Machine Learning Approaches to Text Representation using Unlabeled Data

THÈSE N^o 3676 (2006)

PRESENTÉE À LA FACULTÉ SCIENCES ET TECHNIQUES DE L'INGÉNIEUR

École Polytechnique Fédérale de Lausanne

pour l'obtention du grade de docteur ès sciences

par

Mikaela Keller

DEA de Modélisation Stochastique et Statistique,

Université de Paris-Sud (XI), France

acceptée sur proposition du jury:

Prof. Hervé Bourlard, directeur de thèse

Dr. Samy Bengio, co-directeur de thèse

Prof. Florence d'Alché-Buc, rapporteur

Dr. Jean-Cédric Chappelier, rapporteur

Prof. Thomas Hofmann, rapporteur

Lausanne, EPFL

2006

Résumé

Avec l'essor de l'usage des ordinateurs pour la création de documents textuels digitalisés, le besoin de systèmes automatiques pour la recherche et l'organisation de l'information contenue dans de grandes bases de données est devenu central. En général, les systèmes de recherche d'information s'appuient sur une description formelle (ou représentation) des documents permettant leur traitement automatique.

Dans la plus commune des représentations, appelée sac-de-mots, les documents sont représentés par l'ensemble des mots les constituant. Deux documents (ou bien un document et une requête) sont considérés comme similaires s'ils ont un grand nombre de mots en commun.

Il est raisonable de penser que les systèmes de recherche d'information devraient pouvoir utiliser les grandes quantités de données textuelles disponibles pour "apprendre", à la façon des humains, les différents emplois d'un mot en fonction de son contexte. Cette information devrait pouvoir être utilisée pour enrichir la représentation des documents.

Dans cette thèse, nous développons plusieurs approches originales d'apprentissage automatique qui tentent d'atteindre ce but.

Comme première approche pour la représentation de documents nous proposons un modelé probabiliste qui suppose que les documents sont tirés d'un mélange de distributions sur des "thèmes", représentés par une variable cachée qui conditionne une distribution multinomiale sur les mots. Simultanément, ce modèle suppose que les mots sont tirés d'une distribution sur les "sujets", représentés quant à eux par une seconde variable cachée dépendante des thèmes.

Comme deuxième approche, un réseau de neurones est proposé. Il est entraîné à donner un score reflétant le fait qu'un mot soit plus ou moins approprié étant donné un contexte.

Finalement, nous présentons une approche multitâche entraînée de façon à résoudre conjointement une tâche de recherche d'information et tout en enrichissant la représentation des documents par l'utilisation des données non-étiquetées.

Mots-clés

Apprentissage automatique, recherche d'information, extraction de caractéristiques, modèles graphiques, réseaux de neurones, apprentissage multitâche, évaluation de performance, apprentissage semi-supervisé.

Abstract

With the rapid expansion in the use of computers for producing digitalized textual documents, the need of automatic systems for organizing and retrieving the information contained in large databases has become essential. In general, information retrieval systems rely on a formal description or representation of documents enabling their automatic processing.

In the most common representation, the so-called bag-of-words, documents are represented by the words composing them and two documents (or a user query and a document) are considered similar if they have a high number of co-occurring words. In this representation, documents with different, but semantically related terms will be considered as unrelated, and documents using the same terms but in different contexts will be seen as similar.

It arises quite naturally that information retrieval systems can use the huge amount of existing textual documents in order to "learn", as humans do, the different uses of words depending on the context. This information can be used to enrich documents' representation.

In this thesis dissertation we develop several original machine learning approaches which attempt at fulfilling this aim.

As a first approach to document representation we propose a probabilistic model in which documents are assumed to be issued from a mixture of distributions over themes, modeled by a hidden variable conditioning a multinomial distribution over words. Simultaneously, words are assumed to be drawn from a mixture of distributions over topics, modeled by a second hidden variable dependent on the themes.

As a second approach, we proposed a neural network which is trained to give a score for the appropriateness of a word in a given context.

Finally we present, a multi-task learning approach, which is trained jointly to solve an information retrieval task, while learning on unlabeled data to improve its representation of documents.

Keywords

Machine Learning, Information Retrieval, Feature Extraction, Graphical Model, Neural Networks, Multi-task Learning, Semi-Supervised Learning, Performance Evaluation.

Acknowledgments

I would like to thank all the people who supported me during the realization of this thesis.

In particular, I would like to thank Samy Bengio, who supervised this work, for his good guidance and friendship. I am very grateful to Johnny Mariéthoz, who shared my office, and without whom I might have left our office by the window rather than by the door more than once. My thanks go also to all those people at IDIAP and in the machine learning group who have contributed to making my stay in this institute enriching and beneficial.

I also would like to thank Florence d'Alché for introducing me to machine learning and for the interest she takes in my work.

Of course, this thesis would not have been possible without the affective support of my family and friends. First of all Benjamin, who on several occasions gave me the impulse that I needed to continue. And my scientist parents who have such confidence in me that it often serves as both a pressure and a reassurance. And I should mention my brothers and friends who reminded me cheerfully that life has something else to offer other than PhDs. To all these people, I would like to express my many thanks.

And finally, considering the verb, *to support*, in the french sense, I would like to emphasize my gratitude and apologies toward those who endured the sometimes dramatic ups and downs generated by this thesis process.

Contents

Chapter 1

Introduction

Description of the Problem

More than five hundred years after Gutenberg's press[1] and the revolution in information propagation it triggered off in Europe, we are facing a world-wide increase of the production and diffusion of the human written documents. In the age of these quite recent tools which computers are, every day, millions[2] of textual documents in digital format are published in as diverse sectors of the society as individuals, education, culture, politics, economics, etc.

The use of automatic systems for retrieving and organizing the information contained in these texts, and which usually end up in huge databases, has become extremely generalized and useful. Many people have now made the experience of the usefulness of retrieval systems for navigating through the billions of hyper-linked documents composing the Internet. Information retrieval activities encompass a broader field than hyper-linked document retrieval and whether the documents under consideration are "hyper-linked" or not, information retrieval systems rely on a formal description or representation of documents enabling their automatic processing.

In most information retrieval systems, documents are represented by the words composing them and two documents (or a user query and a document) are considered similar if they have a high number of co-occurring words. These approaches do not tackle the fact that in human languages a certain topic can be in general expressed with different words and in different ways.

It arises quite naturally that information retrieval systems could use the huge amount of existing textual documents in order to "learn", as humans do, the links between words depending on the context.

Machine Learning is a field developed in the last decades at a crossroads of the computer science and statistics domains. It aims at producing systems able to generalize targeted behaviors from examples. It is strongly related to statistics, with an emphasis however towards models efficient in the resolution of

[1] Johannes Gensfleisch zur Laden zum Gutenberg (ca. 1400-1468) achieved fame for his invention of the technology of printing with movable types during 1447. Note: The front picture is a drawing of Gutenberg made after his death (considered fictional). *source: Wikipedia*

[2] Rough estimate based on the fact that as an example there were 50,000,000 blogs (or web logs) in December 2005 in the website Xanga, one of the earliest hosting blogs. *source: Wikipedia*

problems as opposed to models explaining the data. Machine learning techniques have been applied to a broad range of problems including speech and handwriting recognition, object recognition in computer vision, brain-computer interface systems, etc. There have also been several intents at applying machine learning methods for learning richer document representation, however most approaches are very simple or do not scale well to high amount of data.

Overview of Contributions

The remainder of this document is organized as follows, in Chapter 2 the machine learning framework and information retrieval field are introduced. In addition, the current *state-of-the-art* of machine learning approaches for document representation are described.

Chapter 3 presents the datasets on which we tested our approaches and the performance measures commonly used to assess the quality of information retrieval systems such as Precision, Recall, F_1 score and ROC curves. In addition, in this chapter we argue that ROC curves and some of the measures extracted from them can be misleading in a machine learning framework. We describe an alternative to ROC curves, which was previously proposed in

> CONTRIB S. Bengio, J. Mariethoz, and M. Keller. The expected performance curves. In *Proceedings of the ICML 2005 wokshop on ROC Analysis in Machine Learning*, 2005

Finally, we present a benchmarking experiment we did, and which was previously reported in

> CONTRIB M. Keller, S. Bengio, and S.Y. Wong. Benchmarking non-parametric statistical tests. In *Advances in Neural Information Processing Systems, NIPS 18. MIT Press*, 2005

This experiment evaluates the performance of a statistical significance test designed for the F_1 score in several conditions and in comparison to more classical significance tests.

In Chapter 4 we describe a probabilistic model to text representation, previously presented in

> CONTRIB M. Keller and S. Bengio. TTMM: A graphical model for document representation. In *PASCAL Workshop on Text Mining and Understanding*, January 2004b

We compare this model on several points to related approaches.

In Chapter 5 we present a first non-probabilistic approach based on neural network algorithms to tackle the text representation problem. The experiments presented in this chapter were previously reported in

> CONTRIB M. Keller and S. Bengio. A neural network for text representation. In *Proceedings of International Conference in Artificial Neural Networks (ICANN), Warsaw, Poland*, 2005

and

CONTRIB O. Glickman, I. Dagan, M. Keller, S. Bengio, and W. Daelemans. Investigating lexical substitution scoring for subtitle generation. In *Proceedings of CoNLL*, 2006

Finally, Chapter 6 reports an approach in which the representation is learnt jointly with the task to solve. This work has been the subject of the following technical report:

CONTRIB M. Keller and S. Bengio. A multi-task learning approach to document representation using unlabeled data. IDIAP-RR 36, IDIAP, 2006

In order to close this introduction we would like to issue the following disclaimer. Since in this research we explore methods using the relationship between words, we sometimes use terms such as *concept*, *theme* or *topic*. They are used here for convenience in order to express the intuition of semantic links between textual data components. You will not find in this work an attempt at modelling truly the semantic produced by language, as linguistic may define it. These terms are in fact used to refer to high level statistical correlations.

Chapter 2

Overview of Text Representation Methods

For information retrieval systems a representation of the texts preserving or even emphasizing the relationship between the words which shape their meaning is essential.

The representation commonly used for texts in information retrieval described in Section 2.2, relies for each textual document upon the dictionary's words present in it. Except for the choice of the dictionary, this representation does not take into account any relationship between the words other than their joint occurrence in the text. As a consequence, two sentences such as *a ship in the sea* and *a boat in the ocean*, which we know are closely related, will be considered in this representation as highly dissimilar because they do not share words.

Several machine learning approaches propose to model the existing links between words for overcoming this kind of problems. These methods are described in Section 2.3. In several applications, information retrieval and machine learning fields are already well entangled. It is due to the fact that the machine learning framework, described in Section 2.1, allows to build efficient systems capable of modifying their behavior according to their previous experience, which can be very interesting for information retrieval when what is "taught" to the machine is related to the relationship between words.

2.1 Machine Learning Framework

Machine Learning is a domain situated at a crossroad between computer science and statistics. It is concerned with algorithms which are designed to improve their performance through experience. A machine learning "algorithm" for a computer scientist would be a list of instructions with parameters to be given to a computer. The "experience" encounter by the algorithm are "examples" that it takes as input and which for a statistician would be seen as a sample issued from an unknown distribution. The learning process from a statistical point of view is the estimation of the instruction parameters which will make the algorithm more efficient.

2.1.1 Risk Minimization

The machine learning framework as defined by Vapnik (1995) can be formulated as follows. There is a population, described by a distribution with density function $p : \mathcal{Z} \to \mathbb{R}$, on which we want to solve a problem. The assumptions about what the solution should look like are encoded in the choice of an *hypothesis* space $\mathcal{F} \subset \{f : \mathcal{Z} \to \mathcal{O}\}$, a class of functions where $\mathcal{Z}$ and $\mathcal{O}$ are respectively referred to as input and output spaces. There is at least one function f in $\mathcal{F}$ that represents a better solution than the others. The "goodness" of the solution is judged with respect to a chosen *loss* function $L : \mathcal{Z} \times \mathcal{F} \to \mathbb{R}$, measuring the performance of a solution over the input space. In this formulation, a "good" solution is a function in $\mathcal{F}$, which we note f^* in the remainder of Section 2.1, minimizing the expected loss, which is also called the *expected risk* and defined as follows:

$$R(f) = E[L(z, f)] = \int_{\mathcal{Z}} L(z, f) p(z) dz. \tag{2.1}$$

However, in general, p is not known and thus the expected risk cannot be computed. We only have access to a *finite* sample of N examples $D_N = \{z_1, \ldots, z_N\}$, drawn from the distribution p, referred to as the *training set* (or *development* set). These examples are used to *learn* an approximation of the "good" solution. Instead of searching for the function f^*, the function which we note f_N minimizing the *empirical risk*,

$$\hat{R}(f, D_N) = \frac{1}{N} \sum_{i=1}^{N} L(z_i, f), \tag{2.2}$$

is selected.

The reliability of this approximation is discussed in Section 2.1.2. In the following, a more concrete point of view on the use of machine learning is given. In practice, most problems treated in the machine learning framework fall into one of the following classes:

- **Regression:** In this class of problems the input space $\mathcal{Z}$ is filled with couples $z = (x, y) \in \mathbb{R}^M \times \mathbb{R}$ generated by the random variables $Z = (X, Y)$. The objective of this class of problems is to predict the value of Y for any value of X, *ie* to obtain an estimate of $E[Y|X = x]$. The hypothesis space $\mathcal{F}$ is chosen so that functions $f \in \mathcal{F}$ are potential predictors of Y values, *ie* $f(x) = \hat{y}$. An example of hypothesis space for a regression problem is the set $\mathcal{F}$ of linear functions f, such that:

$$\begin{aligned} f : \quad \mathbb{R}^M &\to \mathbb{R} \\ x &\mapsto f(x) = a \cdot x + b \end{aligned}$$

  where $a \in \mathbb{R}^M$ and $b \in \mathbb{R}$ are the parameters of the function. A typical training criterion for a regression problem is the *mean squared error* (MSE), which is the average over the training examples of the following loss function:

$$L(z, f) = L((x, y), f) = (f(x) - y)^2. \tag{2.3}$$

- **Classification:** This category of problems is a particular case of regression ones, where the random variable Y takes its values in a predefined discrete finite set, the set $\mathcal{Y}$ of possible classes. The objective of classification problems is to attribute to each value of X a class $y \in \mathcal{Y}$, *ie* to estimate the distribution $P(Y|X = x)$ for all possible values of X. In the particular case of two-class classification problems, the input space $\mathcal{Z}$ can be defined as a subset of $\mathbb{R}^M \times \{-1, 1\}$. An example of hypothesis space $\mathcal{F}$ is the partition f obtained by dividing the input space with linear functions:

$$\begin{array}{rll} f: & \mathbb{R}^M & \rightarrow \mathbb{R} \\ & x & \mapsto f(x) = \text{sign}(a \cdot x + b) \end{array}$$

  with $a \in \mathbb{R}^M$, $b \in \mathbb{R}$ and $\text{sign}(v) = 1$ if $v > 0$, $\text{sign}(v) = -1$ otherwise. A typical loss function for a classification problem is the following:

$$L(z, f) = L((x, y), f) = \frac{1}{2}|f(x) - y| = \begin{cases} 0 & \text{if } f(x) = y \\ 1 & \text{otherwise.} \end{cases} \tag{2.4}$$

- **Density estimation:** The objective of this class of problems is to directly estimate with $\hat{p}$ the unknown distribution p generating the examples $z \in \mathcal{Z} = \mathbb{R}^M$. In the case, for instance, where the examples are assumed to be drawn from a Gaussian distribution, the hypothesis space $\mathcal{F}$ will be the set of Gaussian density functions, with mean $\mu \in \mathbb{R}^M$ and variance $\Sigma \in \mathcal{S}(M)$, $\mathcal{S}(M)$ being the set of $M \times M$, real valued, symmetric and semi-definite matrices. The loss function often used for density estimation problems is the *negative log likelihood*:

$$L(z, f) = -\log(f(z)) = -\log(\hat{p}(z)). \tag{2.5}$$

Classification and regression problems which have an input space $\mathcal{Z} = \mathcal{X} \times \mathcal{Y}$ are called *supervised* learning problems. The component $x \in \mathcal{X}$ in $z = (x, y)$ is available for all examples while the component $y \in \mathcal{Y}$, called the supervision, is only available in the training set. Accordingly, density estimation problems are also referred to as *unsupervised* learning problems. A fourth class of problems called *semi-supervised* learning problems is becoming popular since several years, see for example Chapelle et al. (2006). In this kind of problems, the training set is composed of examples which have a supervision and others which do not. This category of problems is very interesting because in general collecting supervised training examples can be expensive, since it requires expert knowledge, while unsupervised examples are usually available in huge quantities.

We now have an insight on the kind of problems approached with machine learning tools. However, convergence issues must still be discussed. Indeed, even if machine learning is concerned with learning with a finite number of examples, asymptotic issues are interesting because they address the ability of the model to generalize to any new example, which is the main goal of machine learning.

2.1.2 Convergence Bounds

In the process of learning it is in general considered less interesting to "learn by heart" the examples at hand, than to have the ability to generalize from these examples to new ones. In the toy regression problem of Figure 2.1, the plain blue line represents a "learned by heart" solution, *ie* a solution which fits perfectly the training examples (red squares). The dashed green line is a much simpler solution making a lot of errors over the training set. However, in presence of new examples, the errors (*eg* according to the MSE measure of (2.3)) of the simpler solution are smaller than that of the complex one. The simpler solution is more "general".

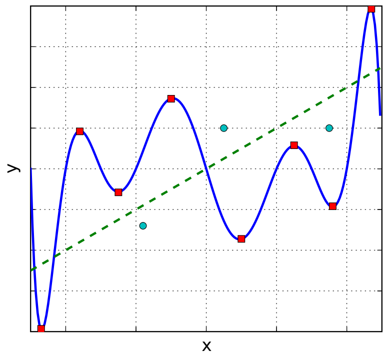


Figure 2.1: A toy regression example of the generalization concept. The red squares represent the training set examples and the cyan circles unseen examples drawn from p. The plain blue line represents a "learned by heart" solution, while the green dashed line is a simpler solution better generalizing to unseen data.

The expected risk of the solution f_N obtained by the minimization of the empirical risk, $R(f_N)$, is also called the *generalization* error. We are interested in the relations between the generalization error $R(f_N)$, the minimum expected risk $R(f^*)$ and the minimum empirical risk $\hat{R}(f_N, D_N)$, also called *training error*. The choice of the empirical risk (2.2) to approximate the expected risk (2.1) is quite natural. Indeed, for any function f independent from D_N, $\hat{R}(f, D_N)$ is an *unbiased* estimate of $R(f)$ (*ie* $E[\hat{R}(f, D_N)] = R(f)$) and according to the weak law of large numbers it is also *consistent*:

$$\forall \epsilon > 0,\ P[|\hat{R}(f, D_N) - R(f)| > \epsilon] \longrightarrow_{N \to \infty} 0, \tag{2.6}$$

ie $\hat{R}(f, D_N)$ converges in probability towards $R(f)$.

However, this result does not guarantee that the minimum empirical risk, $\inf_{f \in \mathcal{F}} \hat{R}(f, D_N) = \hat{R}(f_N, D_N)$, converges towards the minimum expected risk, $\inf_{f \in \mathcal{F}} R(f) = R(f^*)$, when N goes to infinity. It can be shown (Vapnik and Chervonenkis, 1971) that proving the *uniform* convergence in probability of $\hat{R}(f, D_N)$ towards $R(f)$, that is:

$$\forall \epsilon > 0,\ P[\sup_{f \in \mathcal{F}} |\hat{R}(f, D_N) - R(f)| > \epsilon] \longrightarrow_{N \to \infty} 0, \tag{2.7}$$

instead of the simple convergence in probability, of (2.6) is *sufficient* to ensure the convergence in probability of the minima.

The uniform convergence in probability has been proved (Vapnik and Chervonenkis, 1971) in certain conditions which depend on the complexity of the hypothesis space $\mathcal{F}$. A measure of this complexity, called the *capacity* or *VC-dimension* is defined in Vapnik and Chervonenkis (1971). In the particular case of a two-class classification problem, the capacity h of $\mathcal{F}$, is equal to the largest number h of examples $x_1, \ldots, x_h$ for which there exists, for every possible labelling, an $f \in \mathcal{F}$ which could be used as a perfect classifier. As an example, the capacity of the set of linear classifiers in $\mathbb{R}^M$ is $M+1$. Figure 2.2 illustrates for linear classifiers in $\mathbb{R}^2$ that the capacity of this set of function is at least 3.

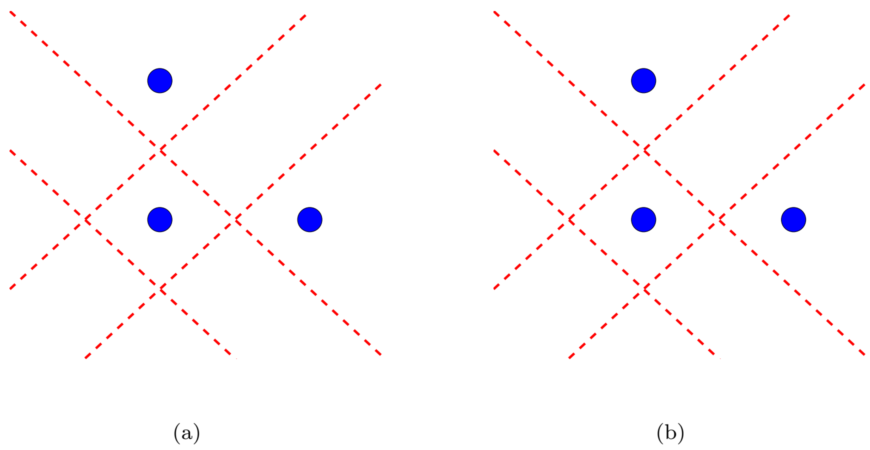


Figure 2.2: Capacity for linear classifiers in $\mathbb{R}^2$. Each red dashed line represents the perfect classifier of one of the possible labelling. 2.2(b) shows a labelling of 4 points that cannot be separated by a linear classifier.

Having defined the VC-dimension, the following theorem gives a bound for the uniform convergence in probability of $\hat{R}(f, D_N)$ towards $R(f)$ when $N \to \infty$.

Theorem 2.1.1 (Vapnik-Chervonenkis) *Consider a set of functions $\mathcal{F}$ with a finite VC-dimension h, and a loss function L such that there exists a constant $\tau = \sup L - \inf L$. $\forall \epsilon > 0$:*

$$P[\sup_{f \in \mathcal{F}} |\hat{R}(f, D_N) - R(f)| > \epsilon] \leq 4 \exp \left[h \left(1 + \log(\frac{2N}{h}) \right) - \left(\epsilon - \frac{1}{N} \right)^2 \frac{N}{\tau^2} \right]. \tag{2.8}$$

In addition to ensure the uniform convergence when $N \to \infty$, this theorem can be derived (Vapnik, 1995) to provide an interesting bound for the difference of empirical and expected risks for the optimal function f_N, for any fixed N. For all $\eta > 0$, with at least probability $1 - \eta$, provided that there exists a constant $\tau = \sup L - \inf L$,

$$R(f_N) \leq \hat{R}(f_N, D_N) + \tau \sqrt{\frac{h \left(\log(\frac{N}{h}) + 1 \right) - \log\left(\frac{\eta}{4}\right)}{N}}. \tag{2.9}$$

As illustrated in Figure 2.3, the inequality (2.9) expresses the necessity of a trade-off between the minimization of the empirical risk and the capacity, in order to obtain good generalization performance. Indeed, for a fixed training set D_N, when the complexity of the hypothesis space increases, the value of the optimal empirical risk (*ie* the training error) decreases, while the bound on the difference of risks in (2.9) increases.

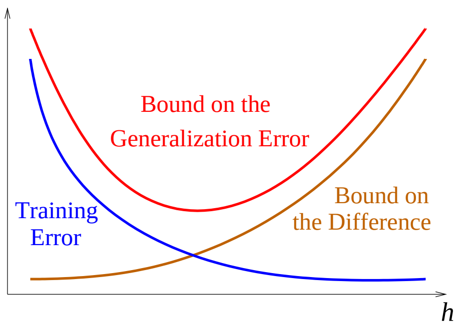


Figure 2.3: Illustration of the Vapnik-Chervonenkis' theorem result for a fixed N.

Finally it follows the intuition we had from Figure 2.1, that a polynomial solution, with its high capacity, *over-fits* the training examples (poor generalization and low training error), while a linear solution having low capacity *under-fits* the training data (maybe too general and high training error). Section 2.1.3 discusses how an "appropriate" capacity can be chosen.

One last point that is interesting to mention is the fact that the capacity of a set of functions often depend on the dimension of its input space. In the previously cited example of the set of linear classifiers in $\mathbb{R}^M$ the capacity is equal to $M+1$. As a consequence for high dimensions the hypothesis space capacity is also high and thus the learning process is prone to over-fitting. This is a manifestation of a more general phenomenon referred to as the *curse of dimensionality* (Bellman, 1961). A common approach to try to overcome this problem is to assume that the data lies in fact in a manifold of lower dimension embedded in the high dimensional space, and to try to discover a mapping of the data on to this manifold.

2.1.3 Model Selection

The capacity of a given hypothesis space is in general difficult to compute, thus in most cases the bounds considered in the previous section are not used. Instead the generalization error is approximated directly using validation tools. The idea behind the validation tools is that by considering a set of $\tilde{N}$ examples, the *validation* set $D_{\tilde{N}} = \{\tilde{z}_1, \ldots, \tilde{z}_{\tilde{N}}\}$, obtained independently but from the same distribution than that of the training set, we can get an estimate of the generalization error. Let us introduce the *validation error*,

$$\hat{R}(f_N, D_{\tilde{N}}) = \frac{1}{\tilde{N}} \sum_{i=1}^{\tilde{N}} L(\tilde{z}_i, f_N), \tag{2.10}$$

which is the empirical risk computed over the validation set of the function f_N minimizing the training error. Unlike the training error, the weak law of large numbers can be applied to the validation error and ensures its convergence in probability towards the generalization error $R(f_N)$ when $\tilde{N} \to \infty$.

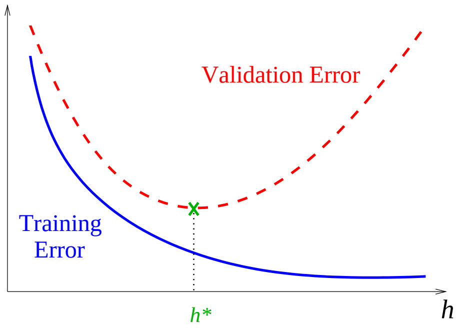


Figure 2.4: Validation and Training Errors variation with respect to the capacity h.

The validation error is thus used as an estimator of the generalization error. As illustrated in Figure 2.4, it follows the same pattern as the generalization error of Figure 2.3 when the capacity h of the hypothesis space is increased. This behavior is used in nested hypothesis spaces of increasing capacity h, such as for example the space of polynomial functions of degree k, $1 \leq k \leq m$, in order to choose an optimal capacity h^*.

To compute the validation error a validation dataset is required. However, in general few training material is available. Several strategies have been proposed to overcome this limitation. The first approach is to sacrifice a subset $D'_{N'} \subset D_N$ of the training set to build a validation set. The training error is computed over $D'_{N-N'} = D_N \backslash D'_{N'}$ and the validation error over $D'_{N'}$. In order to obtain a more accurate estimate of the generalization error one can use instead what is called *cross-validation*. The training set is partitioned into two sets $D^1_{\tilde{N}}$ and $D^2_{\tilde{N}}$, $\tilde{N} = \frac{N}{2}$. Each set acts alternatively as training and validation sets. The final training and validation errors, $\hat{R}_{train}$ and $\hat{R}_{valid}$, are the means of the errors over the subsets:

$$\hat{R}_{train} = \frac{1}{2}\left[\hat{R}(f_{D^1_{\tilde{N}}}, D^1_{\tilde{N}}) + \hat{R}(f_{D^2_{\tilde{N}}}, D^2_{\tilde{N}})\right], \quad \hat{R}_{valid} = \frac{1}{2}\left[\hat{R}(f_{D^1_{\tilde{N}}}, D^2_{\tilde{N}}) + \hat{R}(f_{D^2_{\tilde{N}}}, D^1_{\tilde{N}})\right].$$

As it might be noticed, all the examples in D_N are actually used to computed the two errors. These errors help to choose the optimal capacity of the hypothesis set $\mathcal{H}$ before the whole training set D_N is reused to learn the optimal $f_N \in \mathcal{H}$. A model obtained for N examples might however be quite far from a model learned over half the examples. In order to reduce this variance, a K-*fold* cross-validation is performed instead of a simple cross-validation. The training set is partitioned into K sets, which we note $D^k_{\tilde{N}}$, $k \in \{1, \ldots, K\}$, with $\tilde{N} = \frac{N}{K}$. Each of these subsets $D^k_{\tilde{N}}$ acts alternatively as a validation set for the model learned over the union $\bigcup_{q \in \{1,\ldots,k-1,k+1,\ldots,K\}} D^q_{\tilde{N}}$ of the remaining subsets. The estimate of the generalization error with the smallest variance is obtained by performing a *leave-one-out* cross-validation, that is a K-fold cross-validation where the number of folds K equals the number N of examples in the initial training set D_N.

2.2 Information Retrieval

Information Retrieval is a field that has its origins in library organization needs. According to Sparck-Jones and Willett (1997), soon after the second world war and the advent of computers, the first systems for automating the retrieval of off-line documents were implemented. With the increase of digitalized textual data, the Information Retrieval field now encompasses applications from document retrieval to "knowledge" retrieval, such as text categorization, information extraction, question answering, summarization, etc.

As illustrated in Figure 2.5, an information retrieval system can be decomposed into several modules. The ideas developed in this thesis have been mostly applied to document retrieval and text categorization tasks which are described in Sections 2.2.2 and 2.2.3. Note that the description on how the performance of information retrieval systems is evaluated is delayed until Chapter 3. In order to automate any of these information retrieval tasks the documents must have a formal representation. The most common and intuitive approach is to *index* documents with words. This topic, encompassing the first three steps of Figure 2.5, is discussed in the following Section 2.2.1.

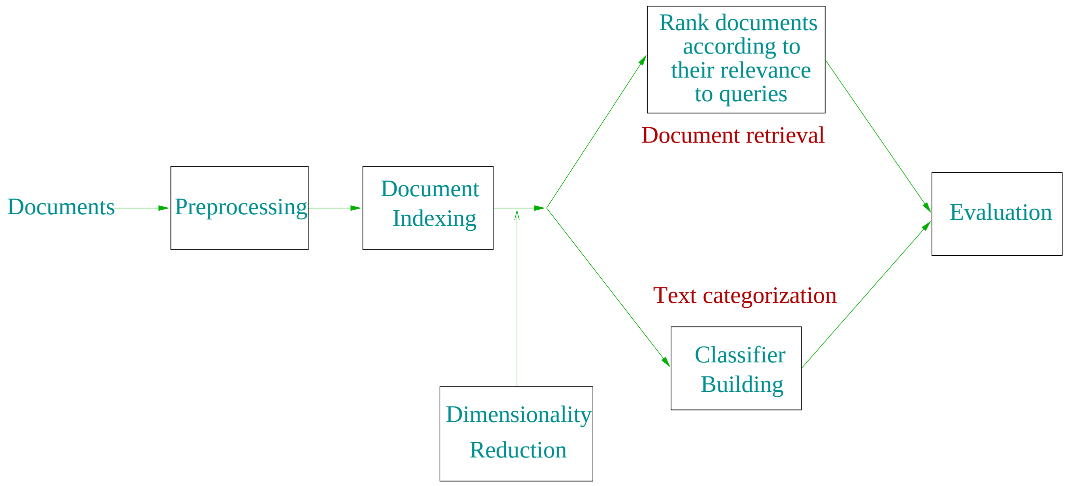


Figure 2.5: Schema of an information retrieval system processing steps.

Note that information retrieval can also be seen as being a particular problem in the broader field of Natural Language Processing (see for example Manning and Schütze, 1999) which is more driven towards linguistic than the approach we will considered.

2.2.1 Indexing

Usually in a library digital database, a list of keywords is attributed manually to each document, in addition to specific information (title, author, etc.), in order to facilitate the search of the physical documents (*eg* books, papers, patents, etc.). This list enables the user to query the documents related to one or several keywords. More generally, these keywords are also called indexes and representing each document in a database to be processed by the computer with a list of indexes is called *indexing*.

Indexing relies on a predefined set of keywords: the *dictionary* or *vocabulary*. In a digitalized text database (also called a *corpus*) the dictionary is usually extracted from the set of words present in the corpus and the expensive manual indexing can be avoided by representing the documents by the dictionary terms they are made of.

The most common approach to automatically index the documents in a corpus is called the *vector space model* (VSM) (Salton et al., 1975), which is described in the following.

The Vector Space Model

In most Information Retrieval applications, documents are represented within the *Vector Space Model* (VSM). In this model, each document d is represented as a vector $(\alpha_1, ..., \alpha_M)$, where α_j is a function of the frequency in d of the j^{th} word w_j of a chosen dictionary of size M.

Note that the VSM does not take into account the order of the words in the document. For that reason it is often referred to as *bag-of-words* representation.

The simplest choice for the weights $\alpha_.$ corresponds to the binary indexing, where:

$$\alpha_j = \begin{cases} 1 & \text{if the word } w_j \text{ appears in } d, \\ 0 & \text{otherwise} \end{cases} \qquad \forall j \in \{1, \ldots, M\}.$$

The more complex *term frequency inverse document frequency* (TFIDF) weighting scheme was proposed by Salton and Buckley (1988) in the context of document retrieval where it is considered that words appearing too frequently across the corpus may not be discriminant. In its standard form it is defined by the following formula:

$$\alpha_j = \mathrm{tf}_j(d) \cdot \log(\frac{N}{\mathrm{df}_j}), \quad \forall j \in \{1, \ldots, M\},$$

where $\mathrm{tf}_j(d)$ corresponds to the number of occurrences of w_j in d, N is the number of documents in the corpus and df_j stands for the number of documents the term w_j appears in. As it is, it give more importance to terms frequent in the document while penalizing words appearing in too many documents. Usually the weights are in addition normalized so that $\|d\| = \sqrt{\sum_{j=1}^{M} \alpha_j^2} = 1$ in order to give the same importance to longer or shorter documents.

Even if the weighting scheme in VSM will considerably affect the importance of a word in a document, the choice of the dictionary is still essential to this model. This choice is done in part automatically during the preprocessing step and the dimensionality reduction step of Figure 2.5.

The Dictionary's Selection

When computing and ranking the frequencies of the words in the corpus, one notices that the most frequent words do not convey a lot of information. Indeed, as can be seen in the example of Table 2.1, where the ten most frequent words in the Reuters 21578 corpus (see database description in Sect. 3.1) are displayed, there are mostly *function* words, such as "the", "of" and "and". The only words that may give us some information about the documents in the database are "said" and "mln" (abbreviation of "million"), which rightly convey the fact that a lot of the news-tories in Reuters 21578 corpus report the discourse someone had and are about economics.

Rank	Word	Frequency	% of the corpus
1	the	144637	5.8
2	of	72943	2.9
3	to	69777	2.8
4	and	54228	2.2
5	in	53198	2.1
6	said	53095	2.1
7	a	51989	2.1
8	for	26429	1.0
9	mln	26060	1.0
10	it	22425	0.9
total			22.9

Table 2.1: The ten most frequent words in the Reuters 21578 database dictionary before preprocessing.

Even if function words have a syntactic importance, due to the fact that the bag-of-words representation neglects this kind of information they appear as non-discriminant. In order to lighten the text representation, a stop-list is often designed manually and the words appearing in it, called *stop-words*, are ignored in the corpus. As can be seen on the first line of Table 2.2, for a stop-list of around 200 words, the number of words' occurrences in the corpus is reduced by 41%.

In this bag-of-words representation of texts, different words sharing the same morphological roots are considered not to add much discriminative power to the representation. They are thus replaced in the corpus by their *stem*. For example the words "connectivity", "connecting", "connected", "connections", etc, are replaced with the stem "connect" by the most popular automatic stemmer, Porter's stemmer (Porter, 1980). As can be seen in Table 2.2, there is a reduction of 29.5% of the size of the dictionary after stemming.

	initial	after stopping	% reduction	after stemming	% reduction
# of corpus' words	2 504 356	1 481 074	41%	1 481 074	0%
# of dictionary's word	41 846	41 627	0.5%	29 485	29.5%

Table 2.2: Illustration of the stopping and stemming respective influence on the numbers of words in the Reuters 21578 corpus and in its dictionary.

There have been several attempts (*eg* Schutze et al., 1995; Srinivasan, 1992) at using *phrases*, *ie* compounds of words such as "machine learning" or "bag-of-words", in addition to words as indexing terms. However, these phrases are expensive to collect and according to Sebastiani (2002) and Baeza-Yates and Ribeiro-Neto (1999) the improvements are not very important.

As it can be noticed in Table 2.2, even after the stemming step the number of terms in the dictionary is quite high. We mentioned in Section 2.1.2, that methods applied to data in high dimensions are exposed to the curse of dimensionality. It is particularly the case for text categorization methods which may rely on more complex models. For that reason the text categorization community has developed several approaches for *feature selection*, others than the design of a stop-list and the stemming of words. These approaches attempt to select from the original vocabulary $\mathcal{V}$, the sub-set $\mathcal{V}'$ of terms (with $\mathcal{V}' \ll \mathcal{V}$) that, when used for document representation, provides the best results. As stated in Sebastiani (2002), the choice of this subset is generally done by a filtering approach, that is selecting the terms that reach the highest scores according to a function that measures the "importance" of each term for the task. This function can simply be the *document frequency* (Yang and Pedersen, 1997; Dumais et al., 1998), or a more complex information-theoretic function such as *information gain* (Yang and Pedersen, 1997; Yang and Liu, 1999) , *mutual information* (Yang and Pedersen, 1997; Dumais et al., 1998), *chi-square* (Schutze et al., 1995; Yang and Pedersen, 1997; Yang and Liu, 1999), etc. Nevertheless, Joachims (1998) shows that the performance of a system trained without the "best" features according to mutual information criterion are still better than a random system. This result proves that even the "worst" features still contain discriminant information, and thus suggest that all features should be selected, when possible.

As can be seen in Figure 2.5 the optional dimensionality reduction step is the last modification to the document representation before the information retrieval task itself. In Section 2.3 a broader view of dimensionality reduction is given, but to have a more concrete base a description of document retrieval and text categorization is first given.

2.2.2 Document Retrieval

In a document retrieval task, a user formulates a query q addressed to a database, and the database documents d are then ranked according to their *Relevance Status Value*, $RSV(q,d)$, which is defined such that documents relevant to q should have higher values than non-relevant ones. The same dictionary is used to index both the documents and the queries. In the VSM (Sect. 2.2.1), $RSV(q,d)$ is defined as the scalar product of the query's and document's representations:

$$RSV(q,d) = \sum_{j=1}^{M} \alpha_j^q \cdot \alpha_j^d, \tag{2.11}$$

where α_j^d (resp. α_j^q) is the weight in the document (resp. query) representation of the j^{th} dictionary word.

Since in general the user's query is formulated with less words than the documents they attempt at retrieving, the observed distribution of terms is noisy. Indeed, a given word absent from the formulation of the query may however be representative of the user need, because it is semantically related to the words present in the query. In the VSM it will unfortunately be assigned a weight of 0. As a first approximation this fact is ignored and α_j^q is chosen to be a binary weight and α_j^d the TFIDF weight. All the words chosen by the user to formulate her query have the same weight. A way to attempt at overcoming this problem is by query expansion methods. These methods consider that the query formulated by the user is a tentative and they alter it by adding other terms. This is generally done by performing a preliminary retrieval process and by selecting the more frequent words in the relevant documents (relevant according to feedbacks given by the user (Salton and Buckley, 1990) or by taking the n first documents in the ranking (*eg* Gauvain et al., 2000)).

In a usual document retrieval benchmark database, there is a corpus of targeted documents and, in order to evaluate the quality of the ranking, a set of training queries and a set of test queries with relevance judgement associated to the corpus' documents. The relevance judgements are in general a binary decision, relevant/irrelevant, provided by human experts for each document in the corpus and each query.

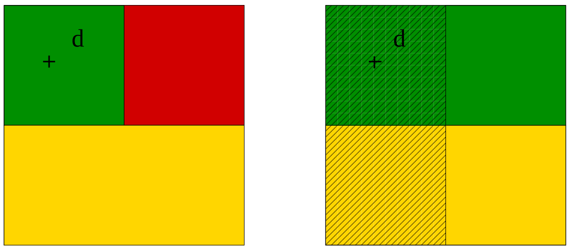
In order to refine the weighting of the terms in documents some hyper-parameters can be tuned by validation over the set of training queries. In the OKAPI's *RSV* formula (Robertson et al., 1993):

$$RSV(q,d) = \sum_{j \in Q} \frac{(K+1)\,\mathrm{tfidf}_j(d)}{K[(1-b)+b\mu(d)] + \mathrm{tf}_j(d)}, \tag{2.12}$$

where $\mu(d)$ is the normalized document length (length of d divided by the average document length) and Q is the set of word indexes present in q; the hyper-parameters $b \in [0,1]$ and $K \in [0, +\infty[$ are tuned to control the influence of the document length and the tf_j the terms' weight.

2.2.3 Text Categorization

The goal of text categorization is to automatically assign, for each documents, categories selected among a predefined set. In opposition to the machine learning typical multi-class classification task which aims at attributing one class among the possible ones to each example, documents in a text categorization task may belong to several categories (see Figure 2.6).



(a) 3 classes (b) 3 categories

Figure 2.6: Classification vs Categorization. In (2.6(a)) d belongs to green dark class only, while in (2.6(b)) d belongs to green dark category and to the striped one.

According to Sebastiani (2002), text categorization techniques were first applied to obtain an automatic document indexing for boolean information retrieval system relying on a *controlled dictionary* (Maron, 1961). In general, a thematic hierarchical thesaurus was the controlled dictionary and the aim was to attribute one or several keywords from the thesaurus to documents for later retrieval. The problem of indexing with a controlled vocabulary is an instance of the broader problem of corpus organization. Text categorization methods are also applied to the text filtering problem (Belkin and Croft, 1992) where there is an incoming stream of documents that need to be dispatched according to their categories. A typical example is that of a news agency producing a stream of thematic news stories that must then be dispatched according to the interest of a newspaper multiple sections. In this case the documents to categorize are not available when the system is built. The first approaches to text categorization were rule-based classifiers, with rules manually designed by experts; however since the early '90s machine learning approaches form the state-of-the-art for that task.

In a typical benchmark database, we have access to a training set of labeled documents, in which each document $x \in \mathbb{R}^M$ represented in the VSM (M being the dictionary's size) has an associated vector $y = (y_1, \cdots, y_K)$, $y_j \in \{-1, 1\}$ indicating the membership of x to category j, for each j among the K predefined categories. There is also a test set with labeled documents on which, after the training of the system is completed, it can be tested for comparison with others systems. In general the text categorization problem is broken into K two-class classification problems, each of which determines the belonging to a given category. Each classifier is composed of a function $f_j : \mathbb{R}^M \to \mathbb{R}$ and a threshold θ_j such that:

$$f_j(x) > \theta_j \textbf{ iff } x \text{ belongs to category } j.$$

Several classifiers have been proposed to solve the text categorization problem. See Sebastiani (2002) for a complete review. However on several benchmark databases the Support Vector Machines (SVMs) approaches proposed by Joachims (1998) and Dumais et al. (1998) outperform the others. We will thus concentrate on this approach.

Since SVMs are a standard method in machine learning we describe it only briefly. SVMs (Cortes and Vapnik, 1995) are linear classifiers which training is driven by the maximization of their margin which can be seen as a kind of security corridor around the decision function. The SVM model is built upon the support vectors which are either the training examples in contact with the corridor or misclassified training examples. Another interesting point about SVMs is that they are able to perform the classification in a space of higher dimension without explicitly computing a projection function, through the use of the so-called kernel trick.

Joachims (1998) lists some facts which may explain the success of SVMs in text categorization. The first one is that because the capacity of their hypothesis space can be independent of the input space dimension, SVMs are impervious to the curse of dimensionality, described in Section 2.1.2.

2.3 Text Representation

As it has been explained in Section 2.2.1, automatic indexing provides a representation of documents. Despite the fact that the obtained bag-of-words representation is widely used, it has some drawbacks that it may be desirable to get rid of.

A first problem is that the dimension of the obtained representation space is equal to the size of the dictionary (order of magnitude 20 000 words). As a general consequence a lot of parameters are required to estimate any system taking bag-of-words documents as inputs. This leads easily to the curse of dimensionality, particularly when we are restricted to a limited amount of labeled data.

Another drawback of the VSM is that it does not take into account the possible semantic links that exist between words. It will for example make a high distinction between the sentences *the ocean is azure* and *blue is the color of the sea*, while we, human, know that these sentences are related.

In the following, several approaches, which we will refer to as text representation methods, are presented. They depart from the bag-of-words representation and, project the data into a new space where the information retrieval task is to be performed. During this extra step, a few features representing the documents are learned. They attempt at incorporating knowledge about the semantic links between words, based on their occurrences on the huge amount of unlabeled documents in digital format to which we have access nowadays.

2.3.1 Latent Semantic Analysis

Latent Semantic Analysis (LSA), the application of Singular Value Decomposition (SVD) to the bag-of-words representation, was originally proposed by Deerwester et al. (1990). The idea of LSA is to perform SVD of the term-by-documents matrix:

$$X_{M\times N} = \begin{pmatrix} \alpha_{11} & \dots & \alpha_{1N} \\ \vdots & \vdots & \vdots \\ \alpha_{M1} & \dots & \alpha_{MN} \end{pmatrix}$$

where, α_{ij} is the value of the i^{th} word in the j^{th} document, N is the size of the training set and M the size of the dictionary, in order to find a new system of axes fitting the data, with the hope that the data is in fact well represented with **few** of these new dimensions.

SVD can be seen as a decomposition of the term-by-document matrix, in this way:

$$X = U\Sigma V', \qquad \Sigma = \begin{pmatrix} \sqrt{\lambda_1} & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & \sqrt{\lambda_P} \end{pmatrix} \tag{2.13}$$

where $P = \min(M, N)$, $\lambda_1 \leq \ldots \leq \lambda_P$, U and V are both orthogonal matrices (*ie* $U'U = V'V = I$). $U = (\boldsymbol{u_1}, \ldots, \boldsymbol{u_P})$ (resp. $V = (\boldsymbol{v_1}, \ldots, \boldsymbol{v_P})$) is the matrix which columns $\boldsymbol{u_j}$ (resp. $\boldsymbol{v_j}$) correspond to the eigenvectors associated to each of the P eigenvalues λ_j of XX' (resp. $X'X$)[1].

Equation (2.13) can also be seen as the reconstruction of the initial data from the data projected in the new axis $\{\boldsymbol{v_1}, \ldots, \boldsymbol{v_P}\}$. If instead of taking the P axis, we choose the first Q, with $Q \leq P$, we have an approximation of the data:

$$\hat{X}_{M\times N} = \hat{U}\hat{\Sigma}\hat{V}', \qquad \hat{\Sigma} = \begin{pmatrix} \sqrt{\lambda_1} & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & \sqrt{\lambda_Q} \end{pmatrix}. \tag{2.14}$$

[1] Note that it can easily be shown that XX' and $X'X$ share theirs eigenvalues. Indeed by definition: $\boldsymbol{u_j}$ is the eigenvector corresponding to the eigenvalue λ_j of $XX' \Leftrightarrow XX'\boldsymbol{u_j} = \lambda_j \boldsymbol{u_j}$, and the same is true for $\boldsymbol{v_j}$ and μ_j of $X'X$ ($X'X\boldsymbol{v_j} = \mu_j \boldsymbol{v_j}$). Thus, $X(X'X\boldsymbol{v_j}) = XX'(X\boldsymbol{v_j}) = \mu_j X\boldsymbol{u_j}$. Then $X\boldsymbol{v_j}$ is an eigenvector of XX' with eigenvalue μ_j. Reciprocally, $X'\boldsymbol{u_j}$ is an eigenvector of $X'X$ with eigenvalue λ_j. And because $\{\lambda_1, \ldots, \lambda_P\}$ and $\{\mu_1, \ldots, \mu_P\}$ are ordered we find that $\forall j \lambda_j = \mu_j$.

The reconstruction quality can be evaluated by the *variance percentage*:

$$\tau_Q = \frac{\sum_i \sum_j \hat{\alpha}_{ij}^2}{\sum_i \sum_j \alpha_{ij}^2} = \frac{\sum_{q \le Q} \lambda_q}{\sum_{q \le P} \lambda_q}.$$

By considering the scalar product of documents in this approximate representation

$$\hat{X}'\hat{X} = \hat{V}\hat{\Sigma}'\hat{U}'\hat{U}\hat{\Sigma}\hat{V}' = \hat{V}\hat{\Sigma}'\hat{\Sigma}\hat{V}',$$

one can see the columns of $\hat{\Sigma}\hat{V}'_{Q\times N}$ as being the new representation of documents in a "latent" space of Q concepts.

For any new document d^*, the Latent Semantic Indexing (LSI) considers, instead of re-computing the SVD of a new matrix X containing d^*, to compute its new representation as:

$$\hat{d^*} = \hat{U}'d^*,$$

the projection of d^* over the Q first eigenvectors $\boldsymbol{u_1}, \dots, \boldsymbol{u_Q}$.

According to Deerwester et al. (1990) and Hofmann (2001), LSI performs better in IR tasks than basic term matching with Vector Space Model in some databases, and worst in some others.

However, for large databases the SVD becomes intractable, since the computation of eigenvalues and eigenvectors requires the inversion of the matrix $X'X$ or the matrix XX', inversion which has a complexity of $\mathcal{O}(P^3)$. Therefore, algorithmic approximations are used instead.

Symmetrically to the LSI, one can also consider that the j^{th} word in the dictionary is represented in LSA by the projection of $w_j = (\alpha_{j1}, \dots, \alpha_{jN})$, on the Q eigenvectors $\boldsymbol{v_1}, \dots, \boldsymbol{v_Q}$:

$$\hat{w}_j = w_j\hat{V}.$$

However, note that the dimension of $\hat{V}_{N\times Q}$ increases with the number of documents in the database. This may point out that LSI has not been designed to extract information about relation between words in big databases. Indeed, in small databases, the size of dictionary exceeds the number of documents. However, as in can be seen in Figure 2.7 eventually the contrary is observed. This meets the intuition that with a finite number of words we can produce an infinite number of documents. However, the symmetry of LSA between words and documents does not seem to tackle this fact.

2.3.2 Probabilistic Models

Several probabilistic models have been proposed to model relationship between words and documents. Among them the Probabilistic Latent Semantic Analysis (PLSA) by Hofmann (2001), the Latent Dirichlet Allocation (LDA) by Blei et al. (2003) also referred to as multinomial Principal Component Analysis (mPCA), proposed by Buntine (2002), and the Theme Topic Mixture Model (TTMM) (Keller and Bengio, 2004b) which will be defined in Chapter 4, have in common their main idea. This idea is to assume that the choice of words in the generation of a document is independent of the document given a hidden

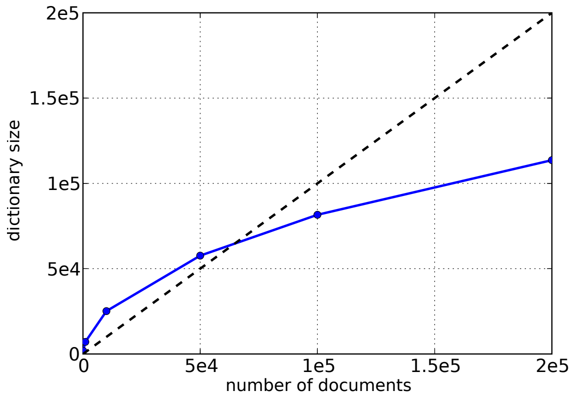


Figure 2.7: Size of the dictionary extracted from several proportion of the RCV1 dataset (see Section 3.1 for a description of this dataset)

variable T which is called **Topic** or sometimes Aspect. T follows a multinomial distribution with K possible values. The three models have a random variable X in the document space and for each word l a random variable w_l taking values in $\{0, 1\}$. The assumption these models share can be written as:

$$P(w_l|X) = \sum_{k=1}^{K} P(T = k|X)P(w_l|T = k). \tag{2.15}$$

Therefore, the probability of each word given the variable X responsible for the observed document is seen as a mixture over the possible topics, with $P(T = k|X)$, $k \in \{1, \ldots, K\}$, as mixing proportions. The main difference between these models lies in the way variable X is defined and from which distribution its values are drawn.

Another popular description of the generic structure of these models is given by Figure 2.8. Probabilistic models which can be described with this kind of graphical representation are called Graphical Models (Jordan, 1999). In the figure, the boxes represent replicates. The outer box represents the repeated trials of the variable X in the document space (N is the number of documents), while the inner box represents the repeated choice of topics within the word space (M is the vocabulary size) for a given value of the variable X. The arrows between the variable X, **Topic** and **Word** represent the dependencies. As shown in this figure the variable X in the document space and the **Word** variable in the word space have no direct dependencies. Note that, in these three models, multiple topic dependencies over words are not taken into account.

In the following a description of PLSA and LDA is given. The description of TTMM and the comparison of the three models is delayed until Chapter 4.

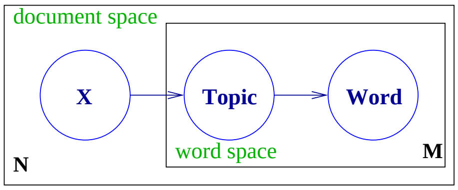


Figure 2.8: A graphical representation of the generic structure of a document density model.

Probabilistic Latent Semantic Analysis

The main distinctive feature of PLSA is that it seeks a generative model for **word/document co-occurrences**, rather than a model for documents themselves. From that, it follows that the variable in the document space, which we note δ (see Figure 2.9), is a variable that picks one document among the others in the database, since we need to model each word occurrence in each document. To say it differently, δ takes a value among the document indexes $\{1, \ldots, N\}$, the probability $P(\delta)$ being proportional to the length of the δ^{th} document.

Equation (2.15) can be re-written as:

$$P(w_l|d_\delta) = \sum_{k=1}^{K} P(T = k|d_\delta)P(w_l|T = k),$$

and the word/document co-occurrences model is expressed by:

$$P(d_\delta, w_l) = P(d_\delta)P(w_l|d_\delta) = P(\delta)\sum_{k=1}^{K} P(T = k|d_\delta)P(w_l|T = k). \tag{2.16}$$

This model can be seen, from another perspective, as being the description of the process generating the data. This process goes through the following steps:

1. Select a document index δ with probability $P(\delta)$
2. Pick a latent topic $T = k$ with probability $P(T = k|d_\delta)$
3. Generate a word w_l with probability $P(w_l|T = k)$

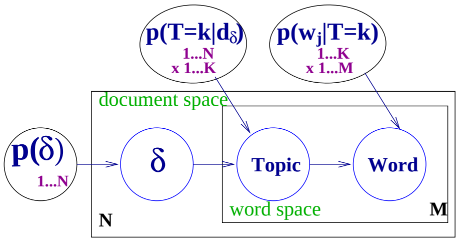


Figure 2.9: A graphical representation of PLSA

The log-likelihood of the model,

$$\mathcal{L} = \sum_{\delta=1}^{N} \sum_{l=1}^{M} \mathrm{tf}_l(d_\delta) \log P(d_\delta, w_l)$$

with $\mathrm{tf}_l(d_\delta)$ the frequency of the word w_l in d_δ, is maximized by Expectation-Maximization (EM) (Dempster et al., 1977), as follows:

The **E-step** consists of computing the posterior probabilities for the latent variable, based on the current estimates of the parameters, that is:

$$P(T = k|w_l, d_\delta) = \frac{P(w_l|T = k)P(T = k|d_\delta)}{\sum_{q=1}^{K} P(w_l|T = q)P(T = q|d_\delta)}.$$

The **M-step** consists of the maximization of the expected joint log-likelihood of the observed and latent variables given the estimations of the previous step. This is achieved in PLSA with the following parameters re-estimation:

$$P(w_l|T = k) = \frac{\sum_{\delta=1}^{N} \mathrm{tf}_l(d_\delta)P(T = k|w_l, d_\delta)}{\sum_{m=1}^{M} \sum_{\delta=1}^{N} \mathrm{tf}_m(d_\delta)P(T = k|w_m, d_\delta)}$$

$$P(T = k|d_\delta) = \frac{\sum_{l=1}^{M} \mathrm{tf}_l(d_\delta)P(T = k|w_l, d_\delta)}{n(d_\delta)},$$

with $n(d_\delta) = \sum_{l=1}^{M} \mathrm{tf}_l(d_\delta)$. PLSA can be used to replace the original document representation by a representation in a low-dimensional "latent" space, in order to perform a text categorization or a document retrieval task. In Hofmann (2001), the components of the document in the low-dimensional space are chosen to be $P(T = k|d), \forall k$, and for each unseen document or query they are computed by maximizing the log-likelihood with $P(w_l|T = k)$ fixed. This representation scheme is referred to as PLSI, for Probabilistic Latent Semantic Indexing. It has been pointed out by Blei et al. (2003) that PLSA is not a well-defined generative model of documents, since there is no direct way to assign probability to an unseen document. However, some experiments in Hofmann (2001) report a comparison between LSI and PLSI, on several corpora. They point out a better performance of PLSI in all cases. In particular PLSI performs well even in the cases where LSI fails completely.

The other weakness of PLSA can be described as follows. The parameters of a K-topics PLSA model are the K multinomials of size M and the K mixing proportions for each of the N documents. Hence, the number of parameters equals $KM+KN$ and therefore grows linearly with the number of documents. This suggests that the model is prone to over-fitting. In practice, to try to overcome this problem, a tempered EM (TEM) is performed instead of the EM. During the TEM iterations the parameters are smoothed in order to achieve an acceptable predictive performance on a validation set. However, according to Blei et al. (2003), over-fitting can occur even with the TEM version.

Latent Dirichlet Allocation

In LDA the documents are assumed to be sampled from a random mixture over latent topics, where each topic is characterized by a distribution over words. In this model the observed variable is the document d, seen as a set of words w_l, $l \in \{1, \ldots, M\}$, each of these words being dependent of the unobserved variable T, the **Topic**, with possible values in $\{1, \ldots, K\}$, and K being an hyper-parameter that must be chosen. In the document space the LDA model has an unobserved variable, $\tau = (\tau_1, \ldots, \tau_K)$, $\tau_k > 0$, $\sum_{k=1}^{K} \tau_k = 1$ (see Fig. 2.10), which follows a Dirichlet distribution with parameter $\zeta \in \mathbb{R}^{+K}$, responsible for the mixing proportions of the topics in each document.

Equation (2.15) can be re-written as:

$$P(w_l|\tau) = \sum_{k=1}^{K} P(T = k|\tau) P(w_l|T = k),$$

and the probability of an observed model d can be decomposed as follows:

$$\begin{aligned} P(d|\zeta) &= \int P(\tau|\zeta) P(d|\tau) d\tau \\ &= \int P(\tau|\zeta) \prod_{w_l \in d} P(w_l|\tau) d\tau \\ &= \int P(\tau|\zeta) \prod_{w_l \in d} \left[\sum_{k=1}^{K} P(T = k|\tau) P(w_l|T = k) \right]^{\mathrm{tf}_l(d)} d\tau \end{aligned} \tag{2.17}$$

The generative process for each document d is the following:

1. Choose $n(d) \sim Poisson(\xi)$: the document size
2. Choose $\tau \sim Dirichlet(\zeta)$: the random mixing proportions
3. For each of the $n(d)$ words of d:
   (a) Choose a topic $T = k$ from $P(T|\tau)$, a multinomial probability with parameter τ
   (b) Choose a word w_l from $P(w|T = k)$, a multinomial probability conditioned on the topic $T = k$

where $\zeta = (\zeta_1, \ldots, \zeta_K)$, $\zeta_k > 0$, is a parameter to estimate. The randomness of the document size $n(d)$, modeled for example by a Poisson distribution with parameter ξ, is necessary for the generative process. However, given that $n(d)$ is independent of all the other data generating variables (τ and T), it is not of real interest for the modelisation.[2] Hence, it will be ignored.

LDA, contrary to PLSA, is a true generative model of documents since both observed and unseen documents can be generated by the process described above. The parameters of a K-topics LDA model are $\zeta \in \mathbb{R}^{+K}$ the Dirichlet parameter and the M parameters of each of the K multinomial estimating

[2] In fact the log-likelihood will have this form: $\mathcal{L} = A(n(d)) + B(\tau, T)$ and thus maximizing it will lead to two distinct problems.

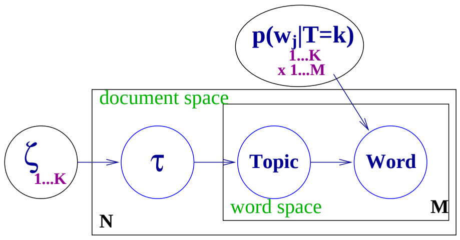


Figure 2.10: A graphical representation of LDA/mPCA

$P(w|T = k), \forall k$. That is $K + KM$ parameters for LDA, which is less than for PLSA and in addition independent of the number of documents.

However, in order to estimate these parameters one has to compute the posterior distribution $P(\tau, T|d)$, which is intractable in general, according to Blei et al. (2003). Therefore, instead of doing an exact inference for LDA, the authors of the paper propose an approximate inference algorithm based on a variational method (Jordan et al., 1999). Indeed, their algorithm maximizes a lower bound on the log-likelihood based on a variational distribution that approximates the posterior distribution $P(\tau, T|d)$. This maximization is done by a so-called variational EM algorithm, which consists in the iteration of the following two steps:

1. **(E-step)** Variational approximation of the posterior distribution. This is performed by an iterative algorithm, which requires approximatively $[n(d)]^2K$ operations for each document, according to Blei et al. (2003).

2. **(M-step)** Maximize the resulting lower bound on the log-likelihood with respect to the parameters of the model.

In order to represent the documents in a space with a lower dimension than the bag-of-words space (for instance to perform a supervised task), the authors of the paper have chosen K variational parameters from the posterior distribution approximation. This gives a representation of documents in terms of topics instead of a representation in terms of words.

For many reasons mentioned above, LDA is an interesting model of document density. However, the approximate inference algorithm is not easy to implement. Therefore, as a first step, another model, tractable by exact inference, will be proposed in Chapter 4.

2.3.3 Other Approaches

Kernel Approaches

As mentioned previously, we can consider kernel approaches as being part of a search for a richer representation for texts. Instead of focusing on a new representation $\phi(d)$ of a document d these approaches are interested in the scalar products $< \phi(d_i), \phi(d_j) >= k(d_i, d_j)$ between documents d_i and d_j, which are required for solving tasks such as text categorization with SVM.

The Latent Semantic Kernel proposed by Cristianini et al. (2002), is the kernel version of LSA presented in Section 2.3.1. The Gram matrix of the latent semantic kernel is:

$$K = [k(d_i, d_j)]_{i,j=1}^{N} = \hat{X}'\hat{X} = \hat{V}\hat{\Sigma}^2\hat{V}',$$

using the notation defined in Section 2.3.1 and where matrices $\hat{\Sigma}^2$ and $\hat{V}$ are obtained by an eigenvalue decomposition of the matrix $X'X = V\Sigma^2V'$ and taking the Q first eigenvalues and corresponding eigenvectors. An iterative algorithm for approximating K is proposed by Cristianini et al. (2002) in order to avoid the expensive eigenvalue decomposition.

Hofmann (2000) proposes two kernel functions derived from the PLSA model:

$$\tilde{k}(d_i, d_j) = \sum_{k=1}^{K} \frac{P(T=k|d_i)P(T=k|d_j)}{P(T=k)} \qquad \text{and}$$

$$\bar{k}(d_i, d_j) = \sum_{l=1}^{M} \hat{P}(w_l|d_i)\hat{P}(w_l|d_j) \sum_{k=1}^{K} \frac{P(T=k|d_i,w_l)P(T=k|d_j,w_l)}{P(w_l|T=k)}$$

with $\hat{P}(w_l|d) = \frac{\mathrm{tf}_l(d)}{n(d)}$. They can be respectively interpreted as capturing documents containing synonyms (scalar product in the "topics" space) and discriminating between documents containing polysems (scalar product in the bag-of-words space weighted by the posterior probabilities overlap). $\tilde{k}$ (resp. $\bar{k}$) can be derived by considering the scalar product of the Fisher scores of the likelihood of the documents with respect to $2\sqrt{P(T=k)}$, (resp. $2\sqrt{P(w_l|T=k)}$) normalized by the corresponding Fisher information.

Auxiliary Expert Knowledge

The two following approaches attempt at including information extracted from expert annotated hierarchical semantic databases. The Semantic kernel proposed by Siolas and d'Alché Buc (2000) is based on a proximity between terms constructed with the help of the *Wordnet* database (Miller et al., 1993a) containing links between words. The result on a text categorization task reported for this method are quite good for very small dictionaries.

The Feature Generator proposed by Gabrilovich and Markovitch (2005) is a kind of "document expansion" (see Section 2.2.1). It allows to enrich the bag-of-words with new words extracted from the *Open Directory Project* (Internet based URL repositories maintained by volunteers, much bigger and less constraint than Wordnet) taking advantage of the expert knowledge encoded there. They also report good result on a text categorization task.

A third approach related the previous ones but relying on slacker expert knowledge is the LinkLearn algorithm presented by Grangier and Bengio (2005). The idea is to learn the optimal term weighting of the *bag-of-words* representation for a document retrieval task. This is done by optimizing a ranking criterion similar to the one we use in Section 6.1, over a huge hyper-linked corpus of document, *Wikipedia*. It is assumed that documents linked should be more similar than documents not linked, in the same way as a query and its relevant documents with respect to a query and any irrelevant document. Because the task solved to learn the document representation is very close to the targeted task we may say that the representation is optimized for the document retrieval task.

2.4 Conclusion

In this chapter we have introduced the machine learning framework in which algorithms for solving problems are built by taking advantage on the statistical properties of the environment the problem lies on. We have also described the information retrieval field and in particular two of its interesting problems: document retrieval and text categorization. We have emphasized the fact that information retrieval systems performances may suffer from a too simple representation of the documents. Several machine learning methods developed to enrich this representation were presented. Before going forward to a description of our contribution to the area of text representation, the next chapter is devoted to the measures of performance commonly used to evaluate document retrieval and text categorization systems and to some propositions we made for the analysis of these evaluation techniques.

Chapter 3

Databases and Performance Measures

In order to discriminate among different approaches to solve a given problem, the definition of a criterion of interest and of a benchmarking protocol are very useful. In general, researchers design a measure of performance that encodes which are the important criteria and agree on a common protocol to test and to let their models compete. Indeed, even if it could be said that benchmarking protocols are restrictive and sometimes unrealistic, the chosen measure always partial, they nevertheless provide a common territory for the objective comparison of different approaches.

In general, the benchmarking protocol intents to be as characteristic as possible of the kind of real life problems to solve. Usually a database to test the different approaches on, is created from a real problem.

In a similar way, the chosen measures of performance are intended to judge the approach with respect to desirable behaviors, which are characteristic of the specific problem. In several application domains, such as information retrieval but also as it will be developed in Section 3.3 in person authentication applications, two criteria of interest are antagonist. In such cases, a measure is usually chosen which aims at giving a picture of the performance of the model at several compromise points between the extreme positions of each criterion.

Finally, to complete the tools needed for performance evaluation, statistical significance tests need to be added. Indeed, obtaining some result by measuring the performance of systems on a benchmark database, does not explain how general this result is, if it is reproducible on a slightly different database. Statistical significance hypothesis tests are designed to answer this question.

This chapter is organized as follows: Section 3.1 describes the databases on which the models presented in this thesis dissertation were tested, Section 3.2 defines the measures of performance commonly used in the information retrieval community, Section 3.3 explains drawbacks that some of these measures may have and offers a solution, finally Section 3.4 proposes a statistical significance test adapted to one of the measures used in information retrieval and benchmarks it in several conditions.

3.1 Databases

In order to compare several approaches to a given problem, experiments over benchmark databases are performed. For the results to be fully comparable, the experiments involving the different approaches must have been performed according to the exact same protocol, that is on the same database, using the same split of the database examples into training and test sets and with the same evaluation measure.

In this section the databases used in the remainder of this thesis are described:

- **TDT-2** (Cieri et al., 1999) is a database of transcribed broadcast news in American English. Two transcriptions are available, a manually annotated one and one obtained with an automatic speech recognition system, in addition to a manual segmentation of data into approximately 25 000 news-stories. For our experiments we used 24 823 documents from a manually produced transcription and segmentation, referred to in the following as TDT2-clean. Two sets of 50 queries for documents of TDT-2, called TREC-8 and TREC-9 were collected during TREC SDR evaluation (*ie* benchmark queries used at the Spoken Document Retrieval session of the Text Retrieval Conference in 1999 and 2000, Garofolo et al., 2000). In this **document retrieval** classical setting, the database documents are available as development data as well as the TREC-8 queries and their corresponding relevance judgements associated to the documents of TDT2, while the TREC-9 queries are for evaluation only.

- **Reuters-21578**[1] is probably the benchmark database which has been used the most to test **text categorization** systems. Reuters-21578 is a corpus of economic news-stories of the news agency Reuters Ltd, annotated with 135 topic categories. Several protocols have been used over this corpus, however the most commonly used is the so-called *ModApte* split, which creates from the 21,578 documents of the corpus a training set of 9,603 examples and a test set of 3,299 examples. On average each document is associated with a little bit more than one category. Table 3.1 reports the percentage of documents of the training set which attributed to each of the ten most frequently assigned categories.

  It can be noted that the two-class classification problems resulting from this task exhibit a huge imbalance between the number of positive (documents belonging to the category) and the number of negative examples.

category	earn	acq	money-fx	grain	crude	trade	interest	wheat	ship	corn
%	30.0	17.2	5.6	4.5	4.1	3.8	3.6	2.2	2.1	1.9

Table 3.1: Percentage of the Reuters-21578 ModApte split training documents associated with each of the ten most populated categories.

- **RCV1** (Reuters Corpus, Version 1, Rose et al., 2002) is the more recent corpus released by the news agency Reuters Ltd, and made available for research. It is much bigger than Reuters-21578, with 806,791 news-stories and which are very precisely annotated. Among the annotations provided

[1]Currently available at `http://www.daviddlewis.com/resources/testcollections/reuters21578`

with the documents on can find the thematic labels selected among 101 possible labels organized in a hierarchy. Note however that in our work we ignored the hierarchical information. Despite the fact that a public protocol have also been released by Lewis et al. (2004), we have designed specific protocols to compare our approaches to state-of-the-art. Instead of splitting the corpus in a little part for training and the remainder for the test set, we used this huge corpus to sample with replacement several datasets issued from the same distribution, and thus obtained an estimate of the variance of the results with respect to the choice of the training/test set (see Section 3.4 and Section 6.2).

3.2 Precision, Recall and F_1 Score

Two-class classification tasks are a kind of problems commonly handled by machine learning tools. Usually, in order to measure the performance of the learned classifiers, the ratio of the number of errors over the number of presented examples is computed.

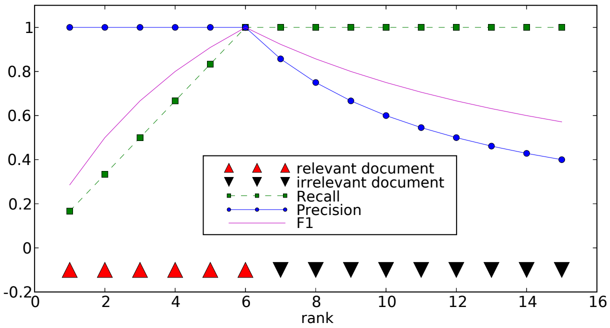
In the information retrieval community, some problems are very close to two-class classification problems and, as it has been mentioned in Chapter 2, they have been successfully modeled with machine learning tools. However, in order to measure the performance of the models, instead of the classification error, information retrieval standard measures are used. These measures are designed to evaluate the effectiveness of the information systems according to a range of specifiable user needs.

As explained in Section 2.2, document retrieval can be seen as the task of ranking documents in a database to find the documents which are more relevant to a query need. To evaluate the effectiveness of the ranking for the information retrieval scope, three sets are considered of interest: the set of selected documents $\mathcal{S}$, the set of relevant documents $\mathcal{R}$ and their overlap $\mathcal{S} \cap \mathcal{R}$. The two measures commonly used in Information Retrieval are:

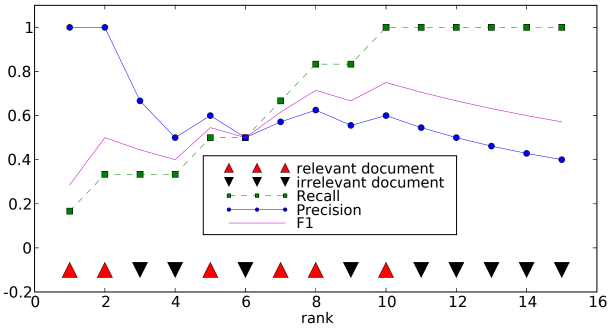
$$\text{Precision} = \frac{|\mathcal{S} \cap \mathcal{R}|}{|\mathcal{S}|}, \quad \text{and} \quad \text{Recall} = \frac{|\mathcal{S} \cap \mathcal{R}|}{|\mathcal{R}|} \tag{3.1}$$

where $|\mathcal{A}|$ is the number of items in set $\mathcal{A}$. Precision measures the proportion of relevant documents among the selected ones, and Recall the proportion of selected documents among the relevant ones. Precision and Recall are effectiveness measures, *i.e.* inside $[0, 1]$ interval, the closer to 1 the better. The behavior of Precision and Recall is illustrated for two toy rankings in Figure 3.1. In an ideal ranking as presented in Figure 3.1(a), all relevant documents (red triangles, pointing up) should be ranked before irrelevant documents (black triangles, pointing down). Precision value is 1, until irrelevant documents are included in $\mathcal{S}$. Simultaneously, Recall increases slowly until all relevant documents are selected. Is it quite clear in this Figure that the system should stop retrieving documents when both Precision and Recall are equal to 1. However, in the more realistic toy ranking presented in Figure 3.1(b), it is unclear when the system should stop retrieving documents and it shows that there is a necessary trade-off between Precision and Recall.

For that reason, the *Receiver Operating Characteristic* (ROC) curve (Green and Swets, 1964) is popular in the information retrieval community. The ROC curve in this field is the graph of Precision against



(a) Ideal ranking



(b) Realistic ranking

Figure 3.1: Precision, Recall and F_1 score when increasing the number of selected documents along the ranking.

Recall when the number of documents selected from the ranking is increased . This graph illustrates the different operating points of the ranking, *ie* the possible Precision, Recall trade-off points the user can choose. See for example Figure 3.2 presenting the ROC curve for the toy ranking of Figure 3.1(b).

In order to extract a single value from this curve the area under the ROC curve, the average Precision (average of Precision at every point of the ranking) or the *eleven-point average Precision* at standard Recall values (average of the maximum values of Precision at Recall $= 0, 0.1, 0.2, \ldots, 1$) are sometime computed. Instead of considering the entire ranking the Precision at top n, the *break-even* point or the maximum of the F_1 score can also be computed as a performance measure. Precision at top n is the value of Precision when selecting only the n first documents in the ranking. The break-even point is the arbitrary choice of taking Precision as close as possible to Recall. Finally, the harmonic mean[2] of

[2]The harmonic mean $\left(\frac{1}{2}(a^{-1}+b^{-1})\right)^{-1}$ tends to be closer than the arithmetic mean $\frac{1}{2}(a+b)$ to the smallest element:

$$min(a,b) \leq \text{harmonic mean} \leq \text{geometric mean} \leq \text{arithmetic mean} \leq max(a,b).$$

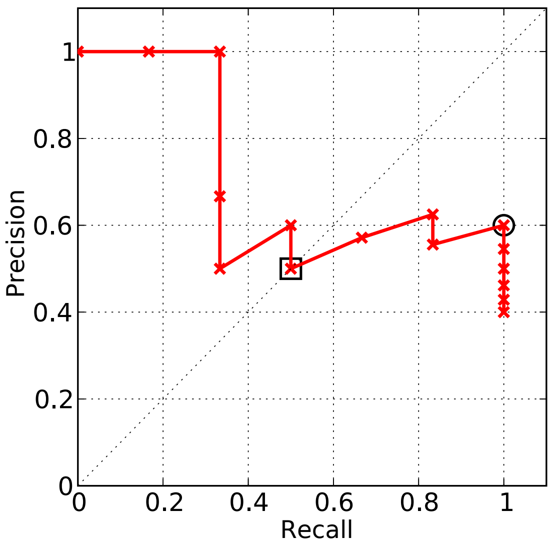


Figure 3.2: ROC curve for the toy ranking in Figure 3.1(b)

Precision and Recall:

$$F_1 = \left(\frac{1}{2}\left[\frac{1}{\text{Recall}} + \frac{1}{\text{Precision}}\right]\right)^{-1} = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}, \tag{3.2}$$

was proposed by van Rijsbergen (1975) to summarize these two measures. Choosing to stop in the ranking when F_1 is maximum can be seen as taking a better compromise between Precision and Recall. See Baeza-Yates and Ribeiro-Neto (1999) for more details.

Text categorization as explained in Section 2.2, is a particular sub-problem of information retrieval with the aim of assigning one or several categories, among a predefined set of K categories, to textual documents. Most approaches to text categorization transform the problem into K two-class classification tasks, each category in a one-against-the-others scheme. For each classification sub-task k a function $f_k(.)$ and a threshold θ_k are looked for, such that for any document x belonging to (resp. not belonging to) the category k, $f_k(x) > \theta_k$ (resp. $< \theta_k$). The traditional IR measures are computed as:

$$\text{Precision} = \frac{N_{tp}}{N_{tp} + N_{fp}}, \quad \text{Recall} = \frac{N_{tp}}{N_{tp} + N_{fn}} \quad \text{and} \quad F_1 = \frac{2N_{tp}}{2N_{tp} + N_{fn} + N_{fp}}$$

where for each category, N_{tp}, N_{fp} and N_{fn} are three of the different types of outcome th classifier as illustrated in Table 3.2. N_{tp} is the number of true positives (documents belonging to the category that were classified as such), N_{fp} the number of false positives (documents out of this category but classified as being part of it) and N_{fn} the number of false negatives (documents from the category classified as out of it).

According to Sebastiani (2002), approaches to solve this problem can be divided into two sets: the

		Desired Class	
		1	0
Obtained Class	1	N_{tp}	N_{fp}
	0	N_{fn}	N_{tn}

Table 3.2: Counts of the different types of outcome of a two-class classification problem.

"hard" categorization versus the ranking categorization. Note that Precision being a measure of the quality of the ranking, it is more appropriate to the second set of approaches than to the first, however it is used in both cases.

To measure the joint performance of the K classifiers two averaging schemes can be applied. In the *macro*-average scheme, each category has the same importance, however populated it is. *Macro*-averaged Precision and Recall are computed as:

$$\text{Precision}^M = \frac{1}{K}\sum_{k=1}^{K}\text{Precision}_k \quad \text{and} \quad \text{Recall}^M = \frac{1}{K}\sum_{k=1}^{K}\text{Recall}_k,$$

where Precision_k and Recall_k are the Precision and Recall values of classifier number k. On the other hand, in the *micro*-average scheme, the same importance is given to each decision taken at document level. From *micro*-average point of view it is more important to correctly classified documents for a populated category than for an infrequent one. The equations for computing *micro*-averaged Precision and Recall are the following:

$$\text{Precision}^\mu = \frac{\sum_{k=1}^{K} N_{tp}^k}{\sum_{k=1}^{K} N_{tp}^k + N_{fp}^k} \quad \text{and} \quad \text{Recall}^\mu = \frac{\sum_{k=1}^{K} N_{tp}^k}{\sum_{k=1}^{K} N_{tp}^k + N_{fn}^k},$$

where N_{tp}^k (resp. N_{fp}^k and N_{fn}^k) corresponds to the number of true positives (resp. false positives and false negatives) of classifier for category k.

3.3 Expected Performance

In several other domains, requirements similar to that of Information Retrieval are expected of the trained systems. The aspects these domains have in common is that on top of selecting the appropriate discriminant function, practitioners also modify the corresponding threshold in order to better suit an independent cost function. Moreover, they compare models with respect to the whole range of possible values this threshold could take, generating curves such as ROCs. In order to also provide quantitative comparisons, they often select one particular point on this curve (such as the so-called *break-even point* or *equal error rate*).

The main purpose of this Section is to argue that such curves, as well as particular points on it like *break-even point* or *equal error rate* can be misleading when used to compare two or more models, or computed to obtain a realistic estimate of the expected performance of a given model.

We thus propose instead the use of a new set of curves, called *Expected Performance Curves* (EPC), which really reflect the expected (and reachable) performance of systems. While EPCs are presented here for general machine learning tasks, they were first presented specifically in the context of person authentication in Bengio and Mariéthoz (2004).

3.3.1 General Framework

In several domains of application, there are tasks which are in fact specific incarnations of two-class classification problems. However, often for historical reasons, researchers specialized in these tasks have chosen different methods to measure the quality of their systems. In general, the selected measures come by pair, which we will call generically here $V1$ and $V2$, and are simple antagonist combinations of N_{tp}, N_{tn}, N_{fp} and N_{fn} values. As defined in Table 3.2, these values are obtained by counting among examples x, the different comparison outcomes of the classification score $f(x)$ to a predefined threshold θ. If $f(x) > \theta$ then the obtained class is the class 1, otherwise it is class 0.

Moreover, a unique measure (V) often combines $V1$ and $V2$. For instance,

- as explained in the previous section, in the domain of text categorization,

$$V1 = \text{Precision}, \quad V2 = \text{Recall and } V = F_1 \ , \tag{3.3}$$

- in the domain of person authentication (Verlinde et al., 2000), the chosen measures are

$$V1 = \text{FAR} = \frac{N_{fp}}{N_{fp} + N_{tp}} \text{ and } V2 = \text{FRR} = \frac{N_{fn}}{N_{fn} + N_{tp}}. \tag{3.4}$$

  They are called *false acceptance rate* (FAR) and *false rejection rate* (FRR) respectively. Several aggregate measures have been proposed, the simplest being the *half total error rate* (HTER)

$$V = \frac{V1 + V2}{2} = \frac{\text{FAR} + \text{FRR}}{2} = \text{HTER} \ , \tag{3.5}$$

- in medical studies,

$$V1 = \frac{N_{tp}}{N_{tp} + N_{fn}} \text{ and } V2 = \frac{N_{tn}}{N_{tn} + N_{fp}} \tag{3.6}$$

  and are called *sensitivity* and *specificity* respectively (Zweig and Campbell, 1993).

In all the cases, in order to use the system effectively, one has to select the threshold θ according to some criterion, which is in general of the following generic form:

$$\theta^\star = \arg\min_\theta g(V1(\theta), V2(\theta)) \ . \tag{3.7}$$

Examples of $g(\cdot, \cdot)$ are the HTER and *F1* functions already defined in equations (3.5) and (3.2) respectively. However, the most often used criterion in text categorization is the *break-even point* (BEP) and in person authentication it is the *equal error rate* (EER). They both correspond to the threshold nearest to a solution such that $V1 = V2$, often estimated as follows:

$$\theta^\star = \arg\min_\theta |\text{V1}(\theta) - \text{V2}(\theta)| \,. \tag{3.8}$$

As already discussed in the previous section for the information retrieval domain, the choice of the threshold can have a significant impact in the resulting system: in general θ represents a trade-off between giving importance to $V1$ or $V2$. As an example, see the very different Precision/Recall compromises obtained in Figure 3.2 when taking as criterion the BEP (square mark) or the maximum of the F_1 score (circle mark). Hence, instead of committing to a single operating point, an alternative method to present results often used in the research community is to produce a graph that presents $V1$ with respect to $V2$ for all possible values of θ, plotting a ROC curve[3] (see Section 3.2). Figure 3.3 shows two typical ROCs. Note that depending on the precise definition of $V1$ and $V2$, the best curve would tend to one of the four corners of the graph. In Figure 3.3, the best curve corresponds to the one nearest to the top right corner (corresponding to simultaneous high values of $V1$ and $V2$).

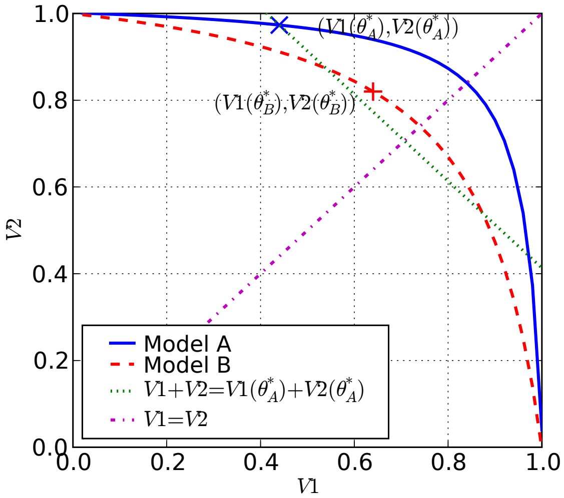


Figure 3.3: Example of two typical ROCs.

Instead of providing the whole ROC, researchers often summarize it by some typical values taken from it; the most common summary measure is computed by using the BEP, described in equation (3.8), which produces a single value of θ and from which some aggregate value $V(\theta)$ (such as *F1* or HTER) can be computed. On Figure 3.3, the pink dashed and dotted line intersecting the two ROCs is the BEP line and the intersections with each ROC correspond to their respective BEP point.

[3]Note that the original ROC plots the true positive rate with respect to the false positive rate, but several researchers use the name *ROC* with various other definitions of $V1$ and $V2$.

3.3.2 Cautious Interpretation of ROC and BEP

As explained above, researchers often use ROC and BEP to present and compare their results; for example, all results presented in Sebastiani (2002), which is a very good survey of text categorization, are presented using the BEP; a recent and complete tutorial on text independent speaker verification (Bimbot et al., 2004) proposes to measure performance through the use of DET curves, which are non-linear versions of ROCs, as well as the error corresponding to equal error rate, hence the BEP. We would like here to point out some potential risk of using ROC or BEP for comparing two systems, as it is done for instance in Figure 3.3, where the test performance of models A and B are compared. As can be seen on this figure, and reminding that in this case $V1$ and $V2$ must be maximized, the best model appears to be model A for all operating points, since its curve is always above that of model B. Moreover, computing the BEP of models A and B yields the same conclusion.

Let us now remind that each point of the ROC corresponds to a particular setting of the threshold θ. However, in real applications, θ needs to be decided prior to seeing the test set. This is in general done using some criterion of the form of equation (3.7) such as searching for the BEP, equation (3.8), using some development data (obviously different from the test set).

Hence, assuming for instance that one decided to select the threshold according to (3.8) on a development set, the obtained threshold may not correspond to the BEP on the test set. There are many reasons that could yield such mismatch, the simplest being that assuming the test and development sets to come from the same distribution but be of fixed (non-infinite) size, the estimate of (3.8) on one set is not guaranteed to be the same as the estimate on the other set.

Let us call $\theta^\star_A$ the threshold estimated on the development set using model A and similarly for $\theta^\star_B$. While the hope is that both of them should be aligned, on the test set, with the BEP line, there is nothing, in theory, that prevents them to be slightly or even largely far from it. Figure 3.3 illustrates such an example, where indeed,

$$V1(\theta^\star_B) + V2(\theta^\star_B) > V1(\theta^\star_A) + V2(\theta^\star_A) \tag{3.9}$$

(see green diagonal dotted line) even though the ROC of model A is always above that of model B, including at the intersection with the BEP line[4]. One might argue that this may only rarely happen, but this scenario has indeed been observed several times in person authentication and text categorization tasks, including a text independent speaker verification application where the problem is described in more details in Bengio and Mariéthoz (2004) and a categorization problem which is described in the following. Two models, *model A* and *model B*, based on two different classification algorithms, have been trained on the Reuters-21578 news stories database described in Section 3.1, with the ModApte split, for the EARN category against the 114 others. We replicate in Figure 3.4 the ROCs obtained on this task using two different models, with model B apparently always better than model A. However, when selecting the threshold on a separate validation set (hence simulating a real life situation), the sum of Precision and Recall of model A becomes lower than that of model B (the graph shows the operating

[4]Note that at BEP, $V1 = V2 = \frac{1}{2}(V1 + V2)$.

points corresponding to BEP computed on a separated development set for the two models).

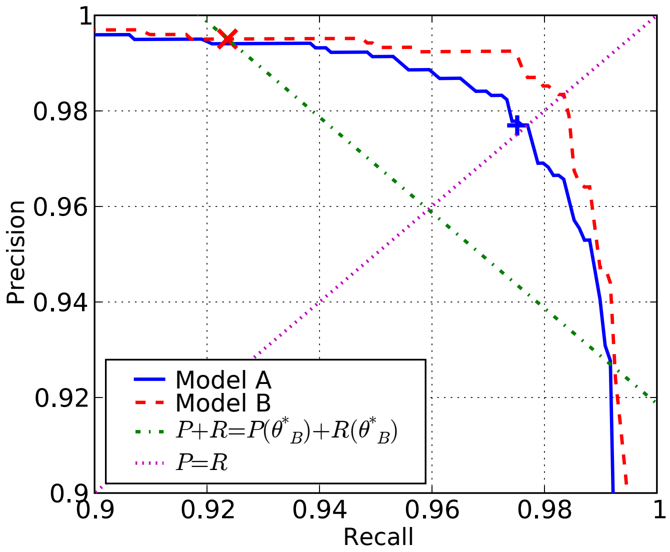


Figure 3.4: ROCs of two real models for a Text Categorization task.

In summary, showing ROCs have potentially the same drawbacks and risks as showing the training error (indeed, one parameter, the threshold, has been implicitly tuned on the test data), expecting that it reflects the expected generalization error: this is true when the size of the data is huge, but false in the general case. Furthermore, real applications often suffer from an additional mismatch between training and test conditions which should be reflected in the procedure.

3.3.3 The Expected Performance Curve

We have seen in Section 3.3 that given the trade-off between $V1$ and $V2$, researchers often prefer to provide a curve that assesses the performance of their model for all possible values of the threshold. On the other hand, we have seen in Section 3.3.2 that ROCs can be misleading since selecting a threshold prior to seeing the test set (as it should be done) may end up in obtaining a different trade-off in the test set. Hence, we would like here to propose the use of new curves which would let the user select a threshold according to some criterion, in an unbiased way, and still present a range of possible expected performances on the test set. We shall call these curves Expected Performance Curves (EPC).

Definition

The general framework of EPCs is the following. Let us define some parametric performance measure $C(V1(\theta, D), V2(\theta, D); \alpha)$ which depends on the parameter α as well as $V1$ and $V2$ computed on some data D for a particular value of θ. Examples of $C(\cdot, \cdot; \alpha)$ are the following:

- in text categorization, since the goal is to maximize Precision and Recall, one could use

$$\begin{aligned} & C(V1(\theta, D), V2(\theta, D); \alpha) \\ & \quad = C(\text{Precision}(\theta, D), \text{Recall}(\theta, D); \alpha) \\ & \quad = -(\alpha \cdot \text{Precision}(\theta, D) + (1-\alpha) \cdot \text{Recall}(\theta, D)) \end{aligned} \tag{3.10}$$

  where $V1$ is the Precision and $V2$ is the Recall;

- in person authentication, one could use for instance

$$\begin{aligned} & C(V1(\theta, D), V2(\theta, D); \alpha) \\ & \quad = C(\text{FAR}(\theta, D), \text{FRR}(\theta, D); \alpha) \\ & \quad = \alpha \cdot \text{FAR}(\theta, D) + (1-\alpha) \cdot \text{FRR}(\theta, D) \end{aligned} \tag{3.11}$$

  which basically varies the relative importance of $V1$ (FAR) with respect to $V2$ (FRR); in fact, setting $\alpha = 0.5$ yields the HTER cost (3.5);

- in general, one could also be interested in trying to reach a particular relative value of $V1$ (or $V2$), such as *I am searching for a solution with as close as possible to 10% false acceptance rate*; in that case, one could use

$$C(V1(\theta, D), V2(\theta, D); \alpha) = |\alpha - V1(\theta, D)| \tag{3.12}$$

  or

$$C(V1(\theta, D), V2(\theta, D); \alpha) = |\alpha - V2(\theta, D)| \,. \tag{3.13}$$

Having defined $C(\cdot, \cdot; \alpha)$, the main procedure to generate the EPC is to vary α inside a reasonable range (say, from 0 to 1), and for each value of α, to estimate θ that minimizes $C(\cdot, \cdot; \alpha)$ on a development set, and then use the obtained θ to compute some aggregate value (say, V), on the test set. Algorithm 3.1 details the procedure, while Figure 3.5 shows an artificial example of comparing the EPCs of two models. Looking at this figure, we can now state that for specific values of α (say, between 0 and 0.5), the underlying obtained thresholds are such that model B is better than model A, while for other values, this is the converse. This assessment is unbiased in the sense that it takes into account the possible mismatch one can face while estimating the desired threshold.

Let us suppose that Figure 3.5 was produced for a text categorization task, where V is the F_1 score, $V1$ is Precision, and $V2$ is Recall. Furthermore let us define the criterion as in (3.10). In that case, α varies from 0 to 1, and when $\alpha = 0.5$ this corresponds to the setting where we tried to obtain a BEP, while when $\alpha < 0.5$ it corresponds to settings where we gave more importance to Recall and when $\alpha > 0.5$ we gave more importance to Precision.

In order to illustrate EPCs in real applications, we have generated them for both a person authentication task and a text categorization task. The resulting curves can be seen in Figures 3.7 and 3.6. Note that the graph reporting $F1$ seems inverted with respect to the one reporting HTER, but this is because we are searching for low HTERs in person authentication but high $F1$ in text categorization. Note also

Algorithm 3.1 Method to generate the Expected Performance Curve

Let *devel* be the development set
Let *test* be the test set
Let $V(\theta, D)$ be the value of V obtained on the data set D for threshold θ
Let $C(V1(\theta, D), V2(\theta, D); \alpha)$ be the value of a criterion C that depends on α, and is computed on the data set D
for values $\alpha \in [a, b]$ where a and b are reasonable bounds **do**
  $\theta^\star = \arg\min_\theta C(V1(\theta, devel), V2(\theta, devel); \alpha)$
  compute $V(\theta^\star, test)$
  plot $V(\theta^\star, test)$ with respect to α
end for

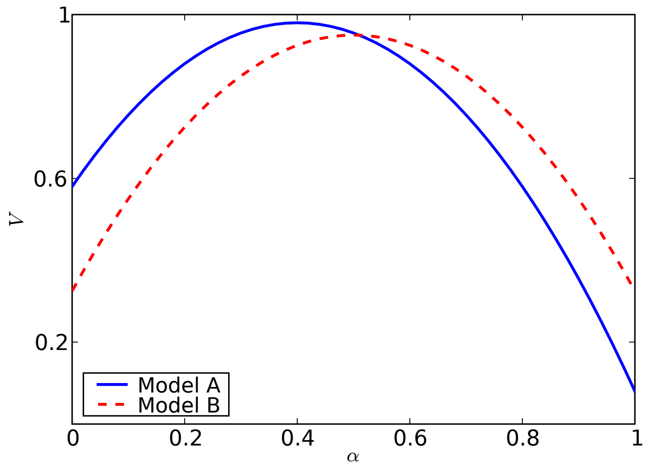


Figure 3.5: Example of two typical EPCs.

that the EPC of Figure 3.6 corresponds to the ROC of Figure 3.4. Finally, note that we kindly provide a C++ tool that generates such EPCs[5].

Areas Under the Expected Performance Curves

In general, people often prefer to compare their models according to a unique quantitative performance measure, rather than through the use of curves which can be difficult to interpret. One solution proposed by several researchers is to summarize the ROC by some approximation of the area under it.

Knowing that the ROC may in fact be a misleading measure of the expected performance of a system, the corresponding area under it may also be misleading. Would it be possible to obtain a measure of the expected performance over a given range of operating points? We propose here to compute $E[\bar{V}]$, the expected value of $\bar{V}$, which would be defined as the average between two antagonist measures $V1$ and $V2$ given a criterion $C(\cdot, \cdot; \alpha)$. We will show that this is in fact related to the area under the

[5]An EPC generator is available at `http://www.Torch.ch/extras/epc` as a package of the Torch machine learning library.

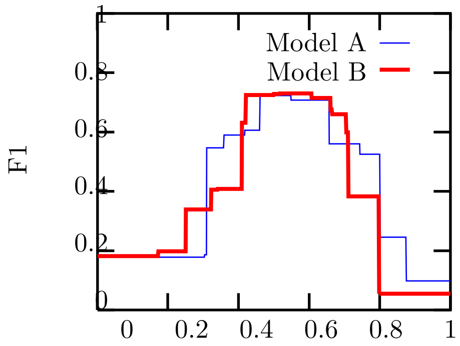


Figure 3.6: Expected Performance Curves for text categorization, where one wants to trade-off precision and recall and print the $F1$ measure.

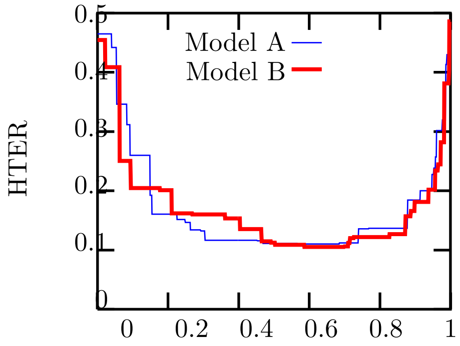


Figure 3.7: Expected Performance Curves for person authentication, where one wants to trade-off false acceptance rates with false rejection rates.

ROC curve (AUC), although it now concerns an area under a curve of *reachable* solutions instead of theoretical solutions. Note that there are several theoretical properties of the AUC measure which makes it appealing, such as the fact that, when $V1$ and $V2$ are respectively defined as the true and false positive rates, it corresponds to the well-known Mann-Whitney statistical test which can be used to compare two independent groups of sampled data (Hanley and McNeil, 1982).

Let $\theta_{f=\alpha}$ be the threshold such that

$$\theta_{f=\alpha} = \arg\min_{\theta} |\alpha - f(\theta)|, \tag{3.14}$$

we can write the expected value of $\bar{V} = \frac{V1+V2}{2}$ using $V1$ as a threshold selection criterion, as follows:

$$E_{V1}[\bar{V}] = \frac{1}{2} \int_{\alpha \in [0,1]} [V1(\theta_{V1=\alpha}) + V2(\theta_{V1=\alpha})] \, d\alpha, \tag{3.15}$$

and using $V2$ as criterion,

$$E_{V2}[\bar{V}] = \frac{1}{2}\int_{\beta\in[0,1]} [V1(\theta_{V2=\beta}) + V2(\theta_{V2=\beta})]\, d\beta. \tag{3.16}$$

Note that if we select the thresholds θ on the test set then,

$$\begin{gathered} V1(\theta_{V1=\alpha}) = \alpha, \quad V2(\theta_{V2=\beta}) = \beta, \\ \int_{\alpha\in[0,1]} V2(\theta_{V1=\alpha})d\alpha = \text{AUC} \quad \text{and} \\ \int_{\beta\in[0,1]} V1(\theta_{V2=\beta})d\beta = \text{AUC}\ . \end{gathered} \tag{3.17}$$

Thus, using the fact that $\int_0^1 \gamma d\gamma = \frac{1}{2}$, we can obtain the relation between the expected $\bar{V}$ when the thresholds are computed on the test set (which we will call $E[\bar{V}]_{post}$) and the area under the ROC, by computing the average of equations (3.15) and (3.16) when the threshold is chosen on the test set:

$$\begin{aligned} G(\bar{V})_{post} &= \frac{1}{2}\left\{E_{V1}[\bar{V}]_{post} + E_{V2}[\bar{V}]_{post}\right\} \\ &= \frac{1}{2}\left\{\text{AUC} + \frac{1}{2}\right\}. \end{aligned} \tag{3.18}$$

Of course, if we select the thresholds using a separate development set, the result obtained in (3.18) is not true anymore. However, in this case the average $G(\bar{V})$ remains interesting since it can be interpreted as a measure summarizing two EPCs. Indeed, the two components of the average, (3.15) and (3.16), are the area under an EPC computed using respectively criteria (3.12) and (3.13), hence it integrates two antagonist performance measures over a large range of operating points.

Note that in equations (3.15) and (3.16), we integrate $\bar{V}$ over expected values of $V1$ and $V2$ from 0 to 1. However in some cases, a value of $V1$ around, say 0, may be reachable but of no interest. In text categorization, for example, it may be “reasonable” not to pay attention to values of Precision and Recallsmaller than 0.5. The values of “reasonable” bounds are task dependent, but their choice can be decisive, and should be taken into account when computing the expected performance of the system.

3.4 A Statistical Significance Test for the Difference of F_1

Statistical tests are often used in machine learning in order to assess the performance of a new learning algorithm or model over a set of benchmark datasets, with respect to the state-of-the-art solutions. Several researchers (see for instance Dietterich, 1998; Nadeau and Bengio, 2003) have proposed statistical tests suited for two-class classification tasks where the performance is measured in terms of the classification error (ratio of the number of errors and the number of examples), which enables the use of assumptions based on the fact that the error can be seen as a sum of random variables over the evaluation examples. On the other hand, as mentioned previously, various research domains prefer to measure the performance of their models using different indicators, such as the F_1 measure described in Section 3.2. Most classical statistical tests cannot cope directly with such measure as the usual necessary assumptions are no longer correct, and non-parametric bootstrap-based methods are then used (Efron and Tibshirani, 1993). Since several papers already use these non-parametric tests (Bolle et al., 2004; Bisani and Ney, 2004), we were interested in verifying empirically how reliable they were. For this purpose, we used a very large text categorization database the extended Reuters dataset described in Section 3.1.

We purposely set aside the largest part of the dataset and considered it as the whole population, while a much smaller part of it was used as a training set for the models. The large dataset part that was set aside is used to *test* the statistical test in the same spirit as was done in Dietterich (1998), by sampling evaluation sets over which we observed the performance of the models and the behavior of the significance test.

Following the questions of interest taxonomy defined in Dietterich (1998), we can differentiate between statistical tests that analyze learning algorithms and statistical tests that analyze classifiers. In the first case, one intends to be robust to possible variations of the train and evaluation sets, while in the latter, one intends to only be robust to variations of the evaluation set. While the methods discussed in this section can be applied alternatively to both approaches, we concentrate here on the second one, as it is more tractable (for the empirical section) while still corresponding to real life situations where the training set is fixed and one wants to compare two solutions (such as during a competition).

In order to conduct a thorough analysis, we tried to vary the evaluation set size, the class imbalance, the error measure, the statistical test itself (with its associated assumptions), and even the *closeness* of the compared learning algorithms. Section 3.4.3, is a detailed account of this analysis. As it will be seen empirically, the *closeness* of the compared learning algorithms seems to have an effect on the resulting quality of the statistical tests: comparing an MLP and an SVM yields less reliable statistical tests than comparing two SVMs with a different kernel. To the best of our knowledge, this has never been considered in the literature of statistical tests for machine learning.

3.4.1 Statistical Significance Test for the Difference of Classification Errors

Let us first remind the basic classification framework in which statistical significance tests are used in machine learning. We consider comparing two models A and B on a two-class classification task where the goal is to classify input examples x_i into the corresponding class $y_i \in \{-1, 1\}$, using already trained models $f_A(x_i)$ or $f_B(x_i)$. One can estimate their respective performance on some test data by counting the number of utterances of each possible outcome: either the obtained class corresponds to the desired class, or not. Let $N_{e,A}$ (resp. $N_{e,B}$) be the number of errors of model A (resp. B) and N the total number of test examples; The difference between models A and B can then be written as

$$D = \frac{N_{e,A} - N_{e,B}}{N} \; . \tag{3.19}$$

The usual starting point of most statistical tests is to define the so-called *null hypothesis* H_0 which considers that the two models are equivalent, and then verifies how probable this hypothesis is. Hence, assuming that D is an instance of some random variable $\mathbf{D}$ which follows some distribution, we are interested in

$$p\left(|\mathbf{D}| < |D|\right) < \alpha \tag{3.20}$$

where α represents the risk of selecting the *alternate hypothesis* (the two models are different) while the *null hypothesis* is in fact true. α is also referred at as the *level of the test*. This probability can in general be estimated easily when the distribution of $\mathbf{D}$ is known. In the simplest case, known as the *proportion test*, one assumes (reasonably) that the decision taken by each model on each example can be modeled by a Bernoulli, and further assumes that the errors of the models are independent. This is in general wrong in machine learning since the evaluation sets are the same for both models. When N is large, this leads to estimate $\mathbf{D}$ as a Normal distribution with zero mean and standard deviation σ_D

$$\sigma_D = \sqrt{\frac{2\bar{C}(1-\bar{C})}{N}} \tag{3.21}$$

where $\bar{C} = \frac{N_{e,A}+N_{e,B}}{2N}$ is the average classification error. In order to get rid of the wrong independence assumption between the errors of the models, the McNemar test (Everitt, 1977) concentrates on examples which were differently classified by the two compared models. Following the notation of Dietterich (1998), let N_{01} be the number of examples misclassified by model A but not by model B and N_{10} the number of examples misclassified by model B but not by model A. It can be shown that the following statistics is approximatively distributed as a χ^2 with 1 degree of freedom:

$$z = \frac{(|N_{01} - N_{10}| - 1)^2}{N_{01} + N_{10}}. \tag{3.22}$$

More recently, several other statistical tests have been proposed, such as the 5x2cv method (Dietterich, 1998) or the variance estimate proposed in Nadeau and Bengio (2003), which both claim to better estimate the distribution of the errors (and hence the confidence on the statistical significance of the results). Note however that these solutions assume that the error of one model is the average of some random variable (the error) estimated on each example. Intuitively, it will thus tend to be Normally distributed as N grows, following the central limit theorem.

3.4.2 Bootstrap Percentile Test

Let us consider two models A and B, which achieve a performance measured by $F_{1,A}$ and $F_{1,B}$ respectively. The difference $dF_1 = F_{1,A} - F_{1,B}$ does not fit the assumptions of the tests presented in Section 3.4.1. Indeed, it cannot be decomposed into a sum over the documents of independent random variables, since the numerator and the denominator of dF_1 are non constant sums over documents of independent random variables. For the same reason F_1, while being a proportion, cannot be considered as a random variable following a Normal distribution for which we could easily estimate the variance.

An alternative solution to measure the statistical significance of dF_1 is based on the Bootstrap Percentile Test proposed in Efron and Tibshirani (1993). The idea of this test is to approximate the unknown distribution of dF_1 by an estimate based on bootstrap replicates of the data.

Given an evaluation set of size N, one draws, *with replacement*, N samples from it. This gives the first bootstrap replicate B_1, over which one can compute the statistics of interest, dF_{1,B_1}. Similarly, one can create as many bootstrap replicates B_n as needed, and for each, compute dF_{1,B_n}. The higher n is, the more precise should be the statistical test. Literature (Davison and Hinkley, 1997) suggests to create at least $\frac{50}{\alpha}$ replicates where α is the level of the test; for the smallest α we considered (0.01), this amounts to 5000 replicates. These 5000 estimates dF_{1,B_i} represent the non-parametric distribution of the random variable $\mathbf{dF_1}$. From it, one can for instance consider an interval $[a, b]$ such that $p(a < \mathbf{dF_1} < b) = 1 - \alpha$ centered around the mean of $p(\mathbf{dF_1})$. If 0 lies outside this interval, one can say that $dF_1 = 0$ is not among the most probable results, and thus reject the null hypothesis.

Other non-parametric tests such as the sign test are sometimes advocated in the related literature (see for instance van Rijsbergen, 1975; Yang and Liu, 1999). They may assess whether model A is better than model B, but the assessment will not be based on the difference of F_1, and thus will not give any answer about the significance of the difference of such measure.

Goutte and Gaussier (2005) have proposed a probabilistic interpretation of the F_1 measure. This interpretation allows the comparison of two models based on the probability that one model have better F_1 performance than the other. It does not provide, however, a test for the statistical significance of the difference of F_1. Thus, we will not consider it in the following analysis.

3.4.3 Analysis of Statistical Tests

We report in this section an analysis of the bootstrap percentile test, as well as other more classical statistical tests, based on a real large database. We first describe the database itself and the protocol we used for this analysis, and then provide results and comments.

Database, Models and Protocol

All the experiments detailed in this section were conducted on the very large RCV1 Reuters dataset. We divided the set of 806,791 documents as follows: 798,809 documents were kept aside and any statistics computed over this set D_{true} was considered as being the *truth* (*ie* a very good estimate of the actual value. See Appendix A.1, for an explanation on how the decision is taken); the remaining 7982 documents were used as a training set D_{tr} (to train models A and B). There was a total of 101 categories and each document was labeled with one or more of these categories.

We first extracted the dictionary from the training set, removed stop-words and applied stemming to it, as normally done in text categorization. Each document was then represented as a bag-of-words using the usual TFIDF coding. We trained three different models: a linear Support Vector Machine (SVM), a Gaussian kernel SVM, and a multi-layer perceptron (MLP). There was one model for each category for the SVMs, and a single MLP for the 101 categories. All models were properly tuned using cross-validation on the training set.

Using the notation introduced earlier, we define the following competing hypotheses:
$H_0 : |dF_1| = 0$ and $H_1 : |dF_1| > 0$. We further define the **level** of the test $\alpha = p(\text{Reject } H_0|H_0)$, where

α takes on values 0.01, 0.05 and 0.1. Table 3.3 summarizes the possible outcomes of a statistical test. With that respect, rejecting H_0 means that one is confident with $(1-\alpha)\cdot 100\%$ that H_0 is really false.

Table 3.3: Various outcomes of a statistical test, with $\alpha = p(\text{Type I error})$.

	Decision	
Truth	Reject H_0	Accept H_0
H_0	Type I error	OK
H_1	OK	Type II error

In order to assess the performance of the statistical tests on their Type I error, which refer to as the level also called **Size** of the test, and on their **Power** $= 1-$ Type II error, we used the following protocol.

For each category C_i, we sampled over D_{true}, S (500) evaluation sets D_{te}^s of N documents, ran the significance test over each D_{te}^s and computed the proportion of sets for which H_0 was rejected given that H_0 was true over D_{true} (*resp.* H_0 was false over D_{true}), which we note α_{true} (*resp.* π). Figure 3.8 describe graphically the protocol followed for each category C_i.

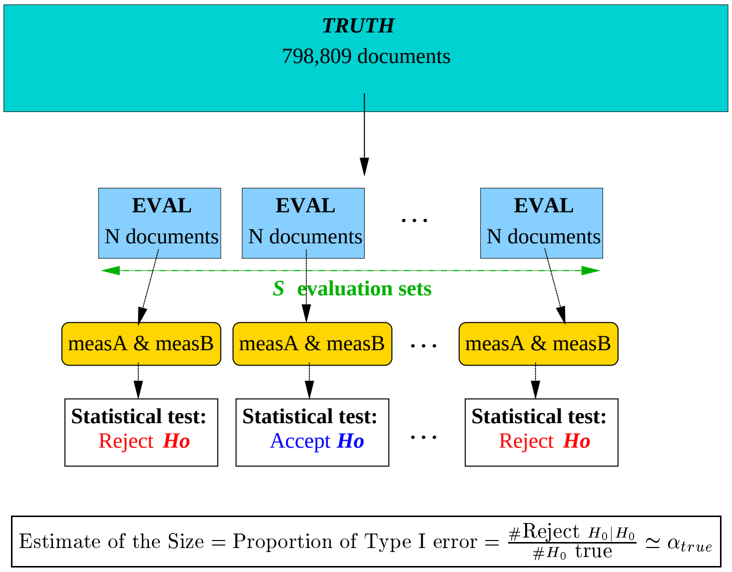


Figure 3.8: Protocol for the evaluation of significance tests.

We used α_{true} as an estimate of the significance test's probability of making a Type I error and π as an estimate of the significance test's Power. When α_{true} is higher than the α fixed by the statistical test, the test underestimates Type I error, which means we should not rely on its decision regarding the superiority of one model over the other. Thus, we consider that the significance test fails. On the contrary, $\alpha_{true} < \alpha$ yields a pessimistic statistical test that decides correctly H_0 more often than predicted.

Furthermore we would like to favor significance tests with a high π, since the Power of the test reflects its ability to reject H_0 when H_0 is false.

Summary of Conditions

In order to verify the sensitivity of the analyzed statistical tests to several conditions, we varied the following parameters:

- the value of α: it took on values in $\{0.1, 0.05, 0.01\}$;
- the two compared models: there were three models, two of them were of the same family (SVMs), hence optimizing the same criterion, while the third one was an MLP. Most of the times the two SVMs gave very similar results, (probably because the optimal capacity for this problem was near linear), while the MLP gave poorer results on average. The point here was to verify whether the test was sensitive to the *closeness* of the tested models (although a more formal definition of *closeness* should certainly be devised);
- the evaluation sample size: we varied it from small sizes (100) up to larger sizes (6000) to see the robustness of the statistical test to it;
- the class imbalance: out of the 101 categories of the problem, most of them resulted in highly unbalanced tasks, often with a ratio of 10 to 100 between the two classes. In order to experiment with more balanced tasks, we artificially created *meta-categories*, which were random aggregations of normal categories that tended to be more balanced;
- the tested measure: our initial interest was to directly test dF_1, the difference of F_1, but given poor initial results, we also decided to assess $dCerr$, the difference of classification errors, in order to see whether the tests were sensitive to the measure itself;
- the statistical test: on top of the bootstrap percentile test, we also analyzed the more classical *proportion test* and *McNemar test*, both of them only on $dCerr$ (since they were not adapted to dF_1).

Results

Figure 3.9 summarizes the results for the Size of the test estimates. All graphs show α_{true}, the number of times the test rejected H_0 while H_0 was true, for a fixed $\alpha = 0.05$, with respect to the sample size, for various statistical tests and tested measures.

Figure 3.10 shows the obtained results for the Power of the test estimates. The proportion of evaluation sets over which the significance test (with $\alpha = 0.05$) rejected H_0 when indeed H_0 was false, is plotted against the evaluation set size.

Figures 3.9(a) and 3.10(a) show the results for balanced data (where the positive and negative examples were approximatively equally present in the evaluation set) when comparing two different models (an SVM and an MLP).

Figures 3.9(b) and 3.10(b) show the results for unbalanced data when comparing two different models.

Figures 3.9(c) and 3.10(c) show the results for balanced data when comparing two similar models (a linear SVM and a Gaussian SVM) for balanced data, and finally Figures 3.9(d) and 3.10(d) show the results for unbalanced data and two similar models.

Note that each point in the graphs was computed over a different number of samples, since *eg* over the (500 evaluation sets $\times$ 101 categories) experiments only those for which H_0 was true in D_{true} were taken into account in the computation of α_{true}.

When the proportion of H_0 true in D_{true} equals 0 (*resp.* the proportion of H_0 false in D_{true} equals 0), α_{true} (*resp.* π) is set to -1. Hence, for instance the first points ($\{100, \ldots, 1000\}$) of Figures 3.10(c) and 3.10(d) were computed over only 500 evaluation sets on which respectively the same categorization task was performed. This makes these points unreliable. See in Appendix A.1, Figures A.2(c) and A.2(d) for more details.

For each of the Size's graphs, when the curves are over the 0.05 line, we can state that the statistical test is optimistic, while when it is below the line, the statistical test is pessimistic. As already explained, a pessimistic test should be favored whenever possible.

Several interesting conclusions can be drawn from the analysis of these graphs. First of all, as expected, most of the statistical tests are positively influenced by the size of the evaluation set, in the sense that their α_{true} value converges to α for large sample sizes[6].

On the available results, the McNemar test and the bootstrap test over $dCerr$ have a similar performance. They are always pessimistic even for small evaluation set sizes, and tend to the expected α values when the models compared on balanced tasks are dissimilar. They have also a similar performance in Power over all the different conditions, higher in general when comparing very different models.

When the compared models are similar, the bootstrap test over dF_1 has a pessimistic behavior even on quite small evaluation sets. However, when the models are really different the bootstrap test over dF_1 is on average always optimistic. Note nevertheless that most of the points in Figures 3.9(a) and 3.9(b) have a standard deviation *std*, over the categories, such that $\alpha_{true} - std < \alpha$ (see in Appendix A.2, Figures A.3(a) and A.3(b) Another interesting point is that in the available results for the Power, the dF_1's bootstrap test have relatively high values with respect to the other tests.

The proportion test have in general, on the available results, a more conservative behavior than the McNemar test and the $dCerr$ bootstrap test. It has more pessimistic results and less Power. It is too often prone to "Accept H_0", *ie* to conclude that the compared models have an equivalent performance, whether it is true or not. This results seem to be consistent with those of Dietterich (1998) and Nadeau and Bengio (2003). However, when comparing *close* models in a small unbalanced evaluation set (Figure 3.9(d)), this conservative behavior is not present.

To summarize the findings, the bootstrap-based statistical test over $dCerr$ obtained a good performance in Size comparable to the one of the McNemar test in all conditions. However both significance test performances in Power are low even for big evaluation sets in particular when the compared models are close. The bootstrap-based statistical test over dF_1 has higher Power than the other compared tests, however it must be emphasized that it is slightly over-optimistic in particular for small evaluation sets. Finally, when applying the proportion test over unbalanced data for *close* models we obtained an optimistic behavior, untypical of this usually conservative test.

[6]Note that the same is true for the variance of $\alpha_{true}(\rightarrow 0)$ (see Figure A.3, Appendix A.2), and this for any of the α values tested.

3.5 Conclusion

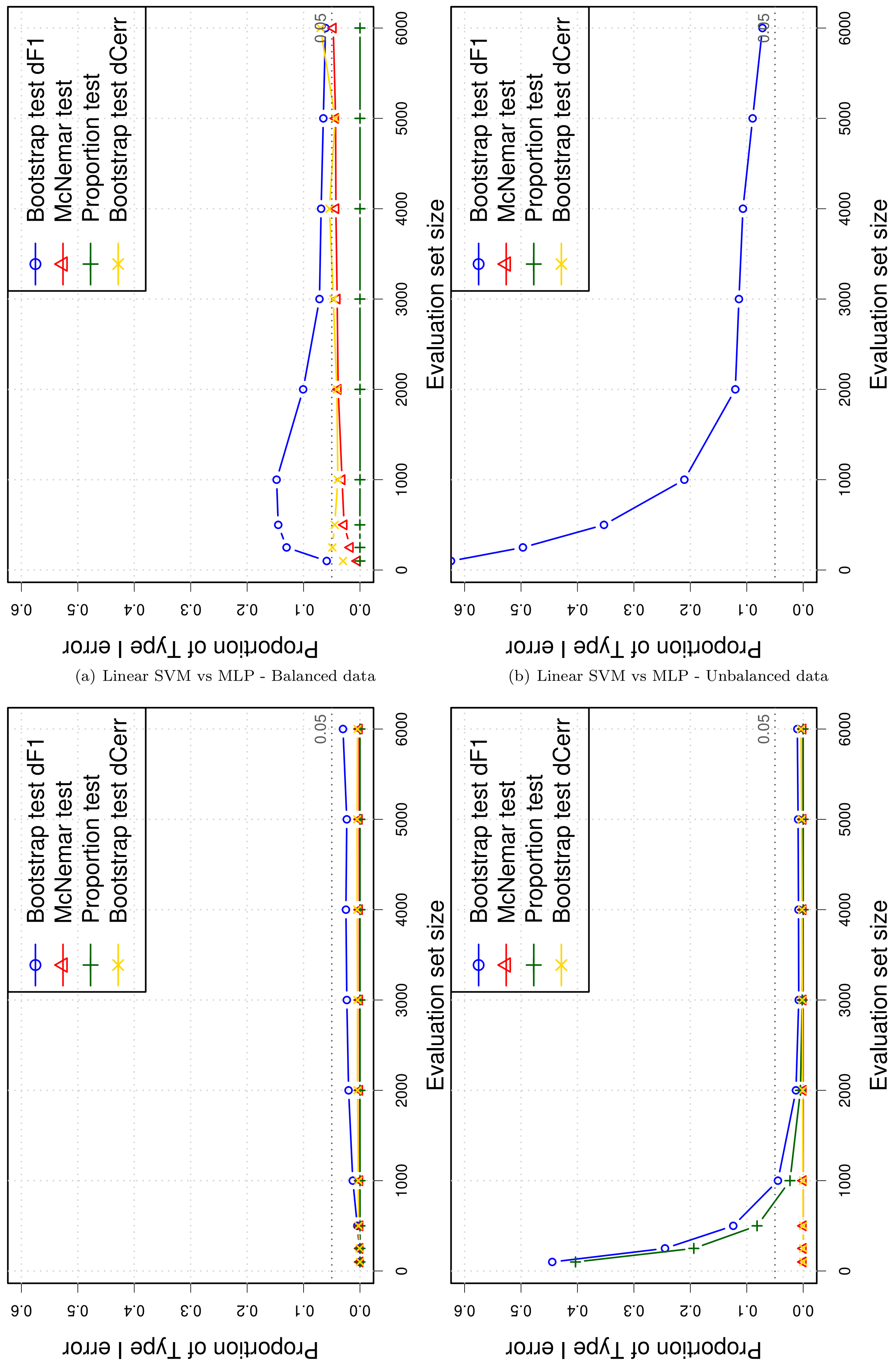
In this chapter we have described the databases on which the experiments of the thesis were conducted. We have defined the measures of performance commonly used to evaluate information retrieval systems. We reported the drawbacks that some of these measures and others similarly built from distinct domains have. We have offered an alternative which we claim may be closer to what we really want to measure. This discussion and the alternative solution were set out in the following publication:

> CONTRIB S. Bengio, J. Mariethoz, and M. Keller. The expected performance curves. In *Proceedings of the ICML 2005 wokshop on ROC Analysis in Machine Learning*, 2005

Finally, we have proposed an hypothesis test that verifies the statistical significance of the difference of two F_1 scores, a measure commonly used in information retrieval. The results of a benchmark conducted in various conditions were also reported here. They were published previously in:

> CONTRIB M. Keller, S. Bengio, and S.Y. Wong. Benchmarking non-parametric statistical tests. In *Advances in Neural Information Processing Systems, NIPS 18. MIT Press*, 2005

This chapter completes the presentation of the framework in which the subject of thesis is embedded. The next chapter proposes a first approach to the problem of text representation described in Section 2.3. This approach is directly inspired by the probabilistic models presented in Section 2.3.2.



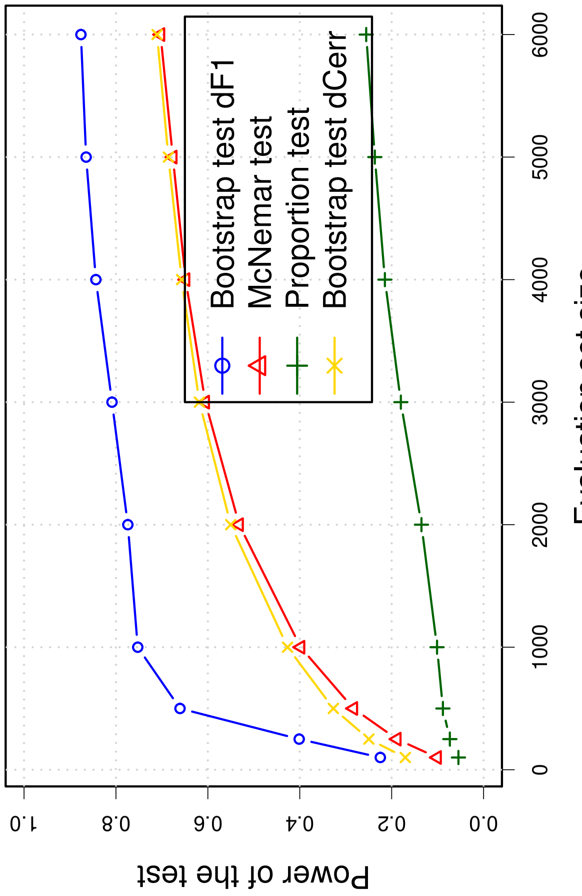
(a) Linear SVM vs MLP - Balanced data

(b) Linear SVM vs MLP - Unbalanced data

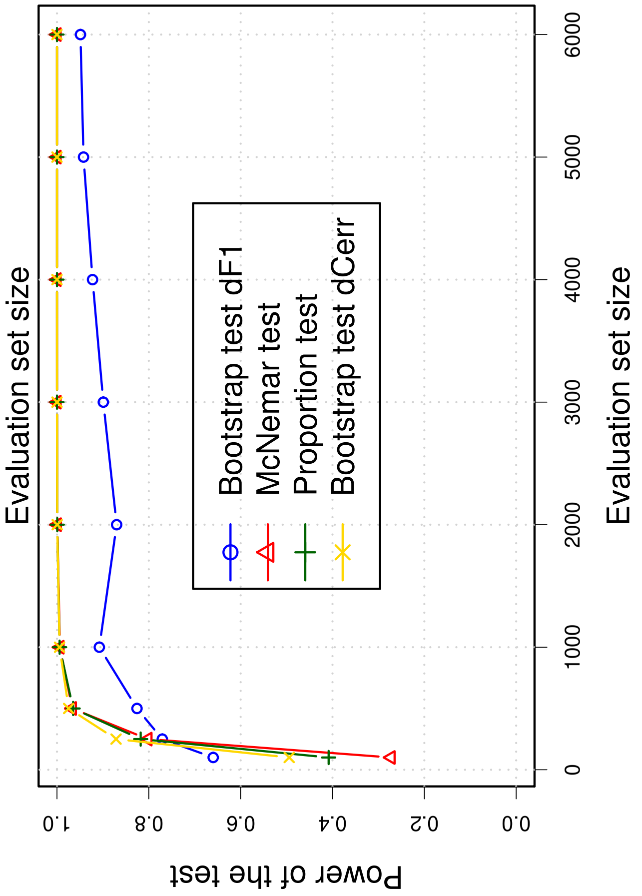
(c) Linear vs RBF SVMs - Balanced data

(d) Linear vs RBF SVMs - Unbalanced data

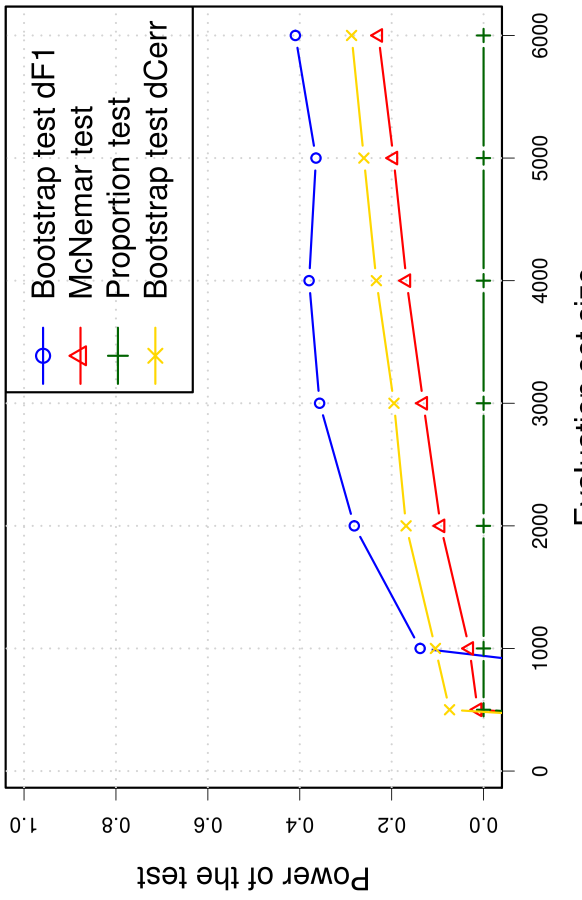
Figure 3.9: Several statistical tests comparing Linear SVM vs MLP or vs RBF SVM. The proportion of



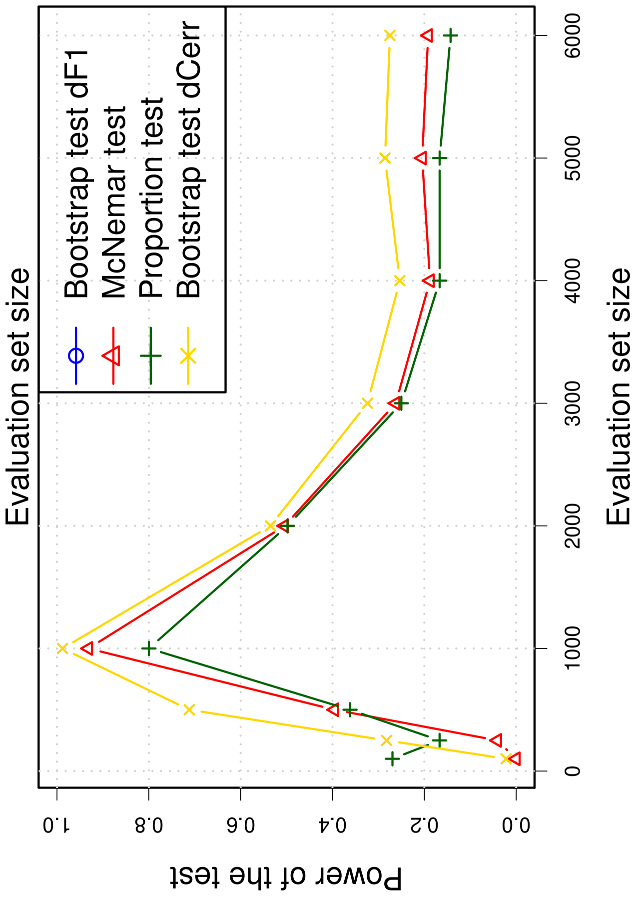
(a) Linear SVM vs MLP - Balanced data



(b) Linear SVM vs MLP - Unbalanced data



(c) Linear vs RBF SVMs - Balanced data



(d) Linear vs RBF SVMs - Unbalanced data

Figure 3.10: Power of several statistical tests comparing Linear SVM vs MLP or vs RBF SVM. The

Chapter 4

Theme Topic Mixture Model

As explained in Chapter 2, documents are often represented by a vector $d = (\alpha_1, ..., \alpha_M)$ of weights α_l, assigned to every word w_l in a vocabulary $\mathcal{V}$ of size M.

However, apart from the stemming step that may be performed during the dictionary extraction, there is no information about the semantic links between words included in this representation. Nevertheless, starting from this simple representation, several other document representations, described in Chapter 2, have been proposed in the literature trying to overcome some problems inherent to the bag-of-words representation. Some, like, the Latent Semantic Indexing (LSI), are based on an algebraic linear transformation of the term by document matrix. Others, based on probabilistic models, such as Latent Dirichlet Allocation (LDA) and Probabilistic Latent Semantic Analysis (PLSA) estimate the density of the documents given a particular model.

In the LSI approach a Singular Value Decomposition of the matrix X, whose columns are documents in the bag-of-words representation, is performed. Documents are then represented by their projections on the K first eigenvectors of X. Each eigenvector is a linear combination of the words' space basis' vectors. These combinations explain the name of "latent semantic" representation, since they tend to gather together words according to their co-occurrence rate. However since our aim is to represent the documents by a few components, highlighting some "concepts" present in the document rather than its words, shouldn't we learn that directly? PLSA and LDA are based on this observation. Indeed, as with LSI, these methods achieve a representation which is constructed to highlight a small number of "concepts" or "topics" present in the documents, instead of a huge number of words. Furthermore, gathering together the words in "concepts" is meant to disambiguate the cases of synonymic or polysemic use of language. However, as opposed to LSI these methods explicitly model latent entities that are meant to group words according to their belonging to this entity.

In this chapter, we present another probabilistic model, the Theme Topic Mixture Model (TTMM, Keller and Bengio, 2004b). This model, like LSI, the linear algebraic method, or like PLSA and LDA from the same family of models, tries to overcome the bag-of-words representation problems.

The chapter is organized as follows. In Section 4.1, TTMM is defined. In Section 4.2 a comparison between the probabilistic models PLSA, LDA and TTMM is conducted on several theoretical aspects.

Finally, Section 4.3 reports an experiment comparing different document representations.

4.1 Definition

The Theme Topic Mixture Model (TTMM), as mentioned in Section 2.3.2, shares its central idea with PLSA and LDA. The idea is to assume that given a hidden variable T which is called **Topic** the choice of words in the generation of a document is independent of the document itself. This idea defines a structure common to the three models, which Figure 2.8 illustrates, and which is reproduced for TTMM in Figure 4.1.

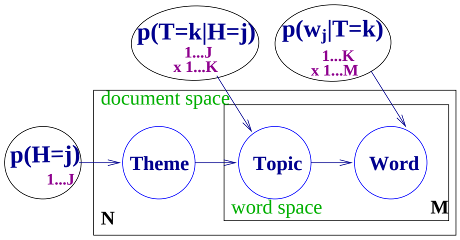


Figure 4.1: A graphical representation of TTMM

In TTMM, the variable H in the document space is called **Theme**. Each theme is characterized by a particular value for the mixing proportions over the topics. TTMM is very similar to LDA, but instead of using a continuous space for the choice of the mixing proportions of the topics, the choice is constrained to a discrete finite set. In this model the observed variable is the document d, seen as a set of words w_l, and the unobserved variables are the themes $H \in \{1, \ldots, J\}$ and the topics $T \in \{1, \ldots, K\}$, with J and K being hyper-parameters that must be chosen. Let us note Υ the set of model parameters that have to be estimated. This set Υ is composed of the tables representing the mixing proportions of themes $P(H = j)$, the mixing proportions of topics given the themes $P(T = k|H = j)$ and the probability of each word given each topic $P(w_l|T = k)$, for all $j \in \{1, \ldots, J\}$, $k \in \{1, \ldots, K\}$ and $l \in \{1, \ldots, M\}$.

A document under TTMM is seen as a sample from a mixture distribution over the themes which partition the space of document into J possibilities, the probability of an observed d can thus be decomposed as follows:

$$P(d) = \sum_{j=1}^{J} P(H = j)P(d|H = j).$$

As in LDA, once the theme has been chosen on the document space, the words composing a document are assumed to be drawn independently from a distribution which is specific to the theme, thus:

$$P(d) = \sum_{j=1}^{J} P(H = j) \prod_{w_l \in d} \left[P(w_l|H = j)\right]^{\mathrm{tf}_l(d)},$$

where $\mathrm{tf}_l(d)$ is the frequency of word w_j in documents d. Equation (2.15) from Section 2.3.2, encoding the principal assumption of this model, which is also common to PLSA and LDA, can be re-written as:

$$P(w_l|H=j) = \sum_{k=1}^{K} P(T=k|H=j)P(w_l|T=k),$$

for each word w_l and each theme $j \in \{1, \dots, J\}$. Under TTMM we can reformulate it saying that the words given a theme are sampled from a mixture distribution over the topics which mixing proportions depend on the theme. Finally, the probability of a document is given by:

$$P(d) = \sum_{j=1}^{J} P(H=j) \prod_{w_l \in d} \left[\sum_{k=1}^{K} P(T=k|H=j)P(w_l|T=k) \right]^{\mathrm{tf}_l(d)}. \tag{4.1}$$

Another way of understanding this model is by describing the underlying generative process which it assumes and that for each document is the following:

1. Choose $n(d) \sim Poisson(\xi)$: the document size.
2. Choose a theme $H = j$ from $P(h)$, a multinomial distribution representing the mixing proportions.
3. For each of the $n(d)$ words in d:
   (a) Choose a topic $T = k$ in $\{1, \dots, K\}$ from $P(t|H = j)$, a multinomial distribution conditioned on the theme $H = j$.
   (b) Choose a word w_l from $P(w|T = k)$, a multinomial distribution conditioned on the topic $T = k$.

The randomness of the document size $n(d)$, as for LDA, is necessary for the generative process, and for the same reason will be ignored (*cf* footnote 2 in Chapter 2).

Let D be a given corpus of N documents. The log-likelihood of the corpus D given the model then becomes:

$$\mathcal{L}(D|\Upsilon) = \sum_{i=1}^{N} \log \left[\sum_{j=1}^{J} P(H=j) \prod_{l=1}^{M} \left(\sum_{k=1}^{K} P(w_l|T=k)P(T=k|H=j) \right)^{\mathrm{tf}_l(d_i)} \right]. \tag{4.2}$$

The maximization of this log-likelihood can be done by EM (see complete derivations in Appendix B) as for PLSA or by Gradient Ascent Optimization (Keller and Bengio, 2004a).

In the **E-step** the posterior probabilities of the latent variables are estimated, as follows:

$$\begin{aligned} P_{ij} &= P(H=j|d_i) = \frac{P(H=j)P(d_i|H=j)}{\sum_{q=1}^{J} P(H=q)P(d_i|H=q)} \\ &= \frac{P(H=j)\prod_{l=1}^{M}\left[\sum_{k=1}^{K} P(T=k|H=j)P(w_l|T=k)\right]^{\mathrm{tf}_l(d_i)}}{\sum_{q=1}^{J} P(H=q)\prod_{l=1}^{M}\left[\sum_{k=1}^{K} P(T=k|H=q)P(w_l|T=k)\right]^{\mathrm{tf}_l(d_i)}} \end{aligned}$$

$$Q_{jkl} = P(T=k|w_l, H=j) = \frac{P(T=k|H=j)P(w_l|T=k)}{\sum_{p=1}^{K} P(T=p|H=j)P(w_l|T=p)}.$$

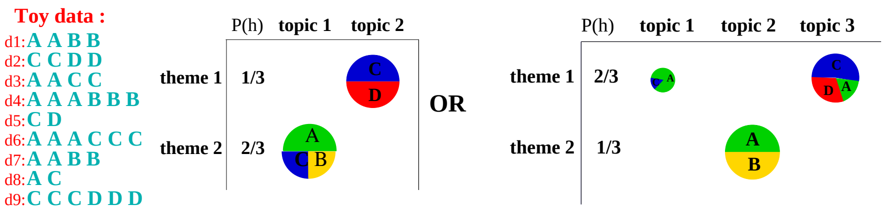
The **M-step** leads to an update of the model's parameters using the posterior estimates of the E-step:

$$P(H=j) = \frac{\sum_{i=1}^{N} P_{ij}}{N},$$

$$P(T=k|H=j) = \frac{\sum_{i=1}^{N} P_{ij} \sum_{w_l \in d_i} \mathrm{tf}_l(d_i) Q_{jkl}}{\sum_{p=1}^{K} \sum_{i=1}^{N} P_{ij} \sum_{w_l \in d_i} \mathrm{tf}_l(d_i) Q_{jpl}},$$

$$P(w_l|T=k) = \frac{\sum_{i=1}^{N} \sum_{j=1}^{J} \mathrm{tf}_l(d_i) P_{ij} Q_{jkl}}{\sum_{m=1}^{M} \sum_{i=1}^{N} \sum_{j=1}^{J} \mathrm{tf}_m(d_i) P_{ij} Q_{jkm}}.$$

To illustrate the behavior of TTMM, Figure 4.2 shows an example of two different TTMMs, with respective hyper-parameters $J=2$, $K=2$ and $J=2$, $K=3$, trained over a toy corpus of documents which vocabulary is composed of the letters A, B, C and D. The distributions $P(w_l|T=k)$ for each topic k are illustrated by pie charts, the size of the pie chart indicating the $P(T=k|H=j)$ for each theme j. For example, under the first model the document d_2 can be seen as being generated by the theme 1, by picking several times the topic 1. In the second model the probability of the theme 2 $P(h=2)$ reflects the proportion of documents of type AB ($\frac{3}{9}$) in the data.



$$[P(T=k|H=j)] = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \quad \text{OR} \quad [P(T=k|H=j)] = \begin{bmatrix} 2/5 & 0 & 3/5 \\ 0 & 1 & 0 \end{bmatrix}$$

Figure 4.2: Illustration of two different TTMMs learned over the Toy data with respective hyper-parameters $J=2$, $K=2$ and $J=2$, $K=3$. Each pie chart represents $P(w_l|T=k)$ and it size $P(T=k|H=j)$ for fixed j and k. Both models acknowledge the existence of the 3 kinds of documents AB, AC and CD in different proportions.

As for PLSA and LDA, this density estimation method can then be used as text representation. The idea is that instead of considering words as basic units of document representation we will consider a topic basis, with the hope that a few topics will capture more information than the huge amount of words. We can choose as topic component its posterior given the document, $P(T=k|d) = \frac{P(T=k,d)}{P(d)}$, where,

$$P(T=k,d) = \prod_{w_l \in d} [P(w_l|T=k)]^{\mathrm{tf}_l(d)} \sum_{j=1}^{J} P(H=j)P(T=k|H=j).$$

4.2 Comparison

In the following we discuss and compare the three models described in the previous subsections. First note that these models are not completely new from a statistical point of view. Similar modelisations can be found in others domains such as for example the Tied Mixture used in Speech Processing (Bellegarda and Nahamoo, 1989).

Let us take a closer look at their main equations, (2.16), (2.17), and (4.1), since they summarize quite well the similarities and dissimilarities between the underlying models.

$$\textbf{PLSA: } P(d_\delta, w_l) = P(\delta) \sum_{k=1}^{K} P(T=k|d_\delta) P(w_l|T=k). \tag{2.16}$$

$$\textbf{LDA: } P(d) = \int P(\tau) \prod_{l=1}^{M} \left[\sum_{k=1}^{K} P(w_l|T=k) P(T=k|\tau) \right]^{\mathrm{tf}_l(d)} d\tau. \tag{2.17}$$

$$\textbf{TTMM: } P(d) = \sum_{j=1}^{J} P(H=j) \prod_{l=1}^{M} \left[\sum_{k=1}^{K} P(w_l|T=k) P(T=k|H=j) \right]^{\mathrm{tf}_l(d)}. \tag{4.1}$$

As it has been mentioned before, and can be noticed in the three equations, there is a common core which is the mixture of word multinomial over the topics. However, TTMM is something like a hybrid between PLSA and LDA.

Even if PLSA is a word/document occurrence model and not a document model, we can say that PLSA would be a TTMM where the themes would have been identified with the documents. Furthermore, we can almost say that TTMM is a discretized LDA, or LDA a continuous TTMM. Indeed, where we have a discrete mixture of themes in TTMM, we have a continuous mixture of τ in LDA, both themes and τs defining the mixture proportions over the topics. Note however that by discretizing LDA in order to obtain TTMM, we trade something else apart from the continuity. Using a Dirichlet distribution for choosing the parameters of the multinomial over the topics constrains these parameters in a way that does not exist in TTMM. Indeed under the Dirichlet assumption, the parameter $\tau = (\tau_1, \ldots, \tau_K) = (P(T = 1|\tau), \ldots, P(T = K|\tau))$, is chosen with a probability:

$$P(\tau) = P(\tau|\zeta) = \frac{1}{\mathcal{Z}(\zeta)} \prod_{k=1}^{K} \tau_k^{\zeta_k - 1}$$

where $\mathcal{Z}(\zeta) = \frac{\prod_k \Gamma(\zeta_k)}{\Gamma(\sum_k \zeta_k)}$. The Dirichlet parameter ζ can be interpreted as being the *a priori* numbering of an event drawn from a multinomial distribution of parameter τ. Thus, given ζ some τ are more likely than others. Another way of seeing that is that where LDA has K degrees of freedom, in the parameter $\zeta \in \mathbb{R}^{+K}$, TTMM has at least $J \times K$ degrees of freedom, in the choice of the parameters of the J multinomial distributions $P(T|H = j)$ in the $(K-1)$-simplex. Less constraints or more degrees of freedom means more capacity, which as we explained in Chapter 2 may lead to over-fitting.

TTMM needs $J(1 + K) + KM$ parameters while LDA only needs $K + KM$. This is due to the fact that the continuous distribution with one parameter, which generates the mixing proportions τ in LDA, is replaced in TTMM by two discrete distributions, the multinomials representing $P(H)$ and $P(T|H = j)\forall j$. The number of parameters is possibly less than with PLSA (number of parameters: $KM + KN$), since we hope that documents can be clustered together by themes, and so $J < N$.

On the other hand, LDA optimization is intractable, so it has to be approximated. The variational EM algorithm has a complexity in time of $\mathcal{O}(NK\overline{n}[\overline{n} + M])$ at each step, where $\overline{n}$ is the mean of the documents' lengths. Furthermore, because of its structure, PLSA tends to over-fit, but training can be smoothed in order to reach and acceptable solution using Tempered EM. Each EM step for PLSA has a time complexity of $\mathcal{O}(NK\overline{n}[\overline{n} + M])$ like LDA. TTMM optimization can be reached with an exact inference but with a higher time complexity ($\mathcal{O}(NKJ[\overline{n} + M])$) than the two others, if the mean of documents' lengths is smaller than the number of themes J.

4.3 Experiment

In this section, an experiment comparing LDA, PLSA, TTMM, and the bag-of-words representation is reported. In Blei et al. (2003), LDA's features and bag-of-words document representations were compared on a Text Categorization task using support vector machines (SVMs) as classifiers. Using the same data (a subset of Reuters-21578, the database described in Chapter 3), splits and experimental protocol, the experiment is repeated here with TTMM and PLSA features.

The general procedure of the experiment is as follows:

1. A document density estimation model (LDA, PLSA or TTMM) is trained on a set of documents D.
2. The set D is split into a training set Tr_p containing a proportion p of the data, and a test set Te_p containing the remaining data.
3. An SVM is trained on Tr_p using for document representation the features extracted from the document density estimation model.
4. This SVM is tested on Te_p.
5. The steps 2., 3. and 4. are repeated for several splits and several values of p.
6. The experiment goes through steps 1. to 5. for each of the document density estimation models. The results obtained on the Te_p sets are compared to results obtained using SVMs trained on the bag-of-words representation of documents.

Note however that the results of this experiments are optimistic, and not comparable with other Text Categorization published results, since the vocabulary was extracted, and the models trained from the concatenation of Tr_p and Te_p sets. Thus the problem of having unseen words in the test set is not addressed. Nevertheless, in order to make a comparison between the three models, we followed the same experimental protocol as described in Blei et al. (2003).

We give here some details about the training of the models:

- Data: Blei et al. (2003) have selected 8529 Reuters-21578's documents (almost all the training data of ModApte split) for the set called D below.
- Bag-of-words: They stopped but did not stem the data, and from the resulting vocabulary they discarded the less frequent words to finally obtain a vocabulary of 15810 words.
- LDA: A model with 50 topics was trained on all the documents, without reference to their class labels.
- PLSA: A model with 50 topics was trained by simple EM optimization rather than by TEM. We make this choice for the sake of simplicity (fewer hyper-parameters to tune). However note that in this experiment letting PLSA over-fit should favor it instead of harming its performance. Indeed, in this experiment, there is no true test set since the features are learned over the whole set D. The value 50 for the number of topics has been chosen to match LDA's choice.
- TTMM: Models with 50 topics and several values for the number of themes have been trained by EM using early stopping to control the capacity. With 500 and 1000 themes we obtained the highest likelihood among the number of themes values we tried. The value 50 for the number of topics has been chosen to match LDA's choice.
- SVMs: For PLSA and TTMM features, linear and Gaussian kernels were tried. The choice of the Gaussian kernel standard deviation was made using K-Fold cross-validation ($K = 5$) on each of the splits[1]. The Gaussian kernels give the best results, which are the ones that we report on the graphics shown in Fig. 4.3 and 4.4.

As can be seen in Fig. 4.3 and 4.4 the results obtained with the features extracted from the document density estimation models are comparable.

We can see in this experiment that the document density estimation models do capture important information from the data, since even with 99.6% less features than the bag-of-words representation (50 vs 15810) the results are better for small values of p.

However, when the proportion of data used for training the SVMs is bigger, the results do not show a clear advantage of the models' features over the bag-of-words representation. This may be because there is not enough capacity, and the number of features is too small. This hyper-parameter (the number of latent topics), should probably be tuned according to a criterion depending on the number of labeled documents that we have for training the SVMs, and not only on the maximum likelihood of the parameters over D. Another explanation could be that all these density models rely on constraints which may be too strong when there is enough data.

Looking in more detail the results for each of the two categories GRAIN (Fig. 4.3) and EARN (Fig. 4.4), we notice that the three models do not perform equally. Indeed, TTMM give overall better results for GRAIN than for EARN, while PLSA has the converse behavior and LDA an overall good performance

[1]We have not been able to sort out, what kind of kernel was used in the SVM trained on LDA features in Blei et al. (2003), and have supposed thus in the analysis of the results that it may be sub-optimal. The same happens with the choice of the number of topics.

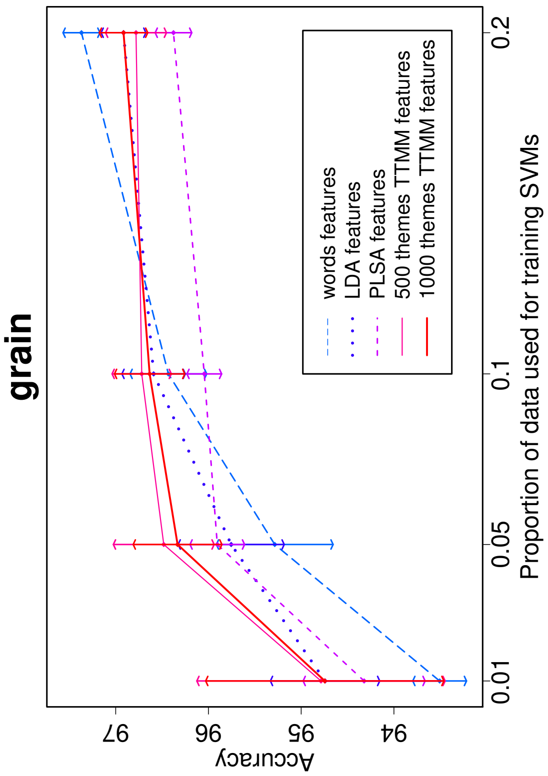


Figure 4.3: Classification results on GRAIN vs. NOT GRAIN binary classification problem for several proportions of training data, and several features.

in both cases. We can explain this by the fact that classifying document as belonging to category EARN or category GRAIN, are different kinds of tasks for which the three models are adapted to greater or lesser degree. Indeed, GRAIN is a category that is only represented in 5% of the data[2] while 35% of the documents are labeled EARN.

4.4 Conclusion

In this chapter, we proposed yet another model in the family of density estimation models for document representation, namely the Theme Topic Mixture Model (TTMM), which lies in between LDA and PLSA, sharing some advantages of both of them. A theoretical comparison between the models was then presented, highlighting advantages and problems of each method. This was followed by an empirical analysis, which shows that no single model is always better than the others, and that the ultimate choice may depend on the actual data configuration. Interestingly, all of the proposed density estimation models fail with respect to the simple bag-of-words representation when the size of the dataset becomes

[2] 5% of the document belongs to the category GRAIN means that the trivial rejector has already an accuracy of 95%...

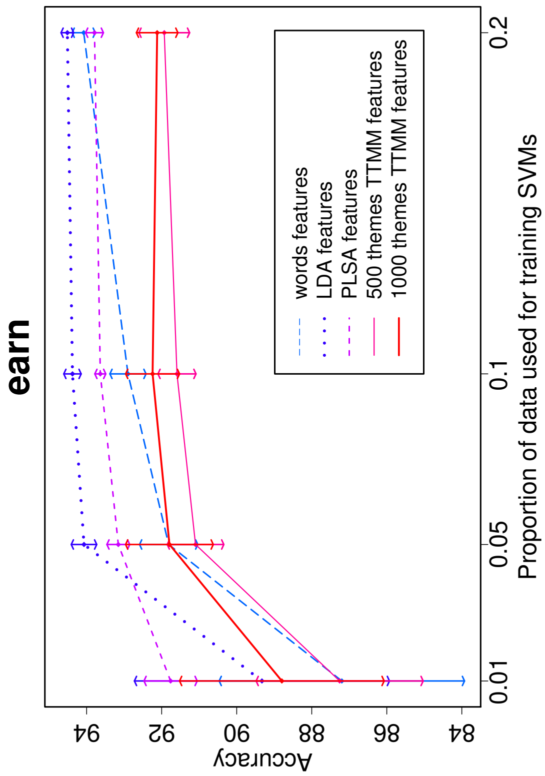


Figure 4.4: Classification results on EARN vs. NOT EARN binary classification problem for several proportions of training data, and several features.

sufficiently large. This probably means that the constraints that have been purposely integrated into all these models (the choice of words in a document is independent of the document itself given a hidden topic variable) may be useful when the data is rare but too strong when it is abundant, in which case constraints should be relaxed somehow. This study has been previously presented in

CONTRIB M. Keller and S. Bengio. TTMM: A graphical model for document representation. In *PASCAL Workshop on Text Mining and Understanding*, January 2004b

In the next two chapters, a distinct machine learning point of view is adopted to model words co-occurrences and document representation. Instead of a generative approach in which a global model of the word/document interaction is pursue, discriminative approaches are explored, in which the model is driven specifically toward the resolution of a task.

Chapter 5

A First Neural Network Approach to Text Representation

As seen so far, most recent research work on machine learning techniques for text representation have concentrated on novel probabilistic models. But is the probabilistic framework really necessary? In the resolution of a machine learning task, such a probabilistic framework appears necessary in two cases: either some probabilities are involved in the final decision, or probabilities are to be used as a tool for exploring the solution space. The tasks related to information retrieval do not necessarily belong to the first case; for example in a document retrieval task, what we seek is a ranking of RSV[1], for which a probabilistic setting is not particularly needed. Regarding the second case, as it has been suggested by LeCun and Huang (2005), while probabilities are a useful tool, they establish constraints, *eg* of normalization or cost function to be minimized, which are not always justified and that may be difficult to deal with. In addition, in a non-probabilistic framework, a lot of powerful tools allowing different kinds of exploration are available, among which the well-established margin and kernel concepts as well as the stochastic approximation.

The model proposed in this chapter intents to take advantage of the huge amount of unlabeled textual documents, using them as a clue *per se* to the links between words. The basic idea is to train a Neural Network using couples *(word, document)* as inputs and the absence or presence of the word in the document as targets.

A similar approach has been first proposed successfully in the context of statistical language modeling under the name of *Neural Probabilistic Language Model* (NPLM) (Bengio et al., 2003), which learns a distributed representation for each word alongside with the probability of word sequences in this representation. Here we develop a similar idea, which simultaneously learns distributed representations for words and documents and scores assessing the "appropriateness" of a word in the context defined by a document in these representations.

[1]Relevance Status Value, see Section 2.2.2

5.1 The Model

The model for text representation that we define in this section, relies on one main assumption. This assumption is close to the ones made by models such as PLSA and LDA presented in Chapter 2 and TTMM of Chapter 4. In all these models we assume that the probability that a given word appears in a given document depends somehow on the other words present in the document. As an illustration of this assumption, we have the intuition that the word *planet* is more likely to appear in conjunction with *astronomy*, *star*, *sun*, *galaxy*, than with *gene*, *DNA*, *cell*, *protein*. While PLSA, LDA and TTMM implement a generative approach, in which the existence of hidden variables is hypothesized in order to model the behavior of the document and word variables, in this chapter we described a model adopting an approach which could be considered as discriminative. Indeed we propose a model which is trained directly to answer the question of a word belonging to a given context.

Since the model presented in this chapter strongly relies on a particular kind of algorithm, the multi-layer perceptron, we begin this presentation by the definition of the latter.

5.1.1 Multi-Layer Perceptron

Multi-Layer Perceptrons (MLP) are among the most popular Artificial Neural Networks (ANN) algorithms. The first ANNs were proposed in the 1940's to model biological neural network but soon enough they were also used as learning algorithms to model a vast number of diverse problems. They are composed of simple processing units (or *neurons*) connected according to a particular pattern and which collectively exhibit a more complex behavior than each unit alone.

One of the first ANN proposed is the Perceptron (Rosenblatt, 1957). It is a linear classifier:

$$\begin{aligned} f_\omega : \quad \mathbb{R}^M &\to \mathbb{R} \\ x &\mapsto f_\omega(x) = a \cdot x + b, \end{aligned}$$

where $\omega = (a, b) \in \mathbb{R}^M \times \mathbb{R}$, the parameters, are estimated by the minimization of the number of misclassified examples, using a process similar to stochastic gradient descent. As any linear classifier, the perceptron has good performance when classes are linearly separable, but is not appropriate when this is not the case.

MLP trained using *error back-propagation* (LeCun, 1985) were introduced to overcome this lack of capacity. Figure 5.1, shows an illustration of a two-layer perceptron, which can be defined formally by a function

$$\begin{aligned} f_\omega : \quad \mathbb{R}^M &\to \mathbb{R}^K \\ x &\mapsto f_\omega(x) = \left[h^o(c_1 \cdot \phi(x) + d_1), \ldots, h^o(c_K \cdot \phi(x) + d_K)\right], \end{aligned} \tag{5.1}$$

where f_ω has K units in the output layer, each unit being composed of a perceptron passed through the *transfer* function h^o. For each $1 \leq k \leq K$, the parameters of the kth output unit $(c_k, d_k) \in \mathbb{R}^J \times \mathbb{R}$ combine the components of the *hidden layer* output $\phi(x) = (\phi_1(x), \ldots, \phi_J(x))^t$. The hidden layer is

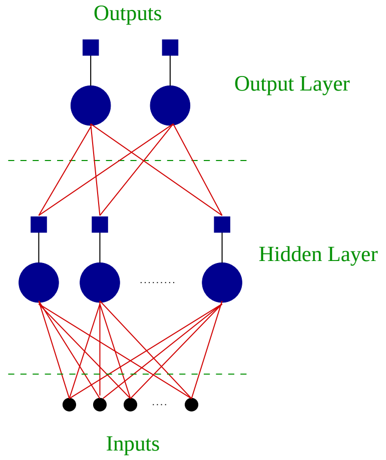


Figure 5.1: Multi-Layer Perceptron with 1 Hidden Layer. Blue disks represent perceptrons and blue squares transfer functions.

defined by J units. Each unit $\phi_j(x) = h^h(a_j \cdot x + b_j)$ is a perceptron, with parameters $(a_j, b_j) \in \mathbb{R}^M \times \mathbb{R}$, passed through a *transfer* function h^h, $1 \leq j \leq J$. Usual transfer functions, illustrated in Figure 5.2, include the identity $h(z) = z$ as well as non-linear transformations of the inputs such as the sigmoid function $h(z) = \frac{1}{1+e^{-z}}$ and the hyperbolic tangent $h(z) = \tanh(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$.

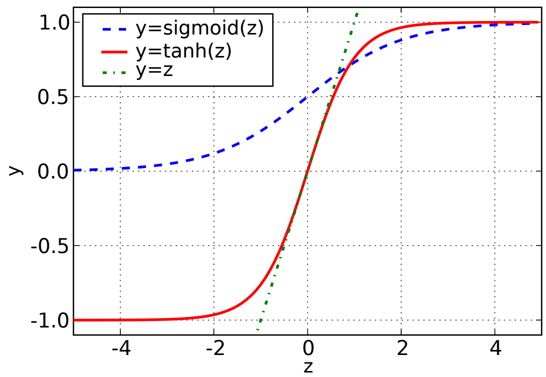


Figure 5.2: Examples of transfer function for Neural Network units.

In order to obtain a three-layer perceptron, another layer can be added on top of the output layer of the function defined by eq. (5.1). The previous output layer then becomes a second hidden layer which outputs are the inputs of each unit of the new layer. This may be repeated as many times as necessary in order to obtain an MLP with as many layers as desired. However, it has been shown by Hornik et al. (1989) that even with only one hidden layer multi-layer perceptrons are universal approximators (in the same sense as Fourier series are).

The parameters of the two-layer perceptron model defined in (5.1) are the weights

$$\omega = (c_1, \dots, c_K, d_1, \dots, d_K, a_1, \dots, a_J, b_1, \dots, b_J).$$

Note that the number of parameters, which depends on the number of layers, the number of units in each layer and the input dimension, is used to control the capacity of this class of models.

5.1.2 A Neural Network for Text Representation

The model defined in this section is a model of word/document co-occurrence, which we call for simplicity in the remainder of this chapter, the Neural Network for Text Representation (NNTR).

As illustrated in Figure 5.3, there are two input vectors in an NNTR: the first one is a word w_j represented by a one-hot encoding, and the second one is a document d_i represented as a bag-of-words with TFIDF weighting as defined in Section 2.2.1. The output is a score in $\mathbb{R}$ which target is high if w_j is *in the context of* d_i, and low otherwise. As depicted in Figure 5.3, the word (resp. document) vector is first passed through an MLP, referred to as MLP_W (resp. MLP_D) that extracts a richer and more distributed representation of words (resp. documents); these two representations are then concatenated and transformed non-linearly in order to obtain the target score using a third MLP, referred to as MLP_T, as summarized in equation (5.2):

$$f_\omega(w_j, d_i) = \text{NNTR}(w_j, d_i) = \text{MLP}_T \left\{ [\text{MLP}_W(w_j), \text{MLP}_D(d_j)] \right\} \ . \tag{5.2}$$

All the parameters of the model are trained jointly on a text corpus, assigning high scores to pairs (w_j, d_i) corresponding to documents d_i containing word w_j and low scores for all the other pairs. When the word w_j actually occurs in document d_i, we remove from d_i, in order to avoid a trivial over-fitting. We hope that instead the model will be able to generalize to words not present in the document but which are strongly related to its context.

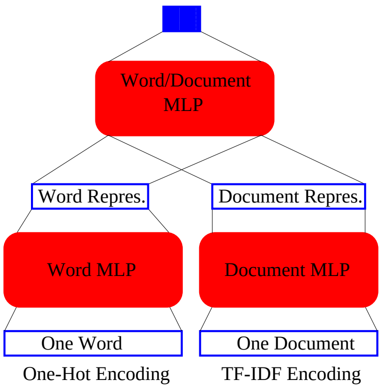


Figure 5.3: Illustration of the NNTR model

In order to train this neural network to perform this quite unusual task, we need to choose a learning criterion carefully. Indeed if we consider that the score output by NNTR for a word w and a document d is equal to a probability or if we consider it as an unnormalized score we should not choose the same criterion to train NNTR.

Let us consider the case where the output of NNTR is a probability and let us refer to this model as NNTR-P. As described in Bishop (1995), we can constrain this neural network to model the Bernoulli distribution $P(y = 1|x)$, with $x = (w, d)$ and $y = 1$ if the word w is in the context of the document d, $y = -1$ otherwise. By choosing the transfer function of MLP$_T$'s output layer to be a sigmoid we have implicitly that

$$P(y = 1|x) + P(y = -1|x) = 1,$$

for

$$P(y|x) = \frac{1}{1 + \exp(-yg_\omega(x))},$$

where g_ω corresponds to the output of NNTR before the transfer function. In order to train NNTR-P we minimize the negative log-likelihood of the correct class:

$$\mathcal{C} = -\ln\left[\prod_{i=1}^{N} P(y_i|x_i)\right] = -\sum_{i=1}^{N} \ln[P(y_i|x_i)] = \sum_{i=1}^{N} \ln[1 + \exp(-y_i g_\omega(x_i))]$$

However, doing so gives the same weight to each seen example. Note that our data presents a huge imbalance between the number of *positive* pairs and the number of *negative* pairs (each document only contains a fraction of words of the vocabulary). We had witnessed that if the errors which are back-propagate during the iterative training of the model by stochastic gradient descent, are mostly from one class, the model is quickly biased towards answering negatively and has then difficulties in learning anything else. Another kind of imbalance specific to our data is that, among the positive examples, a few words tend to appear really often, while a lot appear only in few documents, which would bias the model to give lower probabilities to pairs with infrequent words independently of the document.

In the case where the output of the neural network is an unnormalized score, which we refer in the following to as NNTR-W, we do not need to constrain its value to be in $[0, 1]$ and can choose any other criterion apart from maximizing the likelihood of the data. Thanks to this additional freedom, we can help the optimization process by weighting the training examples in order to balance the total number of positive examples (words w_j that are indeed in documents d_i) with the total number of negative examples. We can also balance each positive example independently of its *document frequency* (the number of documents into which the word appears). The criterion we thus optimize can be expressed as follows:

$$\mathcal{C} = \frac{1}{L^-}\sum_{l=1}^{L^-} Q(x_l, -1) + \frac{1}{M}\sum_{j=1}^{M}\frac{1}{\mathrm{df}_j}\sum_{i=1}^{\mathrm{df}_j} Q((d_i, w_j), 1), \tag{5.3}$$

where L^- is the number of negative examples, M the number of words in the dictionary extracted from the training set, df_j the number of documents the word w_j appears in, and $Q(x, y)$ the cost function for an example $x = (d, w)$ and its target y. Note that using this weighting technique we do not need to present the whole negative example set but a sub-sampling of it at each iteration, which makes stochastic gradient descent training much faster. We implemented an exploitation-exploration flavored stochastic gradient descent, which goes as follows. For each positive (*word*, *document*) couple's error back-propagation, we also back-propagate the outputs of a number of negative examples chosen randomly among (*word*, d') and (w',*document*) for exploitation, and (w',d') for exploration.

We choose as cost function the *hinge loss*:

$$Q(x, y) = max\left(0, 1 - y f_\omega(x)\right).$$

This choice was motivated by the fact that, as shown by Collobert and Bengio (2004), minimizing this cost function for an MLP is similar to maximizing a margin around its decision boundary.

5.2 Experiments

In this section we report three experiments on which we tested the NNTR model. First, we describe a preliminary experiment performed on artificial data that we designed to assess the importance of the criterion for the training of an NNTR model. In second place we compare several models including NNTR on a document retrieval task as defined in Section 2.2.2 using the TDT2 document corpus and corresponding TREC SDR queries described in Section 3.1. The third experiment is based on a lexical substitution problem, for which NNTR is able to provide scores.

5.2.1 Selecting the Training Method

Figure 5.4 shows the results of a preliminary experiment we did on a subset of the RCV1 corpus (see Section 3.1 for a description of RCV1), containing 660 training documents comprising a vocabulary of 5360 words (after stopping, stemming, and removing words appearing in only one document), as well as 660 other documents for test. The NNTR architecture was as follows: the word and document sub-MLPs had 5360 inputs each, no hidden units, and 10 outputs each; the joint word-document MLP had 20 inputs, 10 hidden units, and one output unit. We compared the two discussed training methods (NNTR-W used the weighted criterion with only 1% of the negative examples for training; NNTR-P used all training examples). We also compared these models with the PLSA model with 50 aspects. Note that the hyper-parameters of the models were tuned superficially using a validation set of another 660 RCV1 documents.

The models were compared as follows on the test data: taking advantage of the RCV1 segmentation of the documents into sentences, we decided to use sentences as "queries". Hence, for all 8454 sentences s of the test corpus, we computed $score(s, d)$ for all documents d (with the particular document d^s being the document originally containing sentence s, with s removed) as follows:

- for PLSA, we used a cosine similarity measure between s and d in the latent space, without combining it to the original cosine similarity score computed on the bag-of-words representation (as opposed to what is done by Hofmann (2001));

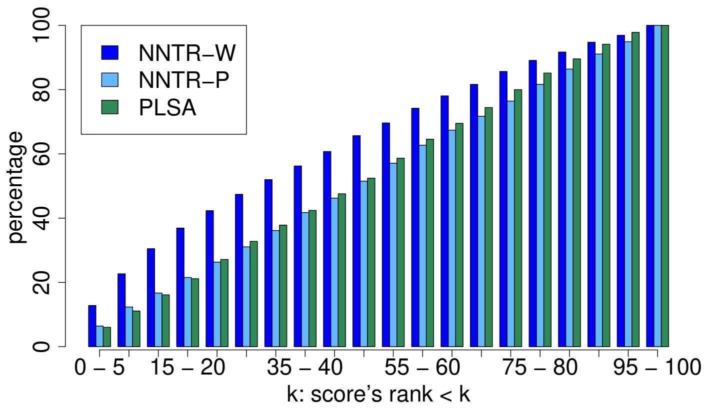


Figure 5.4: Cumulative distribution of the ranks.

- for NNTR, the score was the sum over words $w_i \in d$ of the outputs of NNTR(w_i, s) normalized by the document frequency of word w_i;
- we then computed the rank of $score(s, d^s)$ among the set of all $scores(s, d)$, and report the cumulative distribution of these ranks over all sentences in Figure 5.4. This gives an idea of the average proportion of documents one would have to consider, given a query, before obtaining the correct document. As an example, we can see in the figure that for the NNTR-W around 20% of the correct documents were ranked in the 10 first positions, against only 10% for NNTR-P.

As it can be seen, both probabilistic methods, NNTR-P and PLSA, performed similarly on the task, while NNTR-W yielded a significant improvement in the rank distribution (more queries found the correct corresponding document in the first ranks). In fact, we performed a McNemar test (see Section 3.4 for a description of this test) on each bar of the graph of Figure 5.4, which showed that, with 99% confidence, NNTR-W was statistically significantly better than PLSA for all but the last two bars of the graph (which are the least interesting).

Since NNTR-W achieves better performance than NNTR-P, in the remainder of this chapter all NNTR models described are NNTR-W models.

5.2.2 Document Retrieval

Using TDT2-clean, we trained a PLSA model with 1000 aspects and a NNTR with the following architecture: the word and document sub-MLPs had 25,438 inputs each (corresponding to the size of the training set vocabulary), no hidden unit, and 10 outputs each; the joint word-document MLP$_T$ had 20 inputs, 25 hidden units, and one output unit. The words of the dictionary input in MLP$_W$ are one-hot encoded and the documents input in MLP$_D$ are represented in the VSM with TFIDF weights normalized by the variance computed over the documents.

To tune the models' hyper-parameters we had previously divided the TDT2-clean into training (22,553 documents) and validation (2,246 documents) sets. PLSA was trained by EM optimization of the likelihood as explained in Section 2.3.2 over the training set for several number of EM iterations and several numbers of aspects. The optimal number of iterations was chosen as the one obtaining the maximum likelihood on the validation set for each number of aspects and the optimal number of aspects selected was the one giving the best results on the document retrieval task for the development set of queries TREC-8.

NNTR was trained by our exploration-exploitation version of stochastic gradient descent with early stopping regularization at a fixed number of iterations (900) for several values of the hyper-parameters. We chose over the validation set the combination of hyper-parameters minimizing the mean rank on the scores obtained with NNTR for each document in the validation set queried by itself. In other words, we computed for each document q in the validation set the following score:

$$score_{\mathrm{NNTR}}(q,d) = \sum_{w \in q} \frac{f_\omega(w,d)}{\mathrm{df}(w)} \tag{5.4}$$

for all documents d and found the rank of $score_{\mathrm{NNTR}}(q,q)$. The set of hyper-parameters for which NNTR found the minimal mean rank was selected. Several formulas were tested for computing the NNTR score, finally the one defined in equation (5.4), with a normalization by the document frequency of terms obtained the best results.

However, similarly to what Hofmann (2001) reported for PLSA, a score solely based on NNTR outputs does not yield good performance for the document retrieval task. For that reason the relevance of a document d to a query q was computed, as advocate by Hofmann (2001), like a combination between $score_{\mathrm{NNTR}}(q,d)$ and a score for the document given the query computed over the bag-of-words. Several combinations have been tried and their performance over the set of training queries TREC-8, reported on Figure 5.5. As scores computed over the bag-of-words representation we tested:

$$score_{TFIDF}(q,d) = \sum_{w \in q} \frac{\mathrm{tfidf}(w,d)}{\sum_{v \in d} \mathrm{tfidf}(v,d)}$$

which is similar to the score presented in Section 2.2.2, normalized by the sum of the TFIDF weights of the words in the document to remove the over scoring due to the length of the document. We also tested the classical cosine similarity measure, which is the normalized Euclidean scalar product of the query and document in the bag-of-words space:

$$score_{cosine}(q,d) = \frac{\sum_{j=1}^{M} \mathrm{tf}_j(d)\,\mathrm{tf}_j(q)}{\sqrt{\sum_{j=1}^{M} \mathrm{tf}_j(d)^2}\sqrt{\sum_{j=1}^{M} \mathrm{tf}_j(q)^2}}.$$

Finally the score for PLSA was computed as a cosine similarity measure in the latent space:

$$score_{PLSA}(q,d) = \frac{\sum_{k=1}^{K} P(T=k|d)P(T=k|q)}{\sqrt{\sum_{k=1}^{K} P(T=k|d)^2}\sqrt{\sum_{k=1}^{K} P(T=k|q)^2}}.$$

where K is the number of aspects of the PLSA model.

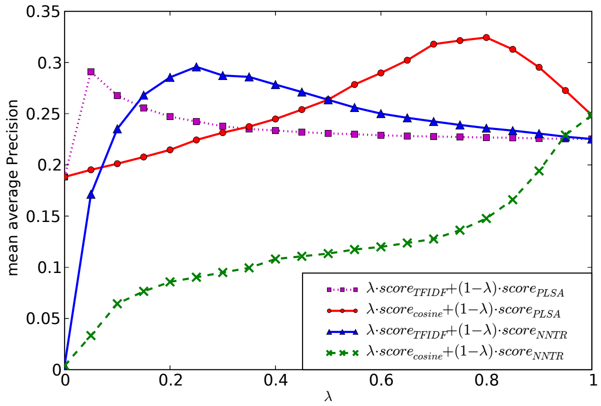


Figure 5.5: Combinations $\lambda \cdot score_{bow}(q,d) + (1-\lambda) \cdot score_{model}(q,d)$ for different scores and varying λ.

Figure 5.5 shows the mean average Precision (defined in Section 3.2) obtained over the TREC-8 set of queries for combinations of scores weighted, as follows, by a coefficient λ:

$$RSV_{model}(q,d;\lambda,bow) = \lambda \cdot score_{bow}(q,d) + (1-\lambda) \cdot score_{model}(q,d),$$

where $score_{bow}$ refers to scores computed over the bag-of-words representation and $score_{model}$ are the NNTR and PLSA scores. On the left side of Figure 5.5, when $\lambda = 0$, we can see the performance achieved by scores computed only with NNTR or PLSA models. On the right side of the figure, when $\lambda = 1$, we have the performance obtained for $score_{TFIDF}$ and $score_{cosine}$ not combined. As can be seen the mean average Precision obtained by using the output of NNTR alone is close to 0. PLSA alone exhibit already a performance almost as good as the bag-of-words scoring techniques. We notice that $score_{cosine}$ obtains a better performance than $score_{TFIDF}$. When combined with $score_{TFIDF}$ the scoring obtained with NNTR achieves a better performance than bag-of-words scoring techniques alone, and higher than that of the combination with $score_{cosine}$. The PLSA cosine similarity combined with $score_{cosine}$ obtained the overall better performance.

After choosing the best combination terms, that is RSV_{cosine}, $RSV_{NNTR}(\cdot,\cdot;0.25,TFIDF)$ and $RSV_{PLSA}(\cdot,\cdot;0.7,cosine)$ we computed them on each document in TDT2-clean for TREC-9 queries. We report in Figure 5.6 the Precision at k document for the 50 first positions in the ranking and in Table 5.1 the mean averaged precision of each model for TREC-9 queries. The differences between the models are significant with 99% confidence according to the Wilcoxon signed-ranked test (commonly used

Task		TFIDF	PLSA	NNTR
Document	λ	1	0.7	0.25
Retrieval	mAvP	0.230	**0.242**	0.210

Table 5.1: Compared mean Averaged Precisions (the higher the better) for the document retrieval task.

to assess the significance of mean average Precisions difference van Rijsbergen, 1975).

As can be seen in Table 5.1 both PLSA model and the cosine similarity measured over bag-of-words yields a higher mean average Precision than NNTR. However, in Figure 5.6 we can see that for the 50 first positions in the ranking the Precision of the three scoring techniques are quite close to each others.

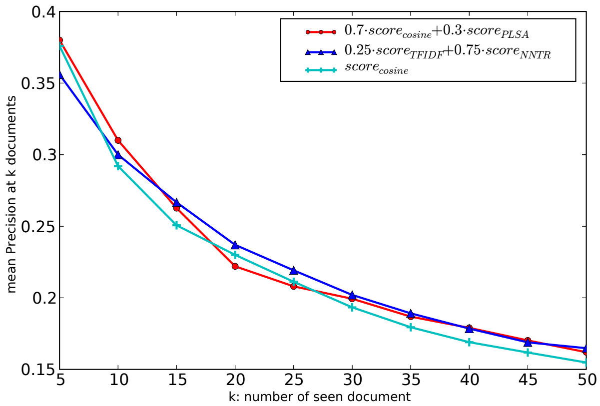


Figure 5.6: Mean over the queries of Precision at top for the document retrieval experiment

We can note that in the NNTR approach the query is not modeled. This could account for its modest result. In addition it is not directly trained to tackle a document retrieval task. However, being a discriminative approach, we can imagine that if being trained with a criterion designed for document retrieval its performance could improve.

5.2.3 Lexical Substitution Scoring

Lexical substitution is the task of replacing a word with another one that conveys the same meaning. In this section we report an experiment conducted in collaboration with two natural language processing groups (Bar-Ilan University, Jerusalem, Israel and CNTS, Antwerpen, Belgium) in the framework of a funding allocated by the PASCAL[2] network of excellence for the Textual Entailment pump priming project[3]. This experiment is focused on one particular step of lexical substitution applied to subtitle generation for video recordings of a source language speech. This experiment was previously reported in Glickman et al. (2006).

[2]PASCAL (Pattern Analysis, Statistical Modelling and Computational Learning) is the European Commission's IST-funded Network of Excellence for Multimodal Interfaces.

[3]`http://www.pascal-network.org/Reports/Pump%20Priming/893/`

As the subtitles for video recordings are restricted in length, it is necessary to compress the transcription of the original speech while maintaining the meaning of the original context. One technique employed for text compression is to replace words in the original transcription with shorter synonyms. In this experiment we are concerned with assigning a score for the correctness of such lexical substitution.

The data used in this experiment comes from BBC documentaries transcripts collected in the context of MUSA (Multilingual Subtitling of Multimedia Content) project (Piperidis et al., 2004). The dataset is composed of sentences $s = w_1 \dots w_i \dots w_n$ extracted from the transcripts and substitution examples. Substitution examples consist of a sentence s, a source word w_i in s and a target w' Wordnet synonym of w_i which fitness for substitution is judged by a human annotator. In total we have access to 918 substitution examples extracted from 231 sentences. In the experiments reported in the following, 31 of these sentences (with 121 corresponding examples) were used as a development set and 200 (corresponding to 797 substitution example) to evaluate and compare the models.

Four models are compared in this experiment. Two of them disregard the context in which the source word appears and used expert knowledge to decide the general likelihood of w' as a substitution to w_i.

- **semcor:** The score computed by this method is based on sense frequencies extracted from Semcor (Miller et al., 1993b), a corpus annotated with Wordnet synonyms. The score for w' given w_j is given by the proportion of occurrence of w_j annotated with w'.

- **sim:** The score given by this method comes from the Lin's dependency-based distributional similarity database[4]. In order to obtain this database, a similarity measure based on syntax (Lin, 1998) was applied to a large corpus (64 million words). As an example, two verbs in this database will be considered as similar if they share a large amount of subjects, objects, etc.

The other two, among which an NNTR model, try to score the fitness of the target word within its substituting context. In order to achieve this goal the two models were trained over the 4,357,500 (un-annotated) sentences of the BNC (British National Corpus, Burnard, 1995) in the bag-of-words representation (vocabulary size: 263,585 words). The hyper-parameter of the models were tuned on the development set mentioned previously.

- **bayes:** This method, proposed by Glickman et al. (2005) adopts a probabilistic naive bayes approach by computing

$$P(w'|C) = \frac{P(C|w')P(w')}{P(C|w')P(w') + P(C|^c w')P(^c w')} \sim \frac{P(w')\prod_{w \in C} P(w|w')}{P(w')\prod_{w \in C} P(w|w') + P(^c w')\prod_{w \in C} P(w|^c w')}$$

  where $C = w_1, \dots, w_{i-1}, w_{i-1}, \dots, w_n$ is the context of the substitution,

$$P(w|w') = \frac{|w \text{ appears in sentences containing } w'|}{|\text{words in sentences containing } w'|},$$

[4] available at http://www.cs.alberta.ca/~lindek/download.htm

is the probability that a word w appears in the context of a sentence containing w', and

$$P(w|^c w') = \frac{|w \text{ occurs in sentences not containing } w'|}{|\text{words in sentences not containing } w'|},$$

is the probability that a word w occurs in the a sentence not containing w'.

- **nntr:** An NNTR model was trained over the sentences of the BNC and its hyper-parameters were tuned over the development set of substitution examples by maximizing the average Precision. For simplicity MLP_W and MLP_D were restricted to be perceptrons. MLP_W had 20 outputs, MLP_D 100 outputs and MLP_T 500 hidden units.

The performances of these four models are shown in Figure 5.7. The models were compared according to two evaluation measures: the accuracy of taking as substitution for each sentence the target word getting the **best** score and the average Precision (defined in Section 3.2) obtained over all the scores. The random baseline and significance level of Figure 5.7 were obtained by running several times a system that gives random scores. In addition knowing that 25.5% of the original sentences in the evaluation set do not have a correct substitution and that for 15.5% of them all substitution are correct, the accuracy of the models must range inside $[0.155, 0.745]$, and a system that ranks all incorrect substitution above correct ones have an average Precision of 0.26.

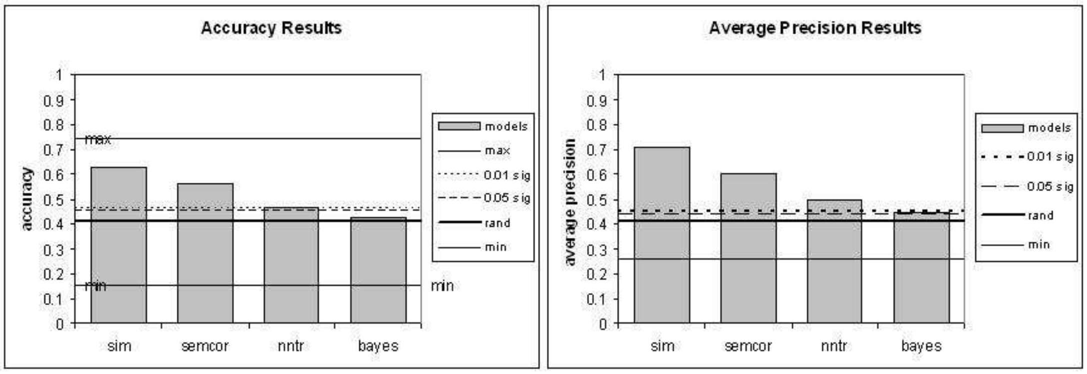


Figure 5.7: Accuracy and average precision for the four models compared in the lexical substitution scoring experiment.

As can be seen, context independent methods yields better results than the two contextual models. In particular the distributional similarity based method (sim) performs better than the others methods and the naive bayes (bayes) is not significantly better than random scoring. In defense of the contextual models, we may say that they are learned without any expert knowledge at hand, as opposed to the context independent methods. However, the bad performance of the NNTR model specially in terms of accuracy (which is important to this task), points out lacks that NNTR models may have. The sim method, even if it does not take into account the context of the substitution, has previously learned the similarity between words based on a *local* context encoded in the syntax. These kind of tasks, which measure the suitability of a word in a sentence, would highly rely on the local context. NNTR, on the contrary, because it is based on the bag-of-words representation, is based on the information contained in the global context of the sentence.

5.3 Conclusion

In this chapter, we have proposed a novel, non-probabilistic, text representation model. It estimates the suitability of a word in a document through the learning of a distributed representation for both words and documents. For this purpose it uses (*word*,*document*) examples labeled by the actual presence or absence of the word in the document. Despite the huge amount of data needed to learn this kind of models and the disproportion between the number of negative examples and positive ones, the algorithm behind the model scales. It is so, mainly because of the relative freedom gained from choosing a non-probabilistic model.

In this chapter we reported a preliminary artificial task and two real world text-related tasks, namely **document retrieval** and **lexical substitution scoring**, to which the model presented here have been applied. The experiments on real data have been previously presented in the following publications:

CONTRIB M. Keller and S. Bengio. A neural network for text representation. In *Proceedings of International Conference in Artificial Neural Networks (ICANN), Warsaw, Poland*, 2005

and

CONTRIB O. Glickman, I. Dagan, M. Keller, S. Bengio, and W. Daelemans. Investigating lexical substitution scoring for subtitle generation. In *Proceedings of CoNLL*, 2006

The experiment performed using artificial data which simulates a situation close to the document retrieval task, allowed us to select a training criterion for our neural network among a criterion mimicking a probabilistic behavior and another one which trades with un-normalized scores. This experiment gave us an insight that non-probabilistic approaches can yield interesting results.

From the real data document retrieval experiment we conclude that a non-probabilistic model such as NNTR can achieve performances comparable with a probabilistic model such as PLSA. However, we would like to point out that in NNTR the user query is not represented directly by the model which may be a problem when dealing with a document retrieval task.

In the second experiment on real data, NNTR performance was really modest as compared to a similarity measure between words taking into account the syntactic context in which they appear. Once again the model proposed by NNTR is too generic and does not consider during the training phase the specific constraints of the task to solve.

In the next chapter, a second neural network structure to text representation is presented. It relies in a Multi-Task approach on which the representation of the text is learned jointly with the actual targeted task.

Chapter 6

A Multi-Task Learning Approach to Text Representation

Most machine learning approaches to IR tasks propose to learn a useful representation from the basic bag-of-words, using either labeled documents (and a supervised learning approach) or unlabeled documents (and an unsupervised learning approach). While the supervised learning approaches are in general better to solve a given task, the small amount of available labeled data is often problematic; on the other hand, unsupervised approaches can use large amounts of data but the obtained representation is not specifically chosen to solve a real task, rather to optimize some form of data likelihood (which may or may not be appropriate for a given task (LeCun and Huang, 2005)).

In this chapter, we propose a method that would jointly make use of unlabeled and labeled documents in order to solve one or more supervised learning tasks.

Let us consider a raw unlabeled text document. If we split it into two parts we can reasonably assume that these two parts are related somehow to each other, otherwise the author of the document would not have put them together. This *a priori* expected relatedness between two parts of the same document could help us in finding what makes two documents related to each other, based on the words they contain. We would like to incorporate this information in the representation of documents, using a large unlabeled corpus of documents, while constraining the obtained representation to also solve (at least) one supervised IR task on some other, labeled, documents.

For that we have chosen to learn a mapping from the bag-of-words representation to a more compact and useful representation. We learn this mapping in a Multi-task learning framework, where one task relies on the labeled training data available for the IR task, while the second task is to learn a representation in which related documents are close to each other and unrelated documents are far from each other.

The chapter is organized as follows. In Section 6.1, we describe our proposed approach. Section 6.2 presents some experimental comparison between our approach and two competing approaches on a text categorization task over the Reuters RCV1 database. Finally, we conclude in Section 6.3.

6.1 Our Approach

As explained in the introduction, we would like to learn a mapping from a bag-of-words representation of documents to a richer and more informative representation. Let

$$\phi : \mathbb{R}^M \rightarrow \mathbb{R}^h,$$

$h < M$ be the mapping we are looking for. We would like that, in this new representation, topic related documents be more similar to each others than to documents discussing different subjects.

More formally, given a triplet (x, x^+, x^-) such that x is a document, x^+ is a document similar to x and x^- a document dissimilar to x, we would like the scalar product of the similar ones to be higher than that of the dissimilar ones:

$$\phi(x) \cdot \phi(x^+) > \phi(x) \cdot \phi(x^-). \tag{6.1}$$

Let us consider a large unlabeled corpus of documents $\mathcal{D}$, and in particular two documents d, $d' \in \mathcal{D}$. Let us divide d and d' in reasonably sized sub-parts, such as paragraphs. We stated in the introduction that we make the *a priori* assumption that any two paragraphs of d should by more related to each other than one paragraph of d and one of d'. We can then create a triplet (x, x^+, x^-) from $(d,\ d')$ by sampling any two paragraphs x, x^+ of d, and one paragraph x^- from d'. Let us call $\mathcal{D}_{unlab}$ a dataset of such triplets created from $\mathcal{D}$. The assumption used to create $\mathcal{D}_{unlab}$ can be considered as being quite simplistic. Indeed, two newstories covering the same topic may be closely related and in a long document the first and the last paragraph may be quite far from each others. However, our hope is that on a large corpus of relatively short documents this assumption is verified on average.

While $\mathcal{D}$ could be used to infer an interesting mapping ϕ, we would like this mapping to also be constrained to obtain good performance on some specific information retrieval task for which we have access to labels. We will concentrate in this chapter on a text categorization task, but the ideas presented here may also apply to other information retrieval tasks.

6.1.1 Implementation

The goal of text categorization as explained in Section 2.2.3 is to assign automatically categories, among a predefined set, to documents. In this framework, we have access to a set of labeled training documents $\mathcal{D}_{lab}$, in which each document d has an associated vector $y = (y_1, \cdots, y_K)$, $y_j \in \{-1, 1\}$ indicating the membership of d to category j, for each j among the K predefined categories. A usual supervised approach is to train a function $f(x) = (f_1(d), \cdots, f_K(d))$ in order to maximize the micro-averaged F_1 score at the breakeven point (see Section 3.2). Usually each function f_j is trained independently from the others, however in the following we choose to train them jointly.

Hence, we would like our mapping ϕ to be selected in order to perform well on two separate tasks, as

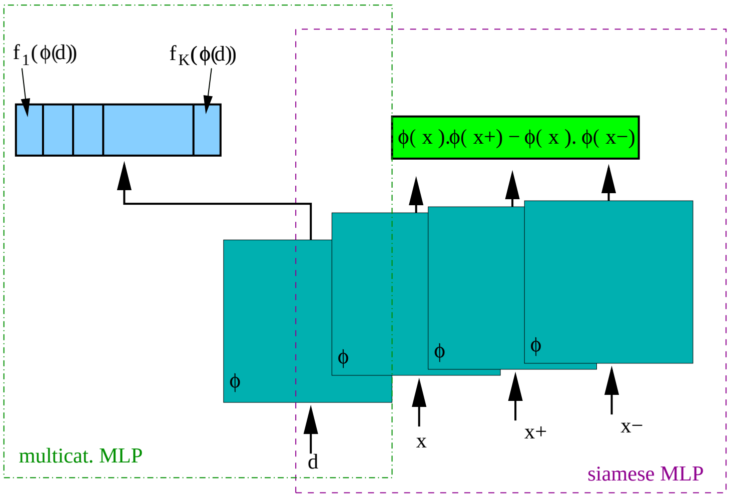


Figure 6.1: Neural Network Structure

illustrated in Figure 6.1: we want to learn jointly a function

$$\begin{array}{rccl} \psi_1: & \mathbb{R}^M & \to & \mathbb{R}^K \\ & d & \mapsto & \psi_1(d) = f(\phi(d)) \end{array}$$

using documents in $\mathcal{D}_{lab}$ and a function

$$\begin{array}{rccl} \psi_2: & \{\mathbb{R}^M\}^3 & \to & \mathbb{R} \\ & (x, x^+, x^-) & \mapsto & \psi_2(x, x^+, x^-) = \phi(x) \cdot \phi(x^+) - \phi(x) \cdot \phi(x^-) \end{array}$$

with triplets of paragraphs from $\mathcal{D}_{unlab}$. This is a *multi-task* framework (Caruana, 1997; Baxter, 2000) which can be reformulated by saying that we want to minimize jointly the expected risk of failing in one of the two tasks, hence to minimize the following risk for all documents:

$$R = \alpha_1 \int_{z=(d,y)} L_1(\psi_1(d), y) dz + \alpha_2 \int_{z=(x,x^+,x^-)} L_2(\psi_2(z)) dz, \tag{6.2}$$

where L_1 and L_2 are the losses associated with each task, while α_1 and α_2 represent the relative weights we put in each task. Indeed, in our case, learning a function to categorize texts is our priority, while learning a representation of documents through unlabeled data is an auxiliary task.

A function such as $\psi_2(x, x^+, x^-)$ can be learned using a model similar to the *Siamese* neural network proposed by Bromley et al. (1993) and more recently explored by Chopra et al. (2005). In our case ϕ is a Multi-Layer Perceptron (MLP, see description in Section 5.1.1), replicated three times for x, x^+ and x^-, and learned by the optimization of a ranking criterion with proximity constraints (Schultz and Joachims, 2003; Burges et al., 2005; Grangier and Bengio, 2005), defined as follows:

$$L_2(\psi_2(x, x^+, x^-)) = |1 - \left[\phi(x) \cdot \phi(x^+) - \phi(x) \cdot \phi(x^-)\right]|_+$$

where $|z|_+ = \max(0, z)$. We chose to infer $\psi_1(d)$ with a K-output MLP. As illustrated in Figure 6.1, the first layers of ψ_1 are a replication of those encoding the function ϕ used by $\psi_2(x, x^+, x^-)$. On top of the output layer of ϕ, the output layer of ψ_1 is added in order to solve the categorization problem on the characteristics space defined by the mapping ϕ. The parameters of $\psi_1(d)$ are trained by minimizing the following loss function:

$$L_1(\psi_1(d), y) = \frac{1}{K}\sum_{k=1}^{K} |1 - y_k \cdot [\psi_1(d)]_k|_+$$

where $[z]_k$ represents the zth component of vector z.

$\sum_{(d,y)} L_1(\psi_1(d), y)$ is an upper bound on the sum of the classification error over categories:

$$\mathcal{C} = \frac{1}{K}\sum_k \frac{N_{fp}^k + N_{fn}^k}{N},$$

using the notation defined in Section 3.2 for the counts on the possible outcomes of a classifier. Note that the loss function,

$$L_1'(\psi_1(x), y) = \frac{1}{K}\sum_{k=1}^{K} \frac{1}{N(y_k)} |1 - y_k \cdot [\psi_1(x)]_k|_+,$$

where $\frac{1}{N(y_k)}$ corresponds to the the proportion of positive (resp. negative) examples of category k for $y_k = 1$ (for $y_k = -1$), would result, when summed over the examples, in an upper bound on

$$\mathcal{C}' = \frac{1}{K}\sum_k \frac{1}{2}\left[\frac{N_{fp}^k}{N_{tn}^k + N_{fp}^k} + \frac{N_{fn}^k}{N_{tp}^k + N_{fn}^k}\right],$$

that is, the mean over the categories of the average of the errors on the positive and on the negative examples. When the data is balanced (proportion of positive examples $\approx$ proportion of negative examples), $\mathcal{C}' = \mathcal{C}$. However, in general, in text categorization problems some categories are much less populated than others and thus making the distinction may be interesting. In preliminary experiments where we considered choosing L_1' as a loss function we notice that its gives better results than L_1 in macro-averaged F_1, but worse results if considering the micro-averaged F_1 score. Based on that we choose to use L_1.

Thus, the empirical risk corresponding to (6.2) is:

$$\hat{R} = \alpha_1 \cdot \sum_{(d,y)\in\mathcal{D}_{lab}} L_1(\psi_1(d), y) + \alpha_2 \cdot \sum_{(x,x^+,x^-)\in\mathcal{D}_{unlab}} L_2(\psi_2(x, x^+, x^-)).$$

The whole model is implemented as a neural network that can be trained by stochastic gradient descent, using both labeled and unlabeled examples. In fact, instead of fixing α_1 and α_2, we alternatively select randomly l labeled documents from $\mathcal{D}_{lab}$ and u unlabeled triplets in $\mathcal{D}_{unlab}$, for which we train the corresponding parameters of the model, and repeat these random selections over several epoch on $\mathcal{D}_{lab}$. The values of α_1 and α_2 are then not easy to explicit. Their values must take into account the ratio of learning rates for ψ_1 and ψ_2, the ratio of the number of unlabeled *versus* labeled examples seen during training and the ratio of sparseness of the documents vs paragraphs. In the experiments reported in Section 6.2, a rough approximation is that we give three times more importance to the text categorization task. Furthermore, as we want to bias the model towards better solving the supervised task, we select the various hyper-parameters of ϕ on a validation set composed of only labeled documents.

6.1.2 Related Work

Probably one of the works most related to our is the Structural Learning framework presented by Ando and Zhang (2005). The idea of the Structural Learning framework, very close to the multi-tasks framework, is to solve several related tasks simultaneously in order to improve the learning of the structure that their solution may share. To encode this assumption the authors propose to model the predictors of their tasks as follow: $f_\Theta(x; w, v) =< [w^T, v^T], [x, \Theta x] >$, w and v being task specific, while the matrix Θ is common to all tasks. Experiments with tasks involving auxiliary unlabeled data have been conducted, leading to significant improvement over the single task performance. While giving a nice generic framework, this approach does not tackle the problem of learning a document representation directly and thus cannot target desired properties of this representation such as similarity between documents.

Others approaches using jointly labeled and unlabeled data for solving information retrieval task have been proposed (for example Joachims, 1999, 2003; Chapelle and Zien, 2005). They belong to the transductive learning paradigm in which the test set without annotation is available during training or to the semi-supervised learning scheme, close to our setting, in which unlabeled data issued from the same distribution as the training set can be used to better model the input space. However, none of these approaches rely on several tasks sharing their input space to complement each others.

6.2 Experiments

In this section, experiments comparing our approach to two other methods on the RCV1 (Reuters Corpus Volume 1) database (see Section 3.1 for a description of the database) are reported. We have chosen to compare the multi-task learning algorithm to its single task counterpart (using only the labeled data), considering the latter as a baseline to see if the unsupervised data helps to better solve the supervised task. Furthermore, in order to measure the usefulness of the multi-task setup we compare it to a successive use of the unlabeled data for learning ψ_2, and thus ϕ, followed by the training of a classifier on the input space defined by ϕ. Details of the implementation of the compared models are as follows:

- **Baseline:** An MLP with M inputs, 2 hidden layers of respectively h_1 and h_2 units and K outputs, M being the size of the vocabulary, and K the number of categories. h_1 and h_2 are chosen on a validation set among $\{50, 100, 200, 300, 500\}$. We refer to this model as a *multicat MLP*.

- **Two-Task:** A siamese MLP with 1 hidden layer learned jointly with a multicat MLP. The mapping ϕ, common to the Siamese MLP and the multicat MLP is composed of the input layer, the first hidden layer of the multicat MLP and uses the second hidden layer as output layer. h_1 and h_2 are kept as optimally chosen for the baseline.

- **Unsupervised:** The siamese MPL, ψ_2, has the structure described for the two-task model. However, the number of units of the hidden and output layers of ϕ, that is h_1 and h_2, are chosen on a validation set of triplets among $\{50, 100, 200, 500, 1000\}$ by the minimization of the L_2 loss function over the validation examples. Once ϕ is trained a perceptron with h_2 inputs and K outputs is trained using L_2 loss function on the training data transformed by ϕ.

These neural networks are trained by gradient descent optimization, with early stopping on a number of iterations tuned on the labeled validation set for the baseline, the two-task approach and the unsupervised approach perceptron, and unlabeled validation set for the unsupervised approach siamese MLP.

The RCV1 database, as explained in Section 3.1, is a corpus of $806,791$ news stories written over one year and labeled. There is a total of 101 categories and each document is labeled with one or more of these categories. For the experiments presented in the following, we have preserved the $6,945$ documents of the 4 last days as a test set, which we will note $\mathcal{D}_{test}$. From the remaining $799,846$ documents, noted $\mathcal{D}_{dev}$, we have sampled randomly sets of several sizes to simulate various $\mathcal{D}_{lab}$ and $\mathcal{D}_{unlab}$.

Each point in Figures 6.2, 6.3 and 6.4 simulates a scenario where we would be given a little set $\mathcal{D}_{lab}$ of labeled documents, and a big set $\mathcal{D}$ of unlabeled documents from which we extracted a set $\mathcal{D}_{unlab}$ of triplets. We used the biggest of the two sets, that is $\mathcal{D}$ to extract the dictionary. This dictionary covers almost entirely $\mathcal{D}_{lab}$ one's[1].

As can be seen in Table 6.1 the number M of words in the dictionary in our experiments is contained between 25K and 113K depending on the size of $\mathcal{D}$. It means that our MLPs have an input layer with a number of dimensions contained between 25K and 113K. However, given that the bag-of-words representation is sparse, only a few of these dimensions are active each time an example is presented to the network and thus the propagation of the gradient, during the training, is also sparse. This allows our networks to scale to relatively huge amount of data despite the high dimension of their input layers.

6.2.1 Varying the Labeled Set Size

In this first experiment, we want to compare the performance of the two-task approach against the baseline when we increase the size of the labeled set. We have first sampled from $\mathcal{D}_{dev}$ a set $\mathcal{D}$ of 10^4 documents, from which we have extracted a set $\mathcal{D}_{unlab}$ of $116,032$ unlabeled triplets. After stopping and stemming the words present in $\mathcal{D}$, as explained in Section 2.2.1, we obtained a vocabulary of $M = 25,138$ words. For each $s \in \{100, 200, 500, 1000\}$, we have then sampled a training set and a validation set of size s. The training set together with its corresponding validation set form a set $\mathcal{D}^s_{lab}$ of $2 \times s$ labeled documents. The documents in $\mathcal{D}^s_{lab}$ are transformed in their bag-of-words representation with TFIDF

[1] *eg* a dictionary extracted from a $\mathcal{D}$ of 10^4 documents covers 94% of the words of the vocabulary of a $\mathcal{D}_{lab}$.

weights using the vocabulary extracted from $\mathcal{D}$. The documents in $\mathcal{D}^s_{lab}$ were used to train the baseline model and, with the help of $\mathcal{D}_{unlab}$, the two-task model. The selection of the labeled training set for each s was repeated 5 times in order to obtain an estimate of the variance in the performance due to the choice of the training set. The mean and standard deviation of the F_1 score at the break-even point (BEP) over the $\mathcal{D}^s_{lab}$s for each size s and each model, are reported in Figure 6.2 and analysis of the results is provided in Section 6.2.4.

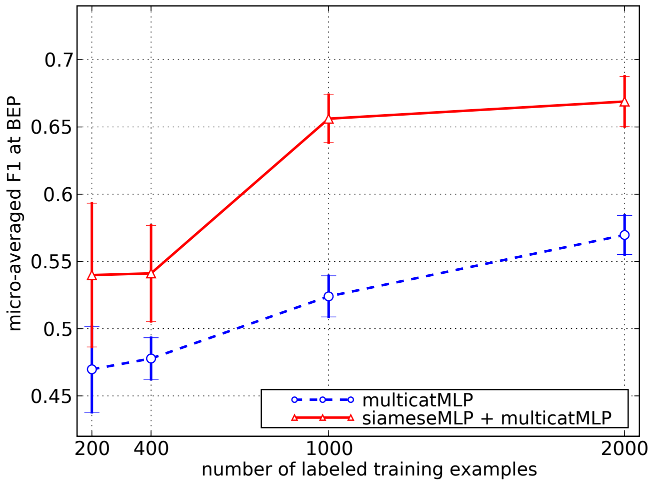


Figure 6.2: F_1 at BEP versus the size of the labeled training set. Each point and bar represent the mean and standard deviation of the performance obtained over the test set for 5 models, each trained on a different training set of the same size.

6.2.2 Varying the Unlabeled Set Size

By this experiment, we want to measure the performance of the two-task approach when the number of unlabeled examples is increased. We have sampled a set $\mathcal{D}_{lab}$ of 1,000 labeled documents. For each $s \in \{10^4, 5\times10^4, 10^5, 2\times10^5\}$ we have sampled a set $\mathcal{D}^s$ of s documents. After stopping and stemming the words in each $\mathcal{D}^s$ we have extracted vocabularies of increasing size and unlabeled sets $\mathcal{D}^s_{unlab}$ of triplets as reported in Table 6.1. The two-task models were trained on $\mathcal{D}_{lab}$ and $\mathcal{D}^s_{unlab}$, in order to see the effect of increasing the unlabeled material. Note however, that the hyper-parameters of the two-task models were kept as optimally chosen for the baseline model. The corresponding results are reported in Figure 6.3 with more analysis given in Section 6.2.4.

6.2.3 Verifying the Usefulness of the Multi-Task Framework

This experiment is a kind of sanity check, in order to see if the multi-task setup was really necessary or whether we could have learned separately the document representation on unlabeled data and then train

	Number s of documents in $\mathcal{D}^s$			
	10^4	5×10^4	10^5	2×10^5
M	$25,138$	$57,649$	$81,603$	$113,621$
$\lvert\mathcal{D}^s_{unlab}\rvert$	$116,032$	$590,043$	$1,179,164$	$2,343,769$

Table 6.1: Number of words in the vocabularies and number of triplets extracted from $\mathcal{D}^s$.

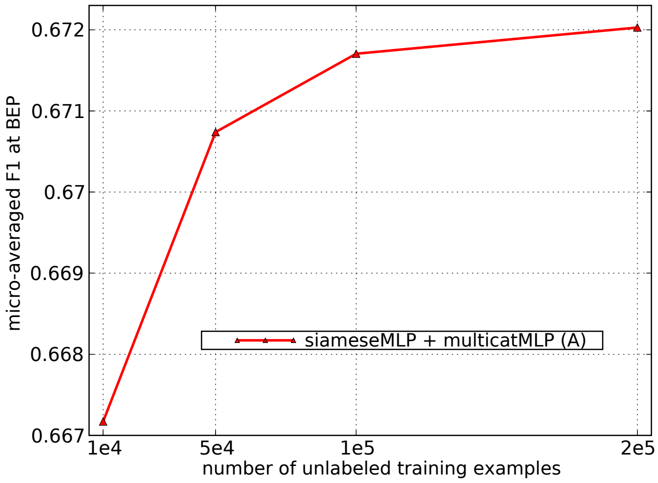


Figure 6.3: F_1 at BEP versus the size of the unlabeled training set for a fixed size of labeled training set (1000 documents).

a K outputs perceptron on the labeled set to obtain similar results. This experiment has been conducted using the same setup as described in experiment 6.2.2. The function $\psi_2(.)$ of Section 6.1 is trained alone on $\mathcal{D}^s_{unlab}$ with hyper-parameters tuned on a validation set of triplets extracted from 100 documents. The resulting ϕ mapping is used to project the documents $d \in \mathcal{D}_{lab}$ into a new representation, and the function $f(.)$ of Section 6.1 is then trained on this transformed data. Figure 6.4 compares the various settings.

6.2.4 Analysis

In Figure 6.2 the performance of the baseline is compared for an increasing number of labeled examples to that of the two-task approach using a fixed amount of unlabeled data. We can infer from experiment 6.2.1 (Fig. 6.2) that the use of unlabeled data in the two-task model provides a considerable improvement with respect to the baseline performance. This improvement is emphasized by the results of experiment 6.2.3 (Fig. 6.4), which shows the performance of the baseline, the two-task and the unsupervised approaches for a fixed amount of labeled data and increasing number of unlabeled data. We can see that if we learn the document representation mapping as a separate task and then learn a categorization model over this new representation, performance improves when the unlabeled data increases, but it remains significantly

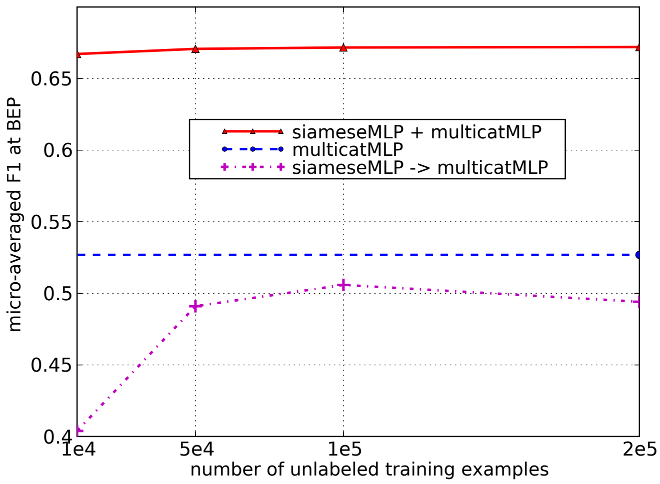


Figure 6.4: F_1 at BEP versus the size of the unlabeled training set for a fixed size of labeled training set (1000 documents). Compared models: baseline, two-task and successive learning of the mapping and the classifier.

lower than with the two-task approach. It is even lower than the baseline performance, which can be linked to the conclusion drawn from Chapter 5, of a learned mapping ϕ too uncorrelated to the task we want to solve.

Experiment 6.2.2, shows the performance of the two-task approach for a fixed number of labeled examples and increasing unlabeled material. From this experiment we can deduce that the increase of unlabeled material yields an improvement of the two-task models performance, opening the possibility of even better performance with very large unlabeled datasets. However, note that support vector machines, which as mentioned in Section 2.2.3, are the state-of-the-art approach, without making use of the unlabeled data, still yields a significant better performance than all the models considered here. As an example the micro-averaged F_1 score of K linear SVMs trained on a training set $\mathcal{D}_{lab}$ of 500 examples, with the hyper-parameters tuned on a validation set of another 500 examples is ~ 0.68 (against ~ 0.65 for the two-task approach as can be seen in Fig. 6.2). We think of several facts that may explain this under-performance of the two-task approach as designed in these experiments. First, out of a comparison with the baseline, the hyper-parameters of the two-task model must be tuned. A second fact, is that the three-layer perceptron used as baseline, and which has been carefully tuned, yields poor results as compared to SVMs (~ 0.52 micro-averaged F_1 score). This could suggest that constraining several categorization problems to share the representation $\phi(d)$ of a document is not a good idea. It would be interesting to compare the performance of two-task approaches when the labeled task is a single categorization problem.

6.3 Conclusion

Many information retrieval tasks are based on the careful selection (or estimation) of an appropriate representation space for documents. Estimating this representation through supervised learning approaches is limited by the scarce amount of available labeled documents. On the other hand, the use of (more easily available) unlabeled documents often leads to optimize a generic criterion, such as data likelihood, which may not necessarily be related to the information retrieval task. In this chapter, we have proposed a model, based on the Siamese neural network and the *a priori* knowledge that similarity among documents should play an important role in the definition of the representation space. This model learns a representation space using simultaneously labeled and unlabeled documents by the optimization of a multi-task criterion. We have shown empirically, on the Reuters RCV1 database, that increasing the amount of unlabeled documents during training yielded better performance on a separate, supervised, task. These preliminary results need to be confirmed on other, more standard databases, such as Reuters 21578, and compared against other unsupervised and semi-supervised approaches. This work have been previously reported in

> CONTRIB M. Keller and S. Bengio. A multi-task learning approach to document representation using unlabeled data. IDIAP-RR 36, IDIAP, 2006

Chapter 7

Conclusion

We began this dissertation by the presentation of the machine learning framework and the information retrieval field in which several approaches, that we described, have been proposed for tackling the problem of document representation. We also described in details several performance measures used in the information retrieval field and proposed an alternative to some of them arguing that the latter may be closer to the expected performance. In addition we reported a benchmarking experiment done for the purpose evaluating the behavior of several statistical significance tests.

We then presented several machine learning approaches to the problem of document representation taking advantage of unlabeled data.

As a first approach to better represent text documents, we presented a probabilistic model. We compared this model on several points to two other density estimation models for document representation. From a theoretical standpoint, we showed that our shares advantages and disadvantages with the two other approaches. In addition, the three models obtained comparable empirical performance on a text categorization task with an increasing number of labeled material. We noticed that when the size of the labeled data reaches a certain value, the document representation learning process based on unlabeled data yielded poorer results than the bag-of-words representation. This may be caused by the fact that the *a priori* information gained from the models may be interesting when the labeled data is scarce, but too constrained when more data is available. Another explanation can be that in this experiment the amount of unlabeled data used to train the models is quite small, and thus the extra information that the models are supposed to provide to the document representation is not very useful. However, it is not clear whether these models are able to scale to much higher amounts of data.

Being concerned by the scaling properties of a potential model aimed at learning the document representation from huge amounts of data, we then presented a first non-probabilistic approach to tackle the text representation problem. The proposed model is based on an artificial neural network approach which, instead of tackling the problem of document representation in a generative framework, rather explores a more discriminative direction. Stochastic gradient descent is used to train the neural network, allowing the processing of substantially more unlabeled data. This method was tested on several tasks, yielding reasonable performances as compared to probabilistic approaches. However, the experiments revealed that despite the fact that some information is gained from the unlabeled data the representation obtained is too general for the task we really want to solve.

Hence, we proposed a second non-probabilistic approach in which the representation is learnt jointly with the task to solve. The idea was to learn the representation on unlabeled data while constraining it to give good results on a specific task. The results obtained on several experiments are encouraging. Indeed, we showed that this approach was able to out-perform a similar model using only labeled data and that the joint learning had an essential role in the obtained good results. However, the performance of this approach is bellow that of the state-of-the-art approach, which suggest that a broader analysis of the implementation of this idea is needed.

To summarize, we have explored in this thesis several approaches for text representation using unlabeled data and showed that is was possible to gain some improvement from it. It also became clear that instead of a very general representation of documents, task specific exploitation of unlabeled data should be favored. However, it is still not clear for us that the knowledge extracted from the repeated occurrences of terms in particular contexts should be incorporated in the model through the use of an explicit representation.

Appendix A

Benchmarking the Bootstrap Percentile Test for F_1: Details

A.1 When is H_0 Verified?

Figure A.1 shows the values of the difference between models, for the different models tested in Section 3.4. These values were computed using the F_1 score and classification error $Cerr$ of each of the categorization problem's model, tested over the dataset simulating the entire population D_{true}, and ranked to obtained a clearer view of the ranges they obtained.

As may be noticed in, for example Figure A.1(d), most values of dC_{err} over D_{true} (noted dC^*_{err}) range very near 0 without actually being equal to 0. Thus, in the protocol defined in Section 3.4.3, in order to decided whether or not H_0 was true on D_{true} we proceeded as follows:

Considering a sample of size N, we will say that H_0 is true over D_{true} if $|dM^*| < bound$

where

$$bound = \frac{1}{N} \text{ if } dM^* = dC^*_{err}$$

and

$$bound = \frac{n}{N \times n_{pos}} \text{ if } dM^* = dF^*_1$$

with $n = |D_{true}|$ and n_{pos} being the number of positive examples (documents belonging to the category) in D_{true}.

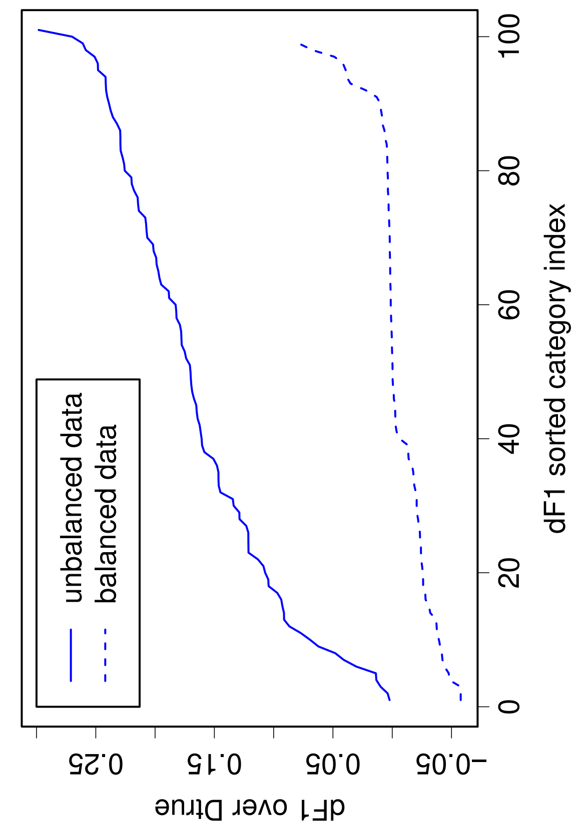
In the case of $dM^* = dC^*_{err}$, the use of such a bound can be justified by the fact that in a set D^s of size N, H_0 will be verified whenever $|\#error_A - \#error_B| < 1 \Leftrightarrow dC_{err} < \frac{1}{N}$. The justification for $dM^* = dF^*_1$ is quite similar. Given that F_1 is computed at the breakeven point we can say that: $F_1 = \text{Recall} = \frac{N_{tp}}{N_{pos}}$ (see eq. (3.2)), where N_{pos} is the number of positive examples in D^s. Thus H_0 will be verified in D^s whenever $|N^A_{tp} - N^B_{tp}| < 1 \Leftrightarrow dF^s_1 < \frac{1}{N_{pos}}$. Finally, since we assume that the sample D^s

is *i.i.d.* and from the same distribution than D_{true} we have: $\frac{N}{N_{pos}} = \frac{n}{n_{pos}}$.

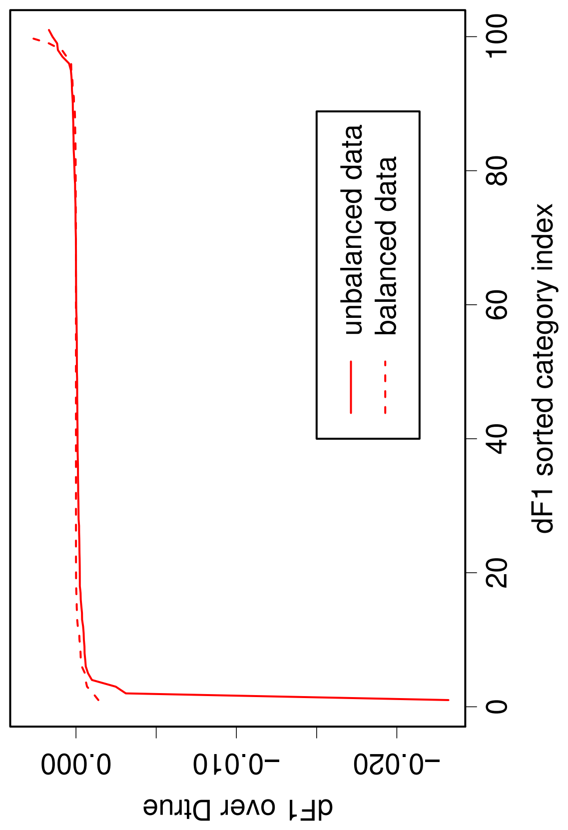
We report in Figure A.2 the proportion of samples for which H_0 was considered as being true in the protocol explained in Section 3.4.3.

A.2 Standard Deviation of Size and Power Estimates

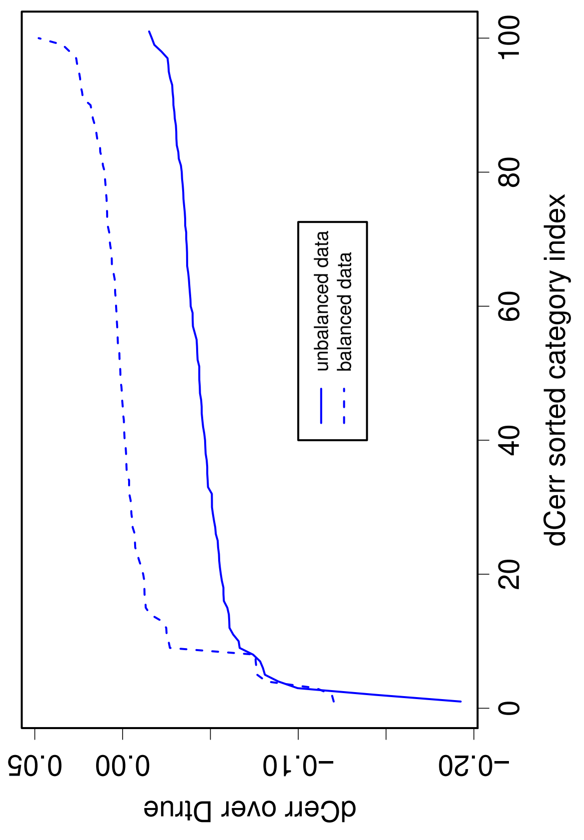
As explained in Section 3.4.3, the Size and Power estimates presented in figures 3.9 and 3.10, are an average of estimates over 101 classification problems (in categories for the unbalanced case, in *meta-categories* for the balanced case). Figures A.3 and A.4 display the standard deviation of the estimates over the classification problems.



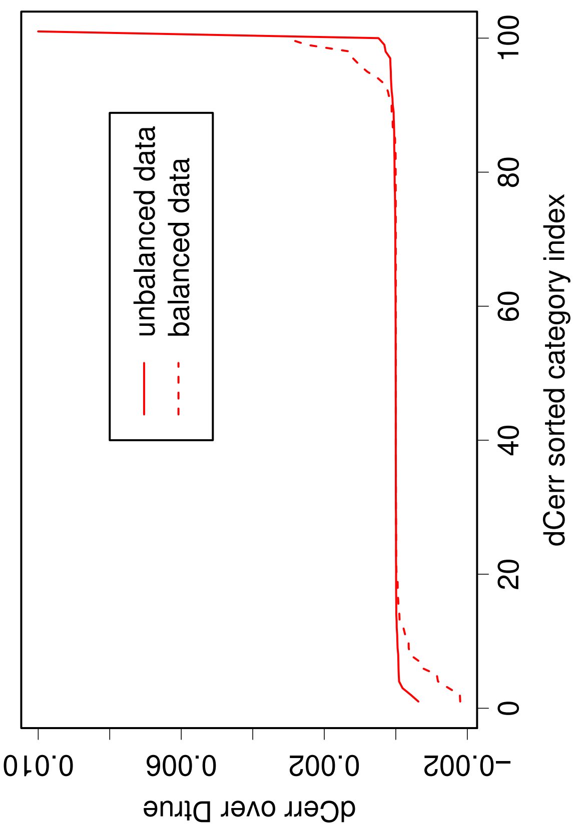
(a) $dF1$ for linear SVM vs MLP



(b) $dF1$ for linear SVM vs RBF SVM

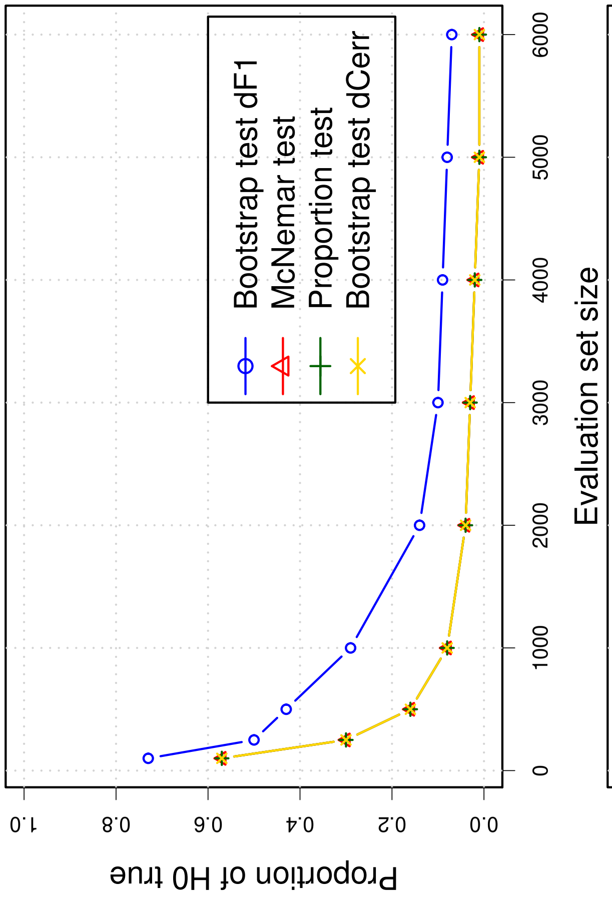


(c) dC_{err} for linear SVM vs MLP

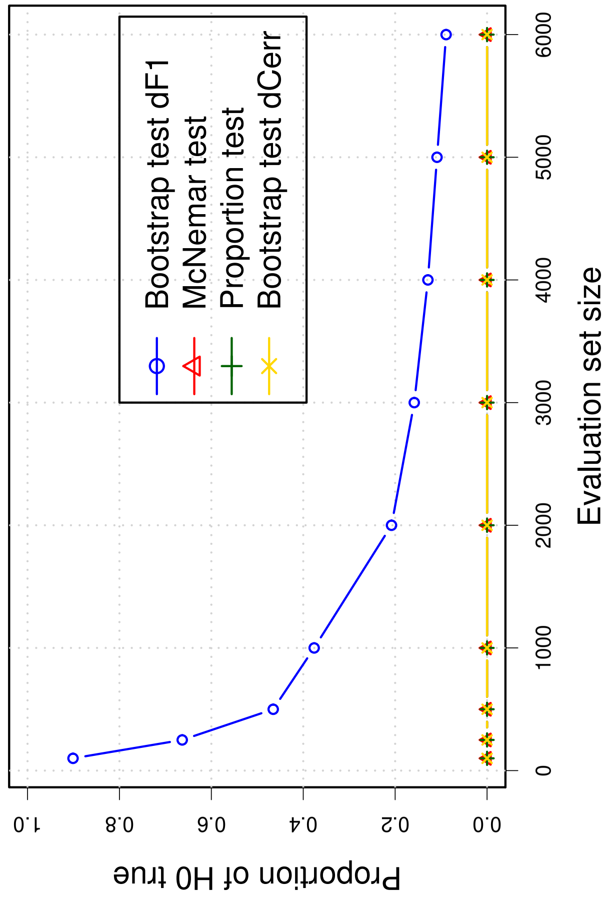


(d) dC_{err} for linear SVM vs RBF SVM

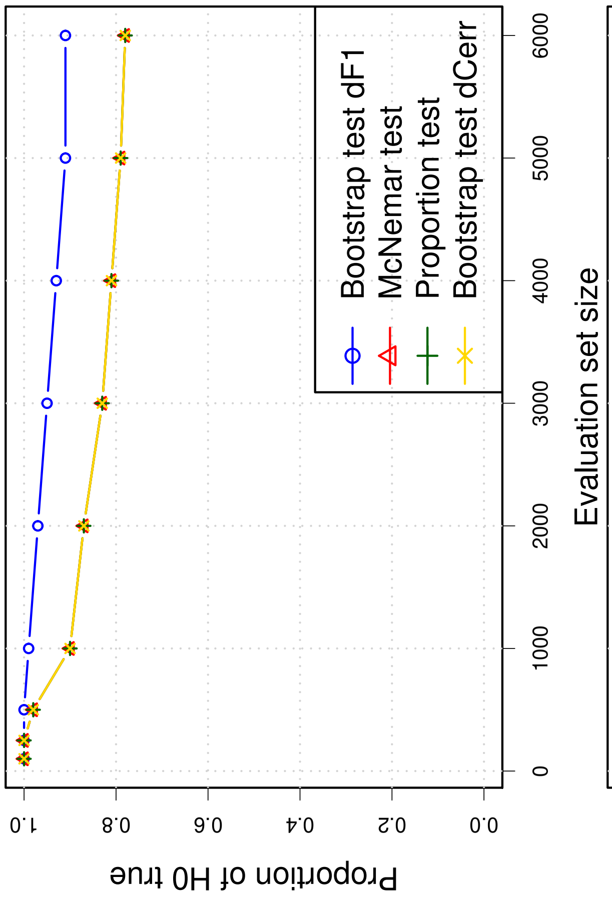
Figure A.1: Values of the differences over D_{true}.



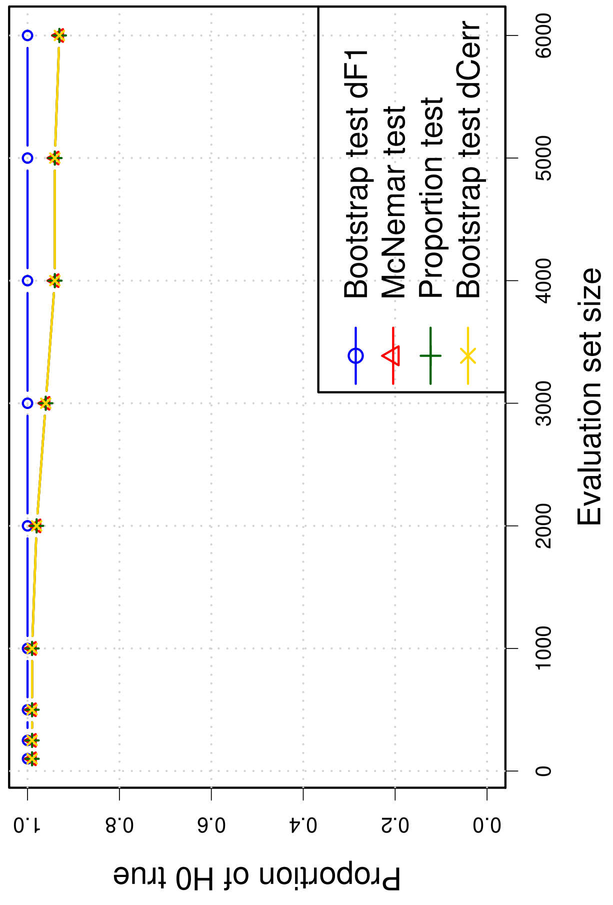
(a) Linear SVM vs MLP - Balanced data



(b) Linear SVM vs MLP - Unbalanced data

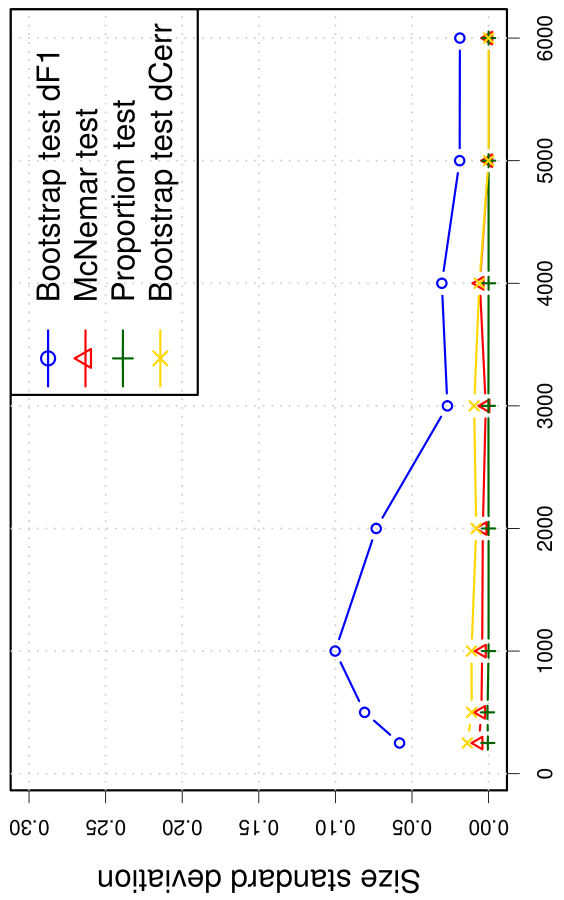


(c) Linear vs RBF SVMs - Balanced data

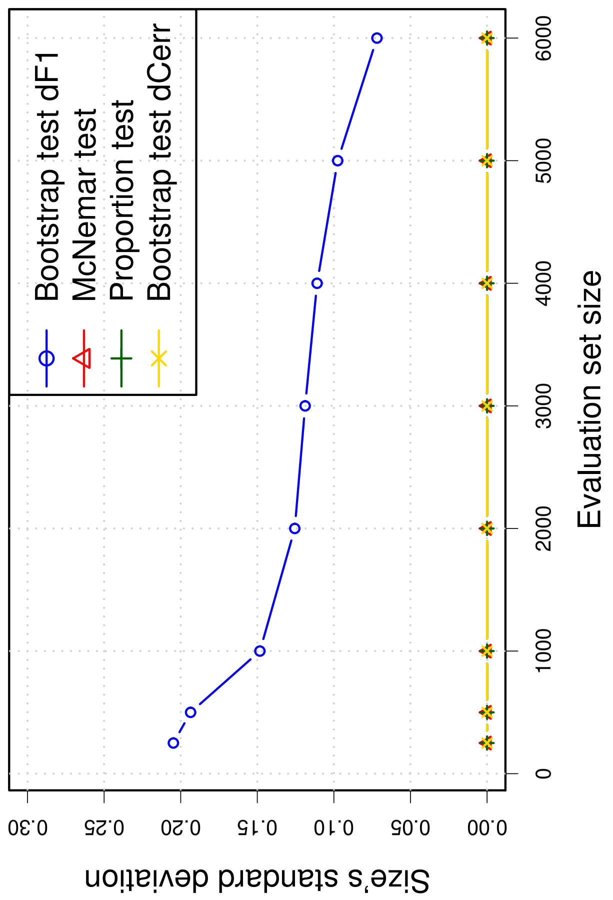


(d) Linear vs RBF SVMs - Unbalanced data

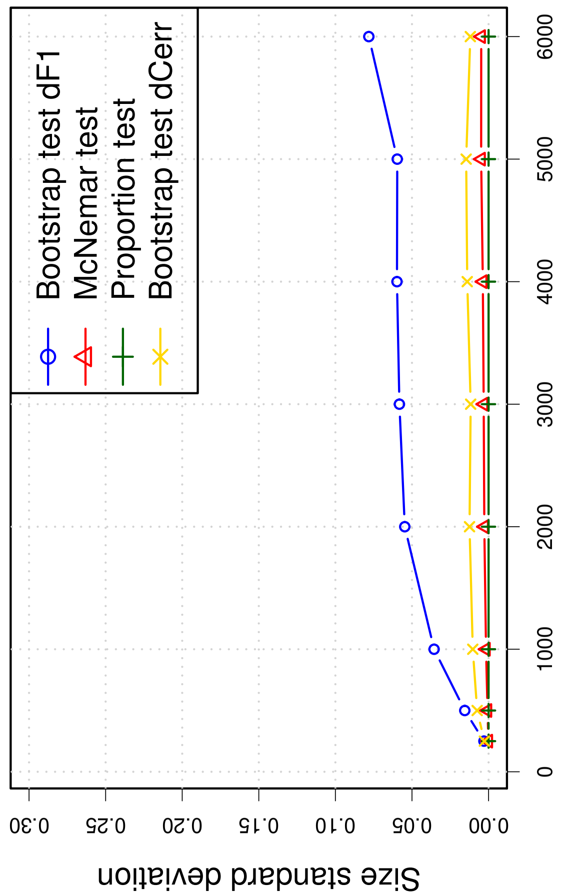
Figure A.2: Proportion of samples for which H_0 was considered as true in D_{true} for several statistical



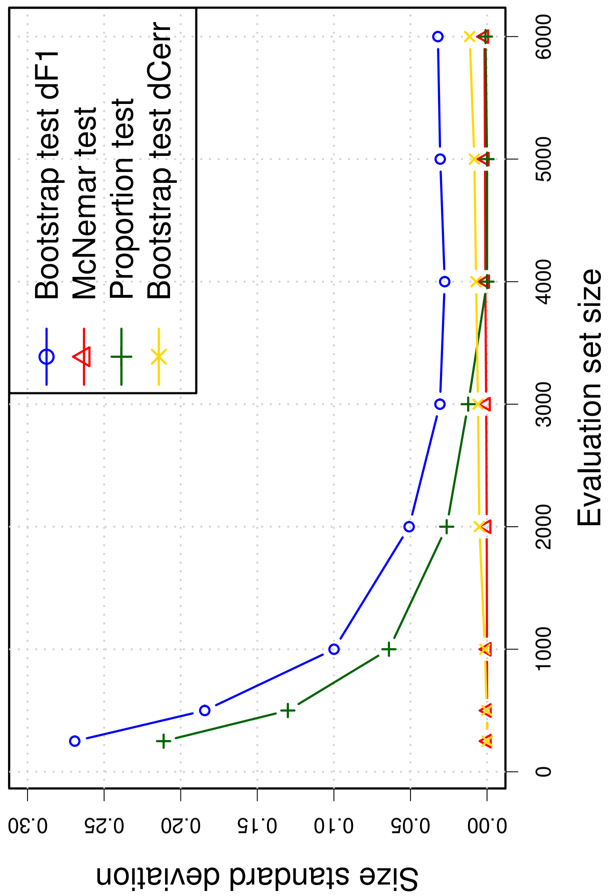
(a) Linear SVM vs MLP - Balanced data



(b) Linear SVM vs MLP - Unbalanced data

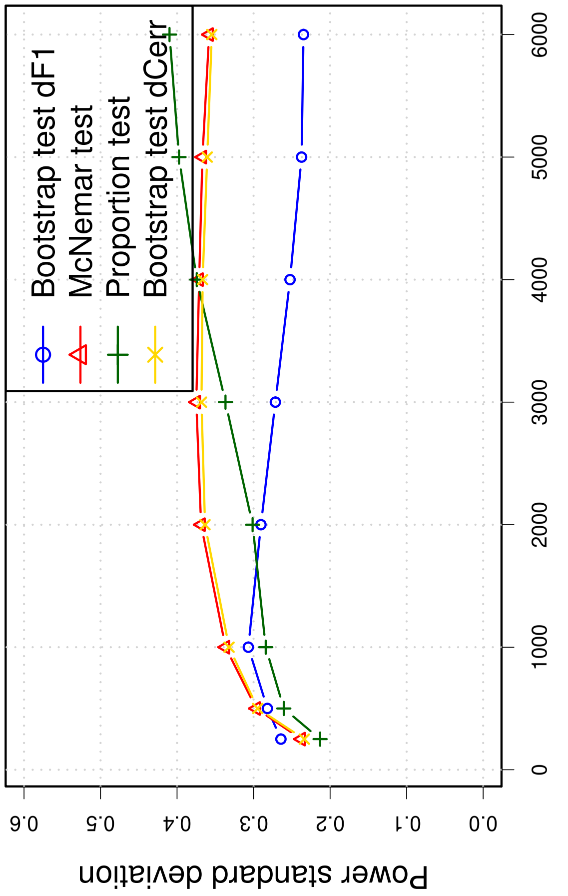


(c) Linear vs RBF SVMs - Balanced data

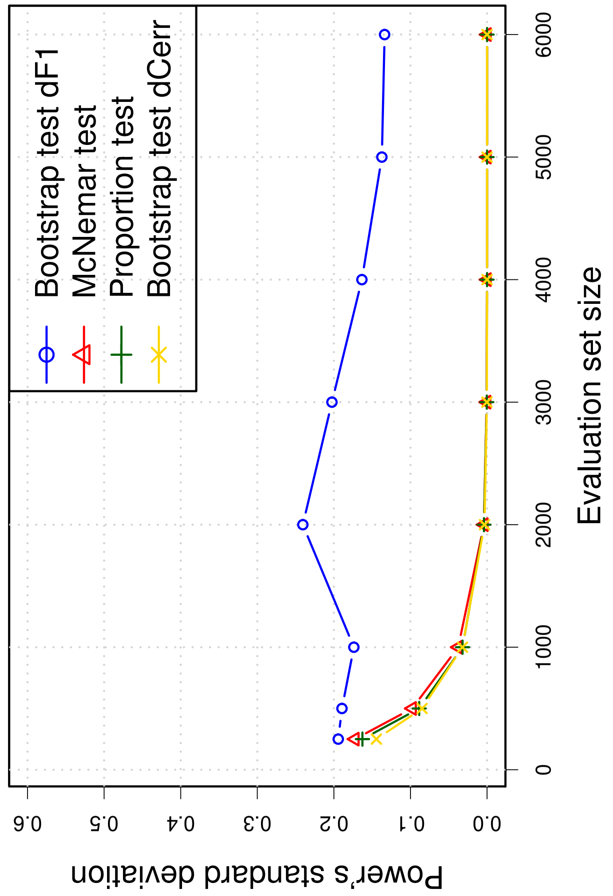


(d) Linear vs RBF SVMs - Unbalanced data

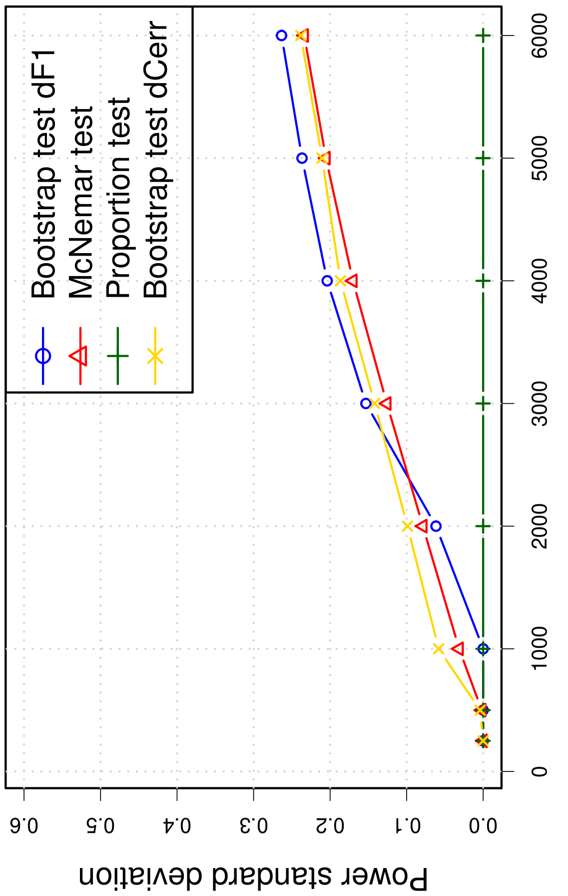
Figure A.3: Standard deviation of the proportion of Type I error over the categories for several statistical



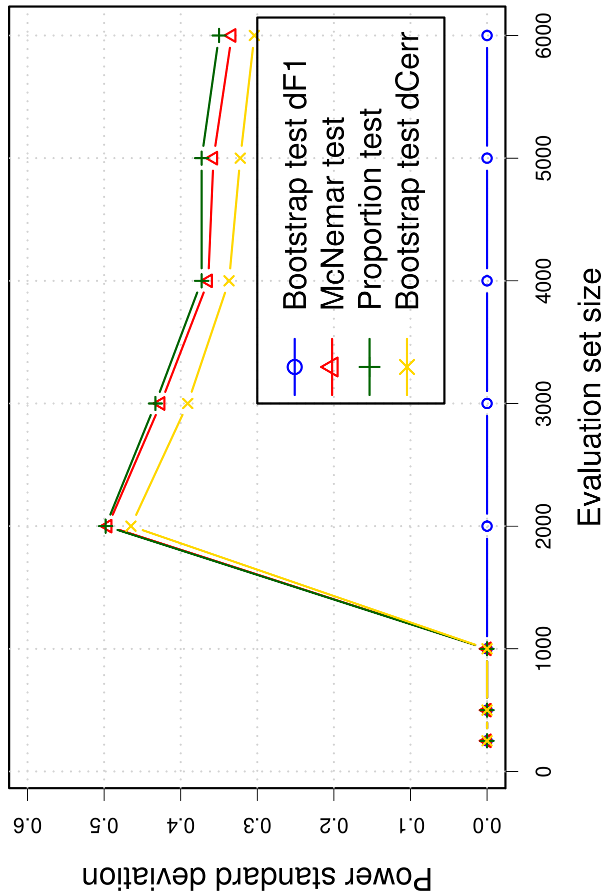
(a) Linear SVM vs MLP - Balanced data



(b) Linear SVM vs MLP - Unbalanced data



(c) Linear vs RBF SVMs - Balanced data



(d) Linear vs RBF SVMs - Unbalanced data

Figure A.4: Standard deviation of the power estimate over the categories for several statistical tests

Appendix B

Expectation-Maximization Derivations for TTMM

In order to perform an EM optimization of a TTMM one has first to get rid of the sum inside the logarithm in the log-likelihood equation (4.2):

$$\mathcal{L}(D|\Upsilon) = \sum_{i=1}^{N} \log \left[\sum_{j=1}^{J} P(H=j) \prod_{l=1}^{M} \left(\sum_{k=1}^{K} P(w_l|T=k)P(T=k|H=j) \right)^{\mathrm{tf}_l(d_i)} \right]. \tag{4.2}$$

This could be easily done if we the hidden variables were instead observed, that is if we were given $\{h_{ij}\}$ the indicator variables specifying which theme j the document d_i was generated from, and $\{t_{jlk}\}$ the indicator variables specifying which topic k the word w_l was generated from given that we were in the theme j context. Indeed, the complete likelihood could be written as:

$$P(D,H,T|\Upsilon) = \prod_{i=1}^{N} \sum_{j=1}^{J} [P(H=j)]^{h_{ij}} \prod_{w_l \in d_i} \left(\sum_{k=1}^{K} [P(T=k|H=j)P(w_l|T=k)]^{t_{jlk}} \right)^{\mathrm{tf}_l(d_i)}.$$

and the complete log-likelihood:

$$\mathcal{L}_{comp}(D|\Upsilon) = \ln[P(D,H,T|\Upsilon)] = \sum_{i=1}^{N} \sum_{j=1}^{J} h_{ij} \left\{ \ln[P(H=j)] + \sum_{w_l \in d_i} \sum_{k=1}^{K} t_{jlk} \ln\left[P(T=k|H=j)P(w_l|T=k)\right] \mathrm{tf}_l(d_i) \right\}. \tag{B.1}$$

with Υ representing the parameters of the model, *ie* $P(H=j)$, $P(T=k|H=j)$ and $P(w_l|T=k)$ $\forall j \in \{1,\ldots,J\}$, $k \in \{1,\ldots,K\}$ and $l \in \{1,\ldots,M\}$.

Notice that the expected values of $\{h_{ij}\}$ and $\{t_{jlk}\}$ are respectively $P(H=j|d_i)$ and $P(T=k|w_l,H=j)$. Hence, the EM algorithm goes as follows.

In the **E-step** the complete log-likelihood is estimated with the help of the posteriors of h_{ij} and t_{jkl} as follows:

$$
\begin{aligned}
P_{ij} &= E[h_{ij}] = P(H=j|d_i) = \frac{P(H=j)P(d_i|H=j)}{\sum_{q=1}^{J} P(H=q)P(d_i|H=q)} \\
&= \frac{P(H=j)\prod_{w_l \in d_i}\left[\sum_{k=1}^{K} P(T=k|H=j)P(w_l|T=k)\right]^{\mathrm{tf}_l(d_i)}}{\sum_{q=1}^{J} P(H=q)\prod_{w_l \in d_i}\left[\sum_{k=1}^{K} P(T=k|H=q)P(w_l|T=k)\right]^{\mathrm{tf}_l(d_i)}}
\end{aligned}
\tag{B.2}
$$

$$
Q_{jkl} = E[t_{jlk}] = P(T=k|w_l, H=j) = \frac{P(T=k|H=j)P(w_l|T=k)}{\sum_{p=1}^{K} P(T=p|H=j)P(w_l|T=p)}. \tag{B.3}
$$

In the **M-step** the expected complete log-likelihood $E[\mathcal{L}_{comp}]$, is maximized with respect to the parameters Υ of the model under their normalization constraints, using the posteriors estimated in the previous step. This leads to maximize:

$$
\begin{aligned}
\mathcal{M} = {} & E[\mathcal{L}_{comp}] + \rho\left(1 - \sum_{j=1}^{J} P(H=j)\right) \\
& + \sum_{j=1}^{J} \lambda_j \left(1 - \sum_{k=1}^{K} P(T=k|H=j)\right) + \sum_{k=1}^{K} \eta_k \left(1 - \sum_{l=1}^{M} P(w_l|T=k)\right).
\end{aligned}
\tag{B.4}
$$

with ρ, λ_j and η_k, $j \in \{1, \dots, J\}$ and $k \in \{1, \dots, K\}$ being the Lagrange multipliers. Which is equivalent to solve the following system of equations:

$$
\begin{aligned}
\frac{\partial \mathcal{M}}{\partial P(H=j)} = 0 \Leftrightarrow \sum_{i=1}^{N} P_{ij} - P(H=j)\rho &= 0, \quad 1 \le j \le J \\
\frac{\partial \mathcal{M}}{\partial P(T=k|H=j)} = 0 \Leftrightarrow \sum_{i=1}^{N} P_{ij} \sum_{w_l \in d_i} n(w_l, d_i) Q_{jkl} - P(T=k|H=j)\lambda_j &= 0, \quad 1 \le j \le J, \quad 1 \le k \le K \\
\frac{\partial \mathcal{M}}{\partial P(w_j|T=k)} = 0 \Leftrightarrow \sum_{i=1}^{N} \sum_{j=1}^{J} n(w_l, d_i) P_{ij} Q_{jkl} - P(w_l|T=k)\eta_k &= 0, \quad 1 \le l \le M, \quad 1 \le k \le K
\end{aligned}
$$

Therefore, the maximum is obtained for the following parameter values:

$$
P(H=j) = \frac{\sum_{i=1}^{N} P_{ij}}{\sum_{q=1}^{J}\sum_{i=1}^{N} P_{iq}} = \frac{\sum_{i=1}^{N} P_{ij}}{N}, \tag{B.5}
$$

given that $\sum_{q=1}^{J} P_{iq} = \sum_{q=1}^{J} P(H = q|d_i) = 1$,

$$\begin{aligned} P(T = k|H = j) &= \frac{\sum_{i=1}^{N} P_{ij} \sum_{w_l \in d_i} Q_{jkl}\,\mathrm{tf}_l(d_i)}{\sum_{p=1}^{K} \sum_{i=1}^{N} P_{ij} \sum_{w_l \in d_i} Q_{jpl}\,\mathrm{tf}_l(d_i)} \\ &= \frac{\sum_{i=1}^{N} P_{ij} \sum_{w_l \in d_i} Q_{jkl}\,\mathrm{tf}_l(d_i)}{\sum_{i=1}^{N} P_{ij} n(d_i)}, \end{aligned} \tag{B.6}$$

given that $\sum_{p=1}^{K} P(T = p|w_l, H = j) = 1$, and

$$P(w_l|T = k) = \frac{\sum_{i=1}^{N} \sum_{j=1}^{J} P_{ij} Q_{jkl}\,\mathrm{tf}_l(d_i)}{\sum_{m=1}^{M} \sum_{i=1}^{N} \sum_{j=1}^{J} P_{ij} Q_{jkm} n(w_m, d_i)}. \tag{B.7}$$

Bibliography

R.K. Ando and T. Zhang. A framework for learning predictive structures from multiple tasks and unlabeled data. *Journal of Machine Learning Research*, 6:1817–1853, Nov 2005.

R. A. Baeza-Yates and B. A. Ribeiro-Neto. *Modern Information Retrieval*. ACM Press / Addison-Wesley, 1999. ISBN 0-201-39829-X.

J. Baxter. A model of inductive bias learning. *Journal of Artificial Intelligence Research*, 12:149–198, 2000.

N.J. Belkin and W.B. Croft. Information filtering and information retrieval: two sides of the same coin? *Commun. ACM*, 35(12):29–38, 1992.

J. R. Bellegarda and D. Nahamoo. Tied mixture continuous parameter models for large vocabulary isolated speech recognition. In *Proc. IEEE Int. Conf. Acoustics, Speech, Signal Processing*, pages 13–16, 1989.

R. E. Bellman. *Adaptive Control Processes*. Princeton University Press, 1961.

S. Bengio and J. Mariéthoz. The expected performance curve: a new assessment measure for person authentication. In *Proceedings of Odyssey 2004: The Speaker and Language Recognition Workshop*, 2004.

S. Bengio, J. Mariethoz, and M. Keller. The expected performance curves. In *Proceedings of the ICML 2005 wokshop on ROC Analysis in Machine Learning*, 2005.

Y. Bengio, R. Ducharme, P. Vincent, and C. Gauvin. A Neural Probabilistic Language Model. *JMLR*, 3:1137–1155, 2003.

F. Bimbot, J.-F. Bonastre, C. Fredouille, G. Gravier, I. Magrin-Chagnolleau, S. Meignier, T. Merlin, J. Ortega-Garcia, D. Petrovsk-Delacrétaz, and D. Reynolds. A tutorial on text-independent speaker verification. *EURASIP Journal on Applied Signal Processing*, 4:430–451, 2004.

M. Bisani and H. Ney. Bootstrap estimates for confidence intervals in ASR performance evaluation. In *Proceedings of ICASSP*, 2004.

C. Bishop. *Neural Networks for Pattern Recognition*. Oxford University Press, 1995.

D. Blei, A. Ng, and M. Jordan. Latent Dirichlet Allocation. *Journal of Machine Learning Research*, 3: 993–1022, 2003.

R. M. Bolle, N. K. Ratha, and S. Pankanti. Error analysis of pattern recognition systems - the subsets bootstrap. *Computer Vision and Image Understanding*, 93:1–33, 2004.

J. Bromley, I. Guyon, Y. LeCun, E. Sackinger, and R. Shah. Signature Verification using a Siamese Time Delay Neural Network. In *Advances in Neural Information Processing Systems 6*, 1993.

W. Buntine. Variational Extensions to EM and Multinomial PCA. In *Proceedings of ECML*, 2002.

C.J.C. Burges, T. Shaked, E. Renshaw, A. Lazier, M. Deeds, N. Hamilton, and G.N. Hullender. Learning to rank using gradient descent. In *ICML*, pages 89–96, 2005.

Lou Burnard. *Users Reference Guide for the British National Corpus*. Oxford University Computing Services, Oxford, 1995.

R. Caruana. Multitask learning. *Machine Learning*, 28(1):41–75, 1997.

O. Chapelle and A. Zien. Semi-supervised classification by low density separation. In *Proceedings of AISTAT*, pages 57–64, 2005.

O. Chapelle, B. Schölkopf, and A. Zien, editors. *Semi-Supervised Learning*. MIT Press, Cambridge, MA, 2006.

S. Chopra, R. Hadsell, and Y. LeCun. Learning a Similarity Metric Discriminatively, with Application to Face Verification. In *Proc. of Computer Vision and Pattern Recognition Conference*, 2005.

C. Cieri, D. Graff, M. Liberman, N. Martey, and S. Strassel. The TDT-2 text and speech corpus. In *Proceedings of the DARPA Broadcast News Workshop*, 1999.

R. Collobert and S. Bengio. Links between perceptrons, MLPs and SVMs. In *Proceedings of ICML*, 2004.

C. Cortes and V. Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, 1995.

N. Cristianini, J. Shawe-Taylor, and H. Lodhi. Latent semantic kernels. *J. Intell. Inf. Syst.*, 18(2-3): 127–152, 2002. ISSN 0925-9902.

A. C. Davison and D. V. Hinkley. *Bootstrap methods and their application*. Cambridge University Press, 1997.

S. C. Deerwester, S. T. Dumais, T. K. Landauer, G. W. Furnas, and R. A. Harshman. Indexing by latent semantic analysis. *Journal of the American Society of Information Science*, 41(6):391–407, 1990. URL `citeseer.nj.nec.com/deerwester90indexing.html`.

A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *J. Royal Statist. Soc. B.*, 39:1–38, 1977.

T.G. Dietterich. Approximate statistical tests for comparing supervised classification learning algorithms. *Neural Computation*, 10(7):1895–1924, 1998.

S. Dumais, J. Platt, D. Heckerman, and M. Sahami. Inductive learning algorithms and representations for text categorization. In *Proceedings of the ACM International Conference on Information and Knowledge Management (CIKM-98)*, 1998.

B. Efron and R. Tibshirani. *An Introduction to the Bootstrap.* Chapman and Hall, 1993.

B. S. Everitt. *The analysis of contingency tables.* Chapman and Hall, 1977.

E. Gabrilovich and S. Markovitch. Feature generation for text categorization using world knowledge. In *Proceedings of The Nineteenth International Joint Conference for Artificial Intelligence*, Edinburgh, Scotland, 2005. URL `http://www.cs.technion.ac.il/ shaulm/papers/pdf/Gabrilovich-Markovitch-ijcai2005.pdf`.

J.S. Garofolo, C.G. Auzanne, and E. Voohees. The TREC spoken document retrieval track: A success story. In *Proceedings of Content-Based Multimedia Information Acces Conference*, 2000.

J.L. Gauvain, L. Lamel, C. Barras, G. Adda, and Y. de Kercardio. The limsi sdr system for TREC-9. In *Proceedings of TREC-9*, 2000.

O. Glickman, I. Dagan, and M. Koppel. A probabilistic classification approach for lexical textual entailment. In *AAAI*, 2005.

O. Glickman, I. Dagan, M. Keller, S. Bengio, and W. Daelemans. Investigating lexical substitution scoring for subtitle generation. In *Proceedings of CoNLL*, 2006.

C. Goutte and E. Gaussier. A probabilistic interpretation of precision, recall and F-score, with implication for evaluation. In *ECIR*, pages 345–359, 2005.

D. Grangier and S. Bengio. Exploiting hyperlinks to learn a retrieval model. In *NIPS Workshop on Learning to Rank*, 2005.

D.M. Green and J.A. Swets. *Signal Detection Theory and Psychophysics.* John Wiley & Sons, 1964.

J.A. Hanley and B.J. McNeil. The meaning and use of the area under a receiver operating characteristic (roc) curve. *Radiology*, 143:29–36, 1982.

T. Hofmann. Learning the similarity of documents. In *Advances in Neural Information Processing Systems 12*, 2000.

T. Hofmann. Unsupervised learning by Probabilistic Latent Semantic Analysis. *Machine Learning*, 42: 177–196, 2001. URL `http://www.cs.brown.edu/people/th/publications.html`.

K. Hornik, M. Stinchcombe, and H. White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359–366, 1989.

T. Joachims. Text Categorization with Support Vector Machines: Learning with Many Relevant Features. In *ECML*, 1998.

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

T. Joachims. Transductive learning via spectral graph partitioning. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2003.

M. I. Jordan, Z. Ghahramani, T. Jaakkola, and L. K. Saul. An introduction to variational methods for graphical models. *Machine Learning*, 37(2):183–233, 1999. URL `citeseer.nj.nec.com/article/jordan98introduction.html`.

Michael I. Jordan, editor. *Learning in graphical models*. MIT Press, Cambridge, MA, USA, 1999.

M. Keller and S. Bengio. Theme Topic Mixture Model: A graphical model for document representation. IDIAP-RR 05, IDIAP, 2004a.

M. Keller and S. Bengio. TTMM: A graphical model for document representation. In *PASCAL Workshop on Text Mining and Understanding*, January 2004b.

M. Keller and S. Bengio. A neural network for text representation. In *Proceedings of International Conference in Artificial Neural Networks (ICANN), Warsaw, Poland*, 2005.

M. Keller and S. Bengio. A multi-task learning approach to document representation using unlabeled data. IDIAP-RR 36, IDIAP, 2006.

M. Keller, S. Bengio, and S.Y. Wong. Benchmarking non-parametric statistical tests. In *Advances in Neural Information Processing Systems, NIPS 18. MIT Press*, 2005.

Y. LeCun. A learning scheme for asymmetric threshold networks. In *Proceedings of Cognitiva 85*, 1985.

Y. LeCun and F. J. Huang. Loss Functions for Discriminative Training of Energy-Based Models. In *Proc. of AIStats*, 2005.

D.D. Lewis, Y. Yang, T. Rose, and F. Li. RCV1: A new benchmark collection for text categorization research. *Journal of Machine Learning Research*, 5:361–397, 2004.

D. Lin. Automatic retrieval and clustering of similar words. In *Proceedings of the 17th international conference on Computational linguistics*, 1998.

C.D. Manning and H. Schütze. *Foundations of Statistical Natural Language Processing*. The MIT Press Cambridge, Massachusetts, 1999.

M. Maron. Automatic indexing: an experimental inquiry. *J. Assoc. Comput. Mach.*, 8(3):404–417, 1961.

G. Miller, R. Beckwith, C. Fellbaum, D Gross, and K. Miller. Five papers on wordnet. Technical report, Stanford University, 1993a.

G.A. Miller, C. Leacock, R. Tengi, and R.T. Bunker. A semantic concordance. In *Proceedings of HLT*, 1993b.

C. Nadeau and Y. Bengio. Inference for the generalization error. *Machine Learning*, 52(3):239–281, 2003.

S. Piperidis, I. Demiros, P. Prokopidis, P. Vanroose, A. Höthker, W. Daelemans, E. Sklavounou, M. Konstantinou, and Y. Karavidas. Multimodal multilingual resources in the subtitling process. In *Proceedings of LREC*, 2004.

M.F. Porter. An algorithm for suffix stripping. *Program*, 14:130–137, 1980.

S.E. Robertson, S. Walker, S. Jones, and M.M. Hancock-Beaulieu. Okapi at TREC-2. In *Proceedings of TREC-2*, 1993.

T.G. Rose, M. Stevenson, and M. Whitehead. The Reuters Corpus Volume 1 - from yesterday's news to tomorrow's language resources. In *Proceedings of the 3rd Int. Conf. on Language Resources and Evaluation*, 2002.

F. Rosenblatt. The perceptron: a perceiving and recognizing automaton. Technical Report 85-460-1, Cornell Aeronautical Laboratory, Ithaca, N.Y., 1957.

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

G. Salton and C. Buckley. Improving retrieval performance by relevance feedback. *Journal of the American Society of Information Science*, 41:288–297, 1990.

G. Salton, A. Wong, and C.S. Yang. A vector space model for automatic indexing. *Communication of the ACM*, 18, 1975.

M. Schultz and T. Joachims. Learning a distance metric from relative comparison. In *Advances in Neural Information Processing Systems 16*, 2003.

H. Schutze, D.A. Hull, and J.O. Pedersen. A comparison of classifiers and document representations for the routing problem. In *Proceedings of the 18th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 229–237, 1995.

F. Sebastiani. Machine learning in automated text categorization. *ACM Computing Surveys*, 34(1):1–47, 2002. URL `http://faure.iei.pi.cnr.it/ fabrizio/Publications/ACMCS02.pdf`.

G. Siolas and F. d'Alché Buc. Support vectors machines based on a semantic kernel for text categorization. In *IEEE-ICANN-IJCNN*, 2000.

K. Sparck-Jones and P. Willett. Overall introduction. In Karen Sparck Jones and Peter Willett, editors, *Readings in information retrieval*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1997. ISBN 1-55860-454-5.

P. Srinivasan. Theasaurus construction. In W.B. Frakes; R. Baeza-Yates, editor, *Information Retrieval: Data Structures and Algorithms*. Prentice Hall, 1992.

C. J. van Rijsbergen. *Information Retrieval*. Butterworths, London, UK, 1975.

V. Vapnik. *The Nature of Statistical Learning Theory*. Springer, second edition, 1995.

V.N. Vapnik and A.Y. Chervonenkis. On the uniform convergence of relative frequencies of events to their probabilities. *Theory of Probability and Its Applications*, 16:264–280, 1971.

P. Verlinde, G. Chollet, and M. Acheroy. Multi-modal identity verification using expert fusion. *Information Fusion*, 1:17–33, 2000.

Y. Yang and X. Liu. A re-examination of text categorization methods. In *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, 1999.

Y. Yang and J. O. Pedersen. A comparative study on feature selection in text categorization. In *Proceedings of ICML*, 1997.

Zweig and Campbell. ROC plots: a fundamental evaluation tool in clinical medicine. *Clinical Chemistry*, 39(4):561–577, 1993.

Curriculum Vitae

Mikaela KELLER

IDIAP Institute
CP 592, rue du Simplon 4
1920 Martigny
Switzerland
tel.: +41 27 721 77 75
email: mikaela.keller@idiap.ch
URL: http://www.idiap.ch/ mkeller/

Born: 11th July 1978 in Paris

Nationalty: French

Education:

2002-2006: PhD candidate on Machine Learning at EPFL (Ecole Polytechnique Fédérale de Lausanne, Switzerland)

2001-2002: DEA Stochastic and Statistic Modelisation, with the option Applied Statistics, at Université Paris XI (data mining, semi-parametric statistic, modelisation and forecast, spatial statistics)

2000-2001: Maîtrise de Mathémathiques Appliquées at Université Paris VI, completed at UQAM (Université du Québec A Montréal) (inferential statistics, data mining, stochastic processes, graph theory, Java and C++ programming)

1998-2000: Licence de Mathématiques pures de l'Université Paris VI (measure theory, differential calculus and geometry, complex analysis)

1996-1998: DEUG MIAS at Université Paris XI

Professional & Scientific experience:

2002-2006: Research Assistant at IDIAP (Dalle Molle Institute for Perceptual Artificial Intelligence)

April-July 2002: Internship in Machine Learning at LIP6 (Computer Science Laboratory of Université Paris VI)

Teaching experience:

2005-2006: Laboratory courses for lectures on "Statistical Machine Learning from Data" at the EPFL's doctoral school

2001-2002: Laboratory courses on C++ programming first year course at IUT d'Orsay

Publications:

- O. Glickman, I. Dagan, M. Keller, S. Bengio and W. Daelemans. Investigating Lexical Substitution Scoring for Subtitle Generation. Proceedings of the 10th Conference on Computational Natural Language Learning (CoNLL-2006).
- S. Bengio, J. Mariéthoz and M. Keller. The Expected Performance Curve. In International Conference on Machine Learning, ICML, Workshop on ROC Analysis in Machine Learning, 2005.
- M. Keller and S. Bengio. A neural network for text representation. In International Conference on Artificial Neural Networks, ICANN, 2005.
- M. Keller, S. Bengio and S.Y. Wong. Benchmarking Non-Parametric Statistical Tests. Advances in Neural Information Processing Systems (NIPS) 18, 2005.
- M. Keller and S. Bengio. Theme Topic Mixture Model: A Graphical Model For Document Representation. In: *Learning Methods for Text Understanding and Mining, 26 - 29 January 2004, Grenoble, France.*

Programming Language Knowledge:

- Caml, Pascal, C, C++, Java, perl, python
- statistical softwares : Splus/R

Languages:

- French, Spanish, English