|X_l|$, (x-axis) graphs for the various algorithms on `20Newsgroups` (left) and `WebKB` (right).

manner using only the percentage of labeled documents in the training set. We can see that semi-supervised algorithms are able to take advantage from unlabeled data. For example, for the `20Newsgroups` dataset (figure 6 .6), with

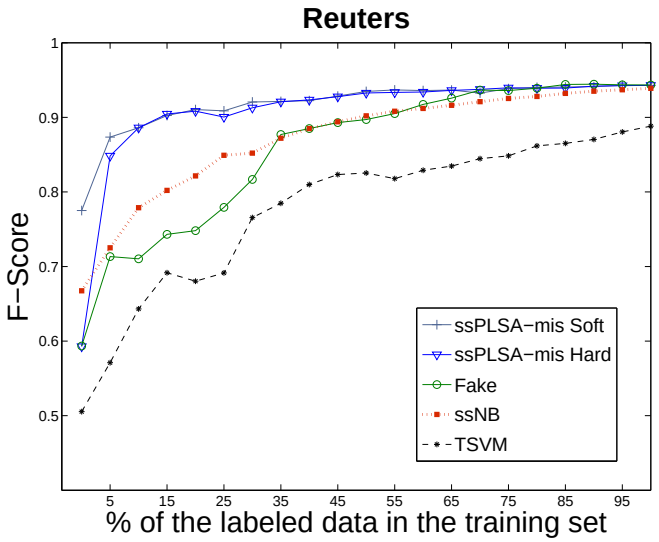

Figure 6 .5: F-Score (y-axis) versus, the percentage of labeled examples in the training set $|X_l|$, (x-axis) graphs for the various algorithms on `Reuters` dataset.

5% labeled data, the fully supervised PLSA reaches 52.5% F-score accuracy while *ssPLSA-fake* achieve 63% and *ssPLSA-mem Hard ssPLSA-mem Soft* achieve 79%. As we can notice on the figure 6 .6, the gain with the use of the unlabeled data is similar for the other two datasets (`WebKB` and `Reuters`).

One interesting aspect of our experimental results is that the behavior of the three *ssPLSA* variants is very different when the number of latent variables per class increases. As we mentioned in the previous section, the latter is defined during the initialization. In the semi-supervised framework, the initialization is more sensitive comparing with the supervised case, and as a result the number of components has an important influence in the performance. As we can see in the table 6 .5, for the *ssPLSA-fake* approach the variability of the results increases when more components are added to the model. Overall, this approach yields consistently lower performance than the *ssPLSA-mem* approaches, which in addition seem less sensitive to varying numbers of components. Notice how, when the number of components per

class is increased from 1 to 2 corresponding respectively to $|A| = 20$ and $|A| = 40$ for `20Newsgroups` dataset or 4, $|A| = 16$ for the `WebKB` dataset, the performance of the *ssPLSA-mem* approach increases slightly, but consistently. In addition, the variability of the results is mostly well contained and generally smaller than for the *ssPLSA-fake* approach. The results for the other two datasets indicate the same trend of the algorithms. This result, gives an advantage in the two ssPLSA-mem methods: the choice of the number of components is not of that crucial importance as it is for the ssPLSA-fake model.

`20Newsgroups`					
		1%	5%	20%	40%
$\|A\| = 20$	ssPLSA-mem Soft	65.96 ± 0.89	79.13 ± 0.11	83.59 ± 0.66	85.63 ± 0.42
	ssPLSA-mem Hard	57.52 ± 0.59	76.42 ± 0.16	83.24 ± 0.57	85.54 ± 0.3
	ssPLSA-fake	57.04 ± 0.68	63.75 ± 0.78	70.05 ± 0.68	79.59 ± 0.28
$\|A\| = 40$	ssPLSA-mem Soft	$\mathbf{66.23 \pm 0.52}$	$\mathbf{80.01 \pm 0.23}$	$\mathbf{84.42 \pm 0.73}$	$\mathbf{85.9 \pm 0.85}$
	ssPLSA-mem Hard	58.24 ± 0.46	77.18 ± 0.2	83.47 ± 0.35	$\mathbf{85.76 \pm 0.69}$
	ssPLSA-fake	57.87 ± 0.41	59.75 ± 1.09	65.75 ± 0.98	78.86 ± 0.39

`WebKB`					
		1%	5%	20%	40%
$\|A\| = 4$	ssPLSA-mem Soft	60.25 ± 0.64	72.97 ± 0.24	79.84 ± 0.96	79.57 ± 0.26
	ssPLSA-mem Hard	49.25 ± 0.73	70.03 ± 0.63	79.61 ± 0.52	79.52 ± 0.17
	ssPLSA-fake	44.42 ± 0.78	59.76 ± 0.84	68.94 ± 0.79	72.63 ± 0.59
$\|A\| = 16$	ssPLSA-mem Soft	$\mathbf{60.56 \pm 0.29}$	$\mathbf{73.67 \pm 0.33}$	$\mathbf{80.56 \pm 0.42}$	$\mathbf{80.94 \pm 0.72}$
	ssPLSA-mem Hard	49.84 ± 0.67	70.85 ± 0.51	80.67 ± 0.32	$\mathbf{80.83 \pm 0.49}$
	ssPLSA-fake	47.97 ± 0.87	62.76 ± 0.58	70.65 ± 0.86	73.78 ± 0.34

Table 6 .5: F-score for varying proportions of labeled-unlabeled training data, for the three variants of the semi-supervised PLSA (ssPLSA-fake, ssPLSA-mem Hard, ssPLSA-mem Soft) and different numbers of the latent topics $|A|$. Bold indicates statistically better results, measured using a t-test at the 5% significance level.

For the `XLS` dataset, as we mentioned before, we compare the semi-supervised models with the current supervised version of the Categorix System. The latter uses the supervised PLSA model. We did not performed any comparison with semi-supervised Naives Bayes and TSVM as we did for the other datasets. As we mentioned before, the proposed models are developped

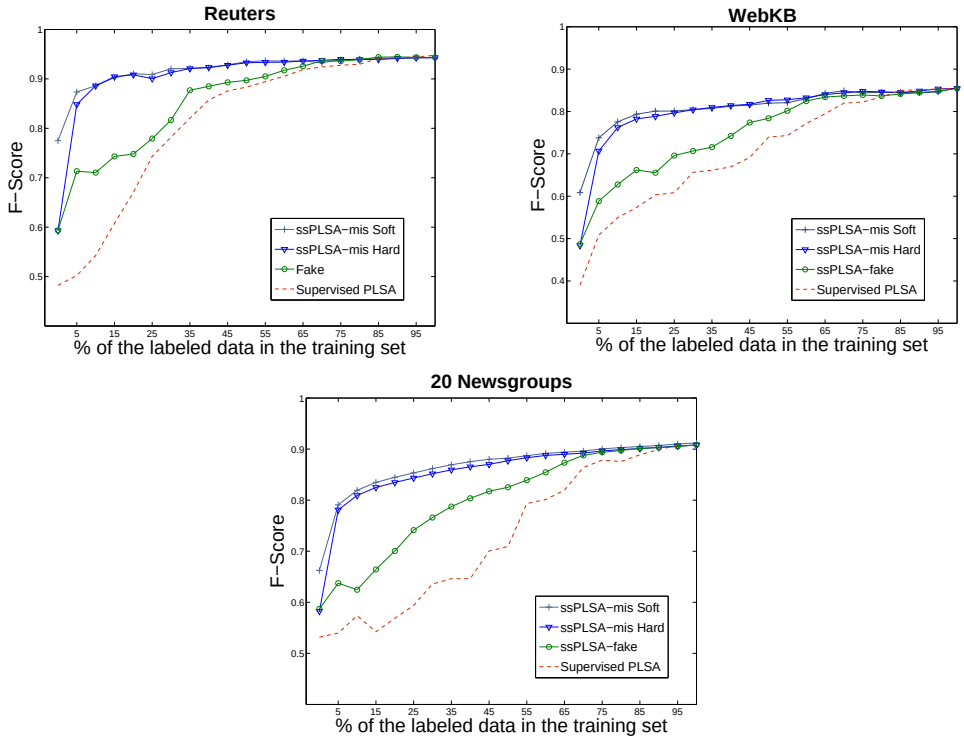

Figure 6 .6: F-Score (y-axis) versus, the percentage of labeled examples in the training set $|X_l|$, (x-axis) graphs for the various algorithms on `20Newsgroups`, `WebKB` and `Reuters`.

as extensions of the current version of the Categorix System. As there are no intentions, at least for the moment, to replace it with a completely different System, we are interrested in the comparison with the current model of this System, instead of comparing with any other model.

In figure 6 .7 we can see the results for different ratio of labeled-unlabeled examples. As we can notice, the semi-supervised learning perform better, especially when very few labels are available. After 10% of labeled examples, the performance of all classifiers does not change much. These results corroborate the conclusions we reached over the previous three datasets: semi-supervised learning can eventually help and the *ssPLSA-mem Soft* outperforms the oth-

ers at all times.

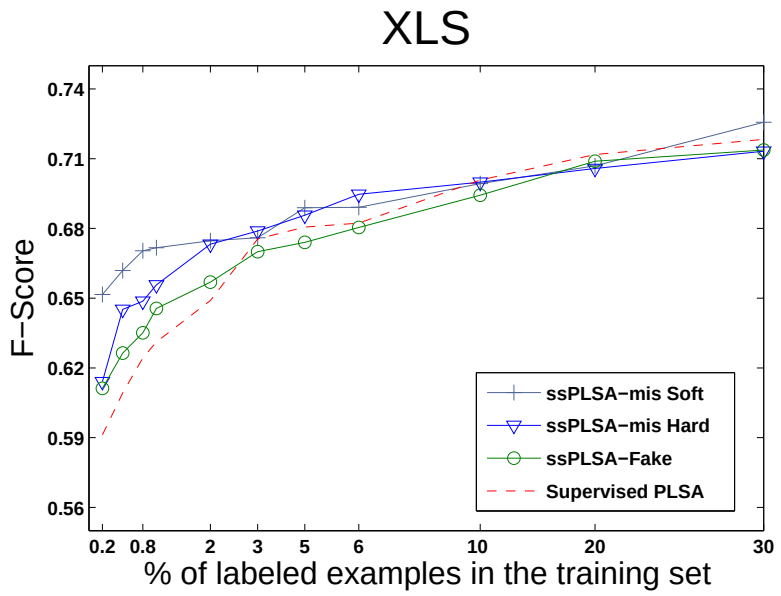

Figure 6 .7: F-Score (y-axis) versus, the percentage of labeled examples in the training set $|X_l|$, (x-axis) graphs for supervised and all semi-supervised variants of PLSA algorithm on `XLS` dataset.

In table 6 .6, we compare *ssPLSA-mem Soft* and *ssPLSA-mem Hard*, in order to give a better insight for the gain of the former method, especially when the number of labeled examples is relatively small. As we can notice, for all four datasets, the *ssPLSA-mem Soft* outperforms the *ssPLSA-mem Hard* when very few labeled data are available in the training set.

6 .5.2 Active ssPLSA Results

In this section we present the results of the combination of semi-supervised and active learning. As introduced in chapter 5, the idea is to perform active learning on the top of the semi-supervised algorithms described above. We run experiments for all semi-supervised variants, for both active learning techniques, and for all four datasets. In our experiments, we label one exam-

ratio	Algorithm	20Newsgroups F-score	WebKB F-score	Reuters F-score	XLS F-score
0.3%	ssPLSA-mem hard	32.62	38.82	47.76	61.41
	ssPLSA-mem soft	**44.05**	**48.78**	**66.34**	**65.16**
0.5%	ssPLSA-mem hard	41.26	40.86	52.02	64.52
	ssPLSA-mem soft	**52.46**	**51.55**	**68.74**	**66.19**
0.8%	ssPLSA-mem hard	51.2	44.16	57.42	64.87
	ssPLSA-mem soft	**60.62**	**56.33**	**75.11**	**67.04**
1%	ssPLSA-mem hard	58.24	49.84	66.93	65.57
	ssPLSA-mem soft	**66.23**	**60.56**	**77.53**	**67.17**

Table 6 .6: Comparison of the two variants of ssPLSA with a mislabeling error model (*ssPLSA-mem Hard ssPLSA-mem Soft*) on `20Newsgroups`, `WebKB`, `Reuters` and `XLS` test sets, trained on different ratio of labeled-unlabeled data

ple in each iteration and 100 iterations are performed for `WebKB`, `Reuters` and 150 for `20Newsgroups` dataset. For the `XLS` dataset we label 2 examples in each iteration, and we perform 100 iterations (as the dataset is bigger than the other three we need more data for achieving a good performance). As we mentioned in chapter 5, for the *Margin Method*, it is not wise to choose 2 examples that are next to each other, as they cannot gives us more information that each of them does. As a result, we chose the two examples with the biggest class-entropy but in addition with different assigned labels.

In order to evaluate the performance of the active learning methods, we also run experiments for the combination of the semi-supervised algorithms with a random selection method, where in each iteration the documents to be labeled are chosen at random.

As we can notice from the figure 6 .8 the use of active learning helps, in comparison with the random query for all three datasets. The performance of the two different active learning techniques are comparable, and their difference is not statistically significant. Nevertheless, they clearly outperfom the random method, especially when very few labeled data are available.

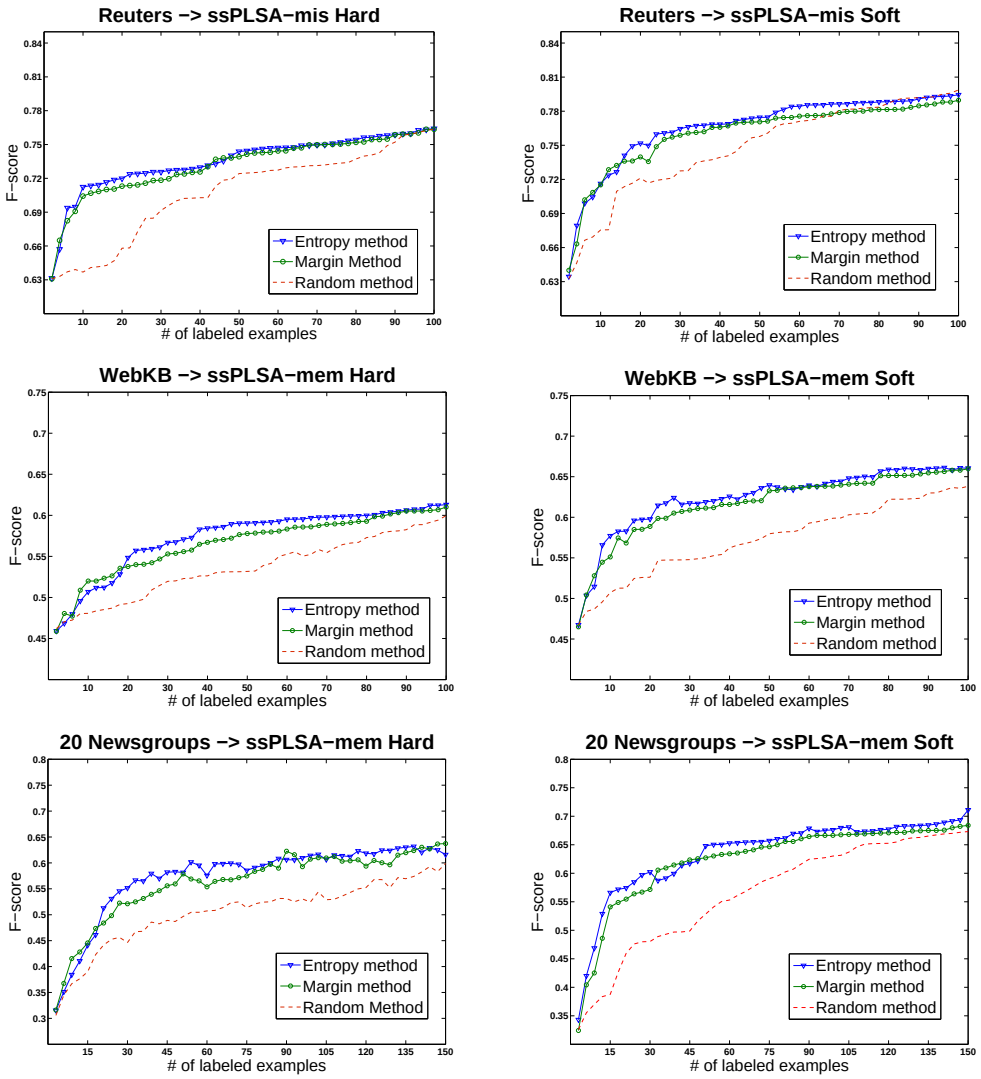

Figure 6 .8: F-Score (y-axis) versus, the number of labeled examples in the training set $|D_l|$, (x-axis) graphs for the combination of the two ssPLSA algorithms with active learning on `Reuters`, `WebKB` and `20Newsgroups` datasets

The results with the ssPLSA-fake gave us the same indications. A comparison of the latter with the other two semi-supervised variants using each of the two active learning methods is presented in figure (6 .9). We can see

that the performance of *active ssPLSA-mem Soft* is constantly better than *active ssPLSA-mem Hard* and *active ssPLSA-fake*.

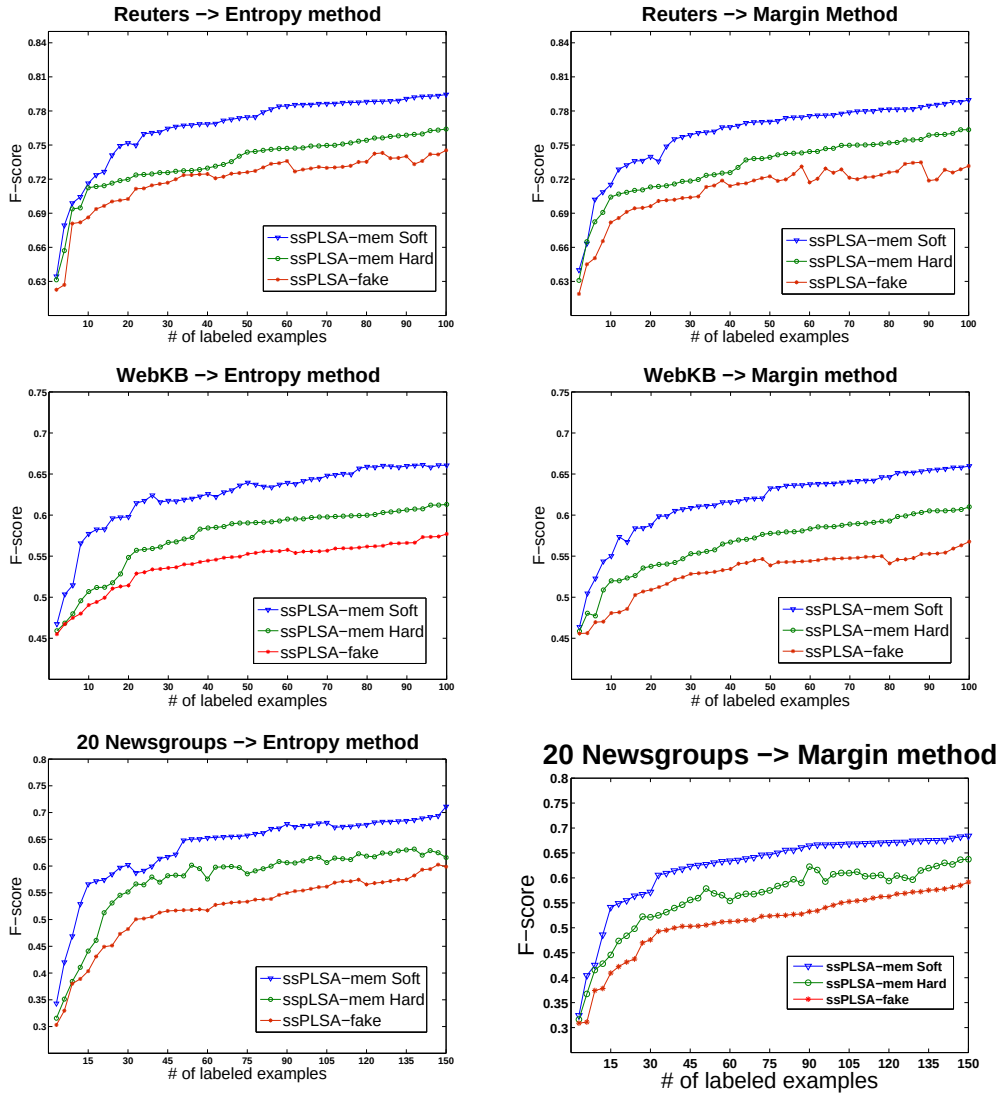

Figure 6 .9: Comparison of the three ssPLSA algorithms using the two different active learning algorithms, on `Reuters`, `WebKB` and `20Newsgroups` dataset

For the `XLS` dataset, the algorithms show again a similar behavior. In

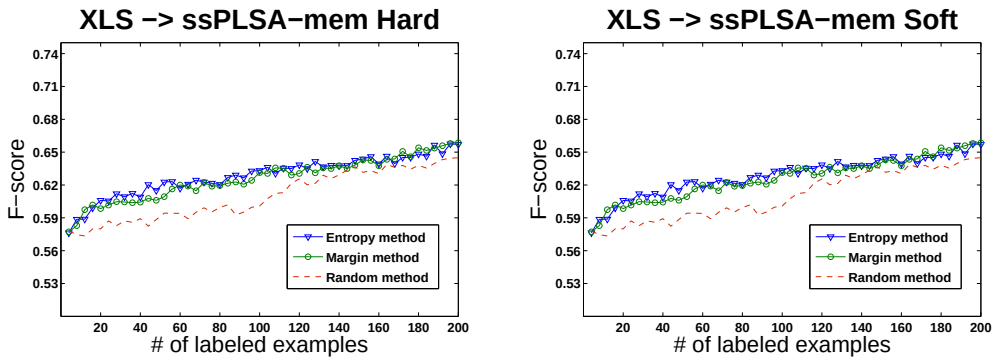

Figure 6 .10: Comparison of the two different active learning techniques and the Random selection, on the ssPLSA-Hard and ssPLSA-Soft algorithms, on `XLS` dataset

figure 6 .10 the results for the latter are presented. As we can notice, active learning helps, comparing to the random method, even if the gain is not as big as in the other three datasets. As before, the two active learning methods give similar results. In 6 .11 the comparison between the three semi-supervised PLSA variants combined with each of the active methods is presented. Also, in this case, the *active ssPLSA-mem Soft* has the better performance.

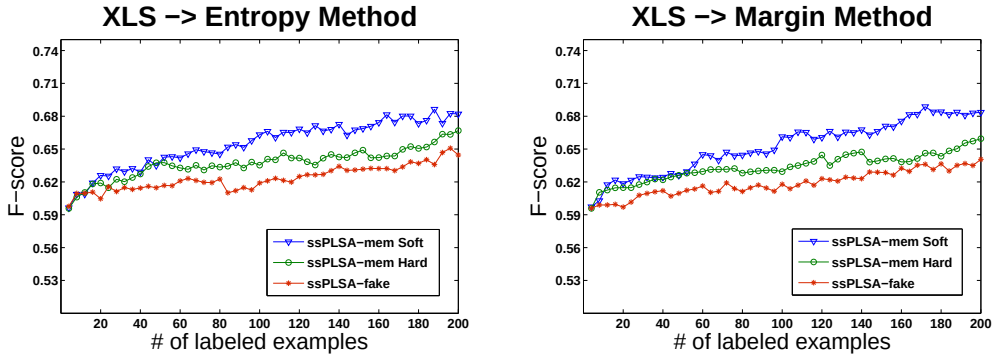

Figure 6 .11: Comparison of the three ssPLSA algorithms using the two different active learning algorithms, on `XLS` dataset

6 .6 Conclusion

In this chapter the evaluation of all the proposed models has been presented and discussed. We saw that the semi-supervised learning can help when very few labeled examples are available. We compared the models with some state-of-the-art algorithms and the results indicate that the *ssPLSA-mem Soft* is the more performant model for all four datasets. The combination with active learning is also evaluated. We compared the two active learning techniques with a random method (where the examples are chosen by random instead of using a active method). The results obtained have proven that active learning can eventually help, especially when the training set contains very few labeled examples.

7

Conclusions

Contents

7 .1 Contributions 113
7 .2 Future Perspectives 114

7 .1 Contributions

This study was motivated by the cost of labeling document collections and the ability of aspect models to explain the generation of textual observations. We focused on semi-supervised and active learning for the task of document classification. This thesis was realized in the framework of a Cifre grant in Xerox Research Centre. We studied the possible extensions of the current classification system and new models were proposed.

In the first part of this thesis, we presented a literature review of existing state-of-the-art methods. We focused on the frameworks of Semi-Supervised Learning, Mislabeling Error Models and Active Learning. The different approaches of these frameworks and their motivation have been discussed.

In the second part, we presented and evaluated the proposed models.

In particular, the contributions of this thesis include:

- Two semi-supervised variants of the PLSA algorithm. The motivation was to take advantange of both the huge amount of available unlabeled data and the properties of aspect models. Our results have proved our initial intuition, that aspect models can benefit from the unlabeled examples. In addition, the incorporation of a model which can capture the mislabeled examples can ameliorate more the performance of our classifier.

- Combination of the above semi-supervised variants with two different active learning techniques. We wanted to benefit from the properties of both frameworks. The evaluation we performed has shown that this combination can further increase classifier's performance. Using active learning we manage to chose our training labeled set carefully, using the most informative results. That way, we can achieve a better performance using less labeled examples.

7 .2 Future Perspectives

This thesis was focused on the PLSA model. Nevertheless, this does not mean that the developped models can exclusively used with the latter. On the contrary, the proposed techniques are very easily applicable to different aspect models, such as Latent Dirichlet allocation (LDA) (Blei et al., 2003). It would be interesting to see how the latter would perform under the framework of semi-supervised and active learning, by incorporating the proposed mislabeling error model.

Another possible extension is the use of different active learning techniques. Also, the combination of more than one active learning technique could be considered.

A different axis of research includes the further investigation of determining the number of components. As we discussed in the evaluation chapter,

this is an important parameter in aspect models. To the best of our knowledge, there has been little effort so far to solve this issue.

The domains of semi-supervised and active learning have still many open problems and further research on the several open problems could be proved fruitful, both in theoritical bases as well as in practical applications.

Bibliography

Agrawala A. K. Learning with a probabilistic teacher. *IEEE Transactions on Information Theory*, 16:373–379, 1970.

Aitchison J. and Begg C. B. Statistical diagnosis when basic cases are not classified with certainty. *Biometrika*, 63(1):1–12, 1976.

Ambroise C. and Govaert G. EM for partially known labels. In *Proceedings of the* 7^{th} *International Federation of Classification Societies*, pages 161–166, Namur, Belgium, 2000.

Amini M. R. and Gallinari P. Semi-supervised learning with explicit misclassification modeling. In *Proceedings of the* 18^{th} *International Joint Conferences on Artificial Intelligence (IJCAI)*, pages 555–560, 2003.

Balcan M.-F., Beygelzimer A., and Langford J. Agnostic active learning. In *Proceedings of the* 23^{rd} *international conference on Machine learning (ICML)*, pages 65–72, New York, NY, USA, 2006.

Balcan M.-F., Broder A. Z., and Zhang T. Margin based active learning. In *Proceedings of the* 20^{th} *Annual Conference on Learning Theory (COLT)*, pages 35–50, San Diego, CA, USA, 2007a.

Balcan M.-F., Even-Dar E., Hanneke S., Kearns M., Mansour Y., and Wortman J. Asymptotic active learning. In *Proceedings of NIPS 2007 Workshop on Principles of Learning Design Problem.*, 2007b.

Balcan N., Bluem A., and Yang K. Co-training and expansion: Towards

bridging theory and practice. In *Advances in Neural Information Processing Systems (NIPS)*, 2004.

Basu S., Banerjee A., and Mooney R. J. Semi-supervised clustering by seeding. In *Proceedings of the* 19^{th} *International Conference on Machine Learning (ICML)*, pages 27–34, San Francisco, CA, USA, 2002.

Basu S., Bilenko M., Banerjee A., and Mooney R. *Probabilistic Semi-Supervised Clustering with constraints*, chapter 5. In Semi-Supervised Learning. O. Chapelle and B. Schölkopf and A. Zien. MIT Press, Cambridge, MA, 2006.

Belkin M. and Niyogi P. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural Computation*, 15(6):1373–1396, 2003.

Blei D., Ng A., and Jordan M. Latent dirichlet allocation. *Journal of Machine Learning Research*, 2003.

Blum A. and Chawla S. Learning from labeled and unlabeled data using graph mincuts. In *Proceedings of the* 18^{th} *International Conference on Machine Learning (ICML)*, pages 19–26. Morgan Kaufmann, San Francisco, CA, 2001.

Blum A., Lafferty J., Rwebangira M.R., and Reddy R. Semi-supervised learning using randomized mincuts. In *Proceedings of the* 21^{st} *International Conference on Machine Learning (ICML)*, page 13, 2004.

Blum A. and Mitchell T. Combining labeled and unlabeled data with co-training. In *Proceedings of the Conference on Computational Learning Theory (COLT),*, pages 92–100, 1998.

Bordes A., Ertekin S., Weston J., and Bottou L. Fast kernel classifiers with online and active learning. *Journal of Machine Learning Research*, 6:1579–1619, September 2005.

Brodley C. E. and Friedl M. A. Identifying mislabeled training data. *Journal of Artificial Intelligence Research (JAIR)*, 11:131–167, 1999.

Burges C. and Platt J. *Semi-supervised learning with conditional harmonic mixing*, chapter 14. In Semi-Supervised Learning. O. Chapelle and B. Schölkopf and A. Zien. MIT Press, Cambridge, MA, 2006.

Campbell C., Cristianini N., and Smola A. J. Query learning with large margin classifiers. In *Proceedings of the* 17^{th} *International Conference on Machine Learning (ICML)*, pages 111–118, San Francisco, CA, USA, 2000.

Castelli V. and Cover T.M. On the exponential value of labeled samples. *Pattern Recognition Letters*, 16(1):105–111, 1995.

Castelli V. and Cover T.M. The relative value of labeled and unlabeled samples in pattern recognition with an unknown mixing parameter. *IEEE Transactions on Information Theory*, 42(6):2102–2117, 1996.

Castro R. and Nowak R. D. Minimax bounds for active learning. In *Proceedings of the* 20^{th} *Annual Conference on Learning Theory (COLT)*, pages 5–19, San Diego, CA, USA, 2007.

Chapelle O., Chi M., and Zien A. A continuation method for semi-supervised svms. In *Proceedings of the* 23^{rd} *international conference on Machine learning (ICML)*, pages 185–192, 2006.

Chapelle O. and Zien A. Semi-supervised classification by low density separation. In *Proceedings of the Tenth International Workshop on Artificial Intelligence and Statistics*, 2005.

Chawla N. V. and Bowyer K. W. Actively exploring creation of face space(s) for improved face recognition. In *Proceedings of the* 22^{nd} *Conference on Artificial Intelligence*, pages 809–814, Vancouver ,Canada, 2007.

Chhikara R.S. and McKeon J. Linear discriminant analysis with misallocation in training samples. *Journal of the American Statistical Association*, 79: 899–906, 1984.

Chittineni C. B. Learning with imperfectly labeled patterns. *Pattern Recognition*, 12(5):281–291, 1980.

Cohn D., Caruana R., and McCallum A. Semi-supervised clustering with user feedback. Technical Report TR2003-1892, Cornell University, 2003.

Cohn D., Ghahramani Z., and Jordan M. Active learning with statistical models. In *Advances in Neural Information Processing Systems*, volume 7, pages 705–712, 1995.

Cohn D. A., Atlas L., and Ladner R. E. Improving generalization with active learning. *Machine Learning*, 15(2):201–221, 1994.

Collins M. and Singer Y. Unsupervised models for named entity classification. In *Proceeding of of the Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora.*, 1999.

Collobert R., Sinz F., Weston J., and Bottou L. Large scale transductive svms. *Journal of Machine Learning Research*, 7:1687–1712, September 2006.

Cooper S., Hertzmann A., and Popović Z. Active learning for real-time motion controllers. *ACM Transactions on Graphics (TOG)*, 26(3):5, 2007.

Cox T. and Cox M. *Multidimensional Scaling.* Chapman & Hall, London, 1994.

Cozman F.G., Cohen I., and Cirelo M.C. Semi-supervised learning of mixture models. In *Proceedings of the* 20^{th} *International Conference on Machine Learning (ICML)*, pages 99–106, Washington DC, USA, 2003.

Craven M., DiPasquo D., Freitag D., McCallum A.K., Mitchell T.M., Nigam K., and Slattery S. Learning to extract symbolic knowledge from the World Wide Web. In *Proceedings of the* 15^{th} *Conference of the American Association for Artificial Intelligence*, pages 509–516, Madison, US, 1998.

Dagan I. and Engelson Sean P. Committee-based sampling for training probabilistic classifiers. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 150–157, 1995.

Davy M. and Luz S. Active learning with history-based query selection for text categorisation. In *Procceedings of the European Conference on Information Retrieval (ECIR)*, volume 4425, pages 695–698, 2007.

De Bie T. and Cristianini N. Convex methods for transduction. In *Advances in Neural Information Processing Systems 16 (NIPS). MIT Press*, Cambridge, MA, 2004.

De Sa V. Learning classification with unlabeled data. In *Neural Information Processing Systems (NIPS)*, pages 112–119, San Francisco, CA, 1993. Morgan Kaufmann Publishers.

De Sa V. *Unsupervised Classification Learning from Cross-Modal Environmental Structure*. PhD thesis, University of Rochester, 1994.

Deerwester S., Dumais S.T., Furnas G.W., Landauer T.K., and Harshman R. Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 41(6):391–407, 1990.

Demiriz A., Bennett K., and Embrechts M. Semi-supervised clustering using genetic algorithms. In *Proceedings of Artificial Neural Networks in Engineering*, pages 809–814, 1999.

Dempster A. P., Laird N. M., and Rubin D. B. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, 39(1):1–38, 1977.

Denoeux T. A k-nearest neighbor classification rule based on dempster-shafer theory. *IEEE Transactions on Systems, Man, and Cybernetics (SMC)*, 25 (5):804–813, 1995.

Dönmez P., Carbonell J. G., and Bennett P. N. Dual strategy active learning. In *Proceedings of the European Conference on Machine Learning (ECML)*, pages 116–127, 2007.

Doyle P. and Snell J. *Random walks and electric networks*. Mathematical Association of America, Washington, 1984.

Ertekin S., Huang J., Bottou L., and Giles L. Learning on the border: active learning in imbalanced data classification. In *Proceedings of the* 16^{th} *ACM conference on Conference on information and knowledge management (CIKM)*, pages 127–136, 2007.

Freund Y. and Schapire R.E. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119–139, 1997.

Freund Y., Seung H.S., Shamir E., and Tishby N. Selective sampling using the query by committee algorithm. *Machine Learning*, 28(2-3):133–168, 1997.

Gaussier E. and Goutte C. Learning from partially labelled data - with confidence. In *Proceedings of the ICML workshop in Learning with Partially Classified Training Data*, Bonn, Germany, 2005.

Goldman S. A. and Zhou Y. Enhancing supervised learning with unlabeled data. In *Proceedings of the* 17^{th} *International Conference on Machine Learning (ICML)*, pages 327–334, San Francisco, CA, USA, 2000.

Grandvalet Y. and Bengio Y. Semi-supervised learning by entropy minimization. In *Advances in Neural Information Processing Systems 17 (NIPS)*, pages 529–536, 2005.

Hagen L. and Kahng A. New spectral methods for ratio cut partitioning and clustering. *IEEE Transactions on CAD*, 11:1074–1085, 1992.

Hakkani-Tür D., Riccardi G., and Tur G. An active approach to spoken language processing. *ACM Transactions on Speech and Language Processing (TSLP)*, 3(3):1–31, 2006.

Hanneke S. A bound on the label complexity of agnostic active learning. In *Proceedings of the* 24^{th} *international conference on Machine learning (ICML)*, pages 353–360, New York, NY, USA, 2007a.

Hanneke S. Teaching dimension and the complexity of active learning. In *Proceedings of the* 20^{th} *Annual Conference on Learning Theory (COLT)*, pages 66–81, San Diego, CA, USA, 2007b.

Hayes P. J., Andersen P. M., Nirenburg I. B., and Schmandt L. M. Tcs: a shell for content-based text categorization. In *Proceedings of the* 6^{th} *conference on Artificial intelligence applications*, pages 320–326, Piscataway, NJ, USA, 1990.

Hofmann T. Unsupervised learning by probabilistic latent semantic analysis. *Machine Learning*, 42(1-2):177–196, 2001.

Iyengar V.S., Apte C., and Zhang T. Active learning using adaptive resampling. In *Proceedings of the* 6^{th} *International Confenrence on Knowledge Discovery and Data Mining*, pages 92–98, 2000.

Joachims T. Transductive inference for text classification using support vector machines. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 200–209, Bled, Slowenien, 1999.

Joachims T. Transductive learning via spectral graph partitioning. In *Proceeding of the* 20^{th} *International Conference on Machine Learning (ICML)*, 2003.

John G. H. Robust decision trees: Removing outliers from databases. In *Proceedings of the* 1^{st} *International conference on Knowledge Discovery and Data Mining*, pages 174–179, Montreal, Quebec, 1995.

Jolliffe I.T. *Principal Component Analysis*. Springer-Verlag, 1986.

Karmaker A. and Kwek S. A boosting approach to remove class label noise. In *Proceedings of the Fifth International Conference on Hybrid Intelligent Systems (HIS)*, pages 206–211, Washington, DC, USA, 2005.

Koller D. and Sahami M. Hierarchically classifying documents using very few words. In *Proceedings of the Fourteenth International Conference on Machine Learning (ICML)*, pages 170–178, San Francisco, CA, USA, 1997.

Krause A. and Guestrin C. Nonmyopic active learning of gaussian processes: an exploration-exploitation approach. In *Proceedings of the* 24^{th} *international conference on Machine learning (ICML)*, pages 449–456, Corvalis, Oregon, 2007.

Krishnan T. Efficiency of learning with imperfect supervision. *Pattern Recognition*, 21(2):183–188, 1988.

Krishnan T. and Nandy S. C. Discriminant analysis with a stochastic supervisor. *Pattern Recognition*, 20(4):379–384, 1987.

Krithara A., Goutte C., Amini M.-R., and Renders J.-M. Reducing the annotation burden in text classification. In *1st International Conference on Multidisciplinary Information Sciences and Technologies (InSCiT)*, Merida, Spain, 2006.

Kullback S. and Leibler R. A. On information and sufficiency. *The Annals of Mathematical Statistics*, 22(1):79–86, 1951.

Kuo J.-S., Li H., and Yang Y.-K. Active learning for constructing transliteration lexicons from the web. *Journal of the American Society for Information Science and Technology*, 59(1):126–135, 2008.

Lachenbrunch P.A. Discriminant analysis when the initial samples are misclassified. ii: Non-random misclassification models. *Technometrics*, 16:419–424, 1974.

Lawrence N.D. and Scholkopf B. Estimating a kernel fisher discriminant in the presence of label noise. In *Proceedings of the* 8^{th} *International Conference on Machine Learning (ICML)*, pages 306–313, San Francisco, CA, USA, 2001.

Lewis D. Naive (bayes) at forty: The independence assumption in information retrieval. In *Proceedings of the* 10^{th} *European Conference on Machine Learning (ECML)*, pages 4–15, London, UK, 1998.

Lewis D. and Gale W. A. A sequential algorithm for training text classifiers. In *Proceedings of the* 17^{th} *International Conference on Research and Development in Information Retrieval (SIGIR)*, pages 3–12, Dublin, 1994.

Lewis D. and Ringuette M. A comparison of two learning algorithms for text categorization. In *Proceedings of the Symposium on Document Analysis and Information Retrieval (SDAIR)*, pages 81–93, 1994.

Li Y., Wessels L., and Reinders M. Class-noise tolerant classification based on a probabilistic noise model. In *Proceedings of the* 12^{th} *annual conference of the Advanced School for Computing and Imaging*, 2006.

Lomsadze A., Hovhannisyan Ter V., Chernoff Y. O., and Borodovsky M. Gene identification in novel eukaryotic genomes by self-training algorithm. *Nucleic Acids Res*, 33(20):6494–6506, 2005.

Lugosi G. Learning with an unreliable teacher. *Pattern Recogn.*, 25(1):79–87, 1992.

McCallum A. and Nigam K. Employing EM and pool-based active learning for text classification. In *Proceedings of the* 15^{th} *International Conference on Machine Learning (ICML)*, pages 350–358, 1998.

McLachlan G. J. Asymptotic results for discriminant analysis when the initial samples are misclassified. *Technometrics*, 14:415–422, 1972.

McLachlan G. J. *Discriminant analysis and Statistical Pattern Recognition.* John Wiley & Sons, Inc. New York, 1992.

Miller D. and Uyar H. A mixture of experts classifier with learning based on both labeled and unlabeled data. In *Advances in Neural Information Processing Systems 9 (NIPS)*, pages 571–577, Cambridge, MA, 1997.

Muslea I., Minton S., and Knoblock C. Active + semi-supervised learning = robust multi-view learning. In *Proceedings of the* 19^{th} *International Conference on Machine Learning (ICML)*, pages 435–442, 2002.

Muslea I., Minton S., and Knoblock C. A. Selective sampling with redundant views. In *Proceedings of the* 17^{th} *National Conference on Artificial Intelligence (AAAI)*, pages 621–626, 2000.

Nigam K. and Ghani R. Analyzing the effectiveness and applicability of co-training. In *Proceedings of the* 9^{th} *international Conference on Information and knowledge Management (CIKM)*, pages 86–93, 2000.

Nigam K., McCallum A. K., Thrun S., and Mitchell T. M. Text classification from labeled and unlabeled documents using EM. *Machine Learning*, 39 (2/3):103–134, 2000.

O'Neill T.J. Normal discrimination with unclassified observations. *American Statistical Association*, 73(36):821–826, 1978.

Probst K. and Ghani R. Towards 'interactive' active learning in multi-view feature sets for information extraction. In *Proceedings of European Conference on Machine Learning (ECML)*, pages 683–690, 2007.

Quinlan J. R. *C4.5: programs for machine learning*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1993.

Rabiner L. R. A tutorial on hidden markov models and selected applications in speech recognition. *Readings in speech recognition*, pages 267–296, 1990.

Ratsaby J. and Venkatesh S. Learning from a mixture of labeled and unlabeled examples with parametric side information. In *Proceedings of the* 8^{th} *annual conference on Computational learning theory (COLT)*, pages 412–417, 1995.

Rosenberg C., Hebert M., and Schneiderman H. Semi-supervised self-training of object detection models. In *Proceedings of the* 7^{th} *IEEE Workshops on Application of Computer Vision*, Washington, DC, USA, 2005.

Roweis S. T. and Saul L. K. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290:2323–2326, 2000.

Roy N. and McCallum A. Toward optimal active learning through sampling estimation of error reduction. In *Proceedings of the* 18^{th} *International Conference on Machine Learning (ICML)*, pages 441–448, San Francisco, CA, 2001.

Salton G. and Buckley C. Term-weighting approaches in automatic text retrieval. *Information Processing and Management*, 24(5):513–523, 1988.

Saul L. and Pereira F. Aggregate and mixed-order markov models for statistical language processing. In *Proceedings of the Second Conference on Empirical Methods in Natural Language Processing*, pages 81–89, Somerset, New Jersey, 1997.

Saul L. K. and Roweis S. T. Think globally, fit locally: Unsupervised learning of low dimensional manifolds. *Journal of Machine Learning Research*, 4: 119–155, 2003.

Schapire R. E. and Singer Y. Improved boosting algorithms using confidence-rated predictions. *Machine Learning*, 37(3):297–336, 1999.

Schapire R. E. and Singer Y. Boostexter: A boosting-based systemfor text categorization. *Machine Learning*, 39(2-3):135–168, 2000.

Schohn G. and Cohn D. Less is more: Active learning with support vector machines. In *Proceedings of the* 17^{th} *International Conference on Machine Learning (ICML)*, pages 839–846, San Francisco, CA, USA, 2000.

Scudder H. J. Probability of error of some adaptive pattern-recognition machines. *IEEE Transactions on Information Theory*, 11:363–371, 1965.

Sebastiani F. Machine learning in automated text categorization. *ACM Computing Surveys*, 34(1):1–47, 2002.

Seung H. S., Opper Manfred, and Sompolinsky Haim. Query by committee. *Computational Learning Theory*, pages 287–294, 1992.

Sindhwani V., Keerthi S. Sathiya, and Chapelle O. Deterministic annealing for semi-supervised kernel machines. In *Proceedings of the* 23^{rd} *international conference on Machine learning (ICML)*, pages 841–848, 2006.

Smets P. The transferable belief model. *Artificial Intelligence*, 66(2):191–234, 1994.

Spragins J. Learning without a teacher. *IEEE Transactions on Information Theory*, 12:223–230, 1966.

Sun J., Boyd S., Xiao L., and Diaconis P. The fastest mixing markov process on a graph and a connection to a maximum variance unfolding problem. *SIAM Review*, 2006.

Symons M. J. Clustering criteria and multivariate normal mixture. *Biometrics*, 37(1):35–43, 1981.

Szummer M. and Jaakkola T. Partially labeled classification with markov random walks. In *Advances in Neural Information Processing Systems*, volume 14, 2002.

Tenenbaum J. B., Silva de V., and Langford J. C. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290:2319–2323, 2000.

Titterington D.M. An alternative stochastic supervisor in discriminant analysis. *Pattern Recognition*, 22(1):91–95, 1989.

Tong S. and Koller D. Support vector machine active learning with applications to text classification. In *Proceedings of* 17^{th} *International Conference on Machine Learning (ICML)*, pages 999–1006, Stanford, US, 2000.

Tür G., Hakkani-Tür D., and Schapire R. Combining active and semi-supervised learning for spoken language understanding. *Speech Communication*, 45(2):171–186, 2005.

Van Hulse J. D., Khoshgoftaar T. M., and Huang H. The pairwise attribute noise detection algorithm. *Knowledge and Information Systems*, 11(2):171–190, 2007.

Vandenberghe L. and Boyd S. Semidefinite programming. *SIAM Review*, 38 (1):49–95, 1996.

Vapnik V. *Estimation of Dependences Based on Empirical Data: Springer Series in Statistics*. Springer-Verlag New York, Inc., 1982.

Vapnik V. *Statistical Learning Theory*. Wiley, 1998.

Vittaut J.-N., Amini M.-R., and Gallinari P. Learning classification with both labeled and unlabeled data. In *Proceedings of the 13th European Conference on Machine Learning (ECML)*, pages 468–476, 2002.

Vogiatzis D. and Tsapatsoulis N. Active learning for microarray data. *International Journal of Approximate Reasoning*, 47(1):85–96, 2008.

Wagstaff K., Cardie C., Rogers S., and Schrödl S. Constrained k-means clustering with background knowledge. In *Proceedings of the* 18^{th} *International Conference on Machine Learning (ICML)*, pages 577–584, San Francisco, CA, USA, 2001.

Wang F. and Zhang C. Label propagation through linear neighborhoods. *IEEE Transactions on Knowledge and Data Engineering*, 20(1):55–67, 2008.

Weinberger K. Q. and Saul L. K. Unsupervised learning of image manifolds by semidefinite programming. *International Journal of Computer Vision*, 70(1):77–90, 2006.

Widrow B. and Stearns S. D. *Adaptive signal processing*. Prentice-Hall, 1985.

Xing E., Ng A., Jordan M., and Russell S. Distance metric learning, with application to clustering with side-information. In *Advances in Neural Information Processing Systems 15 (NIPS)*, pages 505–512, Cambridge, MA, 2003.

Yarowsky D. Unsupervised word sense disambiguation rivaling supervised methods. In *Proceedings of the* 33^{rd} *annual meeting on Association*

for Computational Linguistics, pages 189–196, Cambridge, Massachusetts, 1995.

Yuille A. L. and Rangarajan A. The concave-convex procedure (cccp). In *Advances in Neural Information Processing Systems 14 (NIPS)*, Cambridge, MA, 2002.

Zhou D., Bousquet O., Lal T., Weston J., and Schölkopf B. Learning with local and global consistency. In *Proceedings of the* 16^{th} *Annual Conference on Neural Information Processing Systems (NIPS)*, 2004.

Zhou D., Huang J., and Schölkopf B. Learning from labeled and unlabeled data on a directed graph. In *Proceedings of the* 22^{nd} *international conference on Machine learning (ICML)*, pages 1036–1043, 2005.

Zhou Z.-H., Chen K.-J., and Dai H.-B. Enhancing relevance feedback in image retrieval using unlabeled data. *ACM Transactions on Information Systems*, 24(2):219–244, 2006.

Zhu X. *Semi-Supervised Learning with Graphs*. PhD thesis, Carnegie Mellon University, 2005.

Zhu X. and Ghahramani Z. Learning from labeled and unlabeled data with label propagation. Technical Report CMU-CALD-02-107, Carnegie Mellon University, Pittsburgh, 2002.

Zhu X., Ghahramani Z., and Lafferty J.D. Semi-supervised learning using gaussian fields and harmonic functions. In *Proceedings of the* 20^{th} *international conference on Machine learning (ICML)*, pages 912–919, 2003a.

Zhu X., Lafferty J., and Ghahramani Z. Combining active learning and semi-supervised learning using gaussian fields and harmonic functions. In *Proceedings of the* 20^{th} *International Conference on Machine Learning*, 2003b.