UNIVERSIDADE TÉCNICA DE LISBOA
INSTITUTO SUPERIOR TÉCNICO

Improving Methods for Single-label Text Categorization

Ana Margarida de Jesus Cardoso Cachopo
(Mestre)

Dissertação para obtenção do Grau de Doutor em
Engenharia Informática e de Computadores

Orientador: Doutor Arlindo Manuel Limede de Oliveira
Co-Orientador: Doutor João Emílio Segurado Pavão Martins

Júri:

Presidente: Reitor da Universidade Técnica de Lisboa
Vogais: Doutor Pavel Bernard Brazdil
Doutor João Emílio Segurado Pavão Martins
Doutora Isabel Maria Martins Trancoso
Doutor Arlindo Manuel Limede de Oliveira
Doutor Mário Jorge Gaspar da Silva
Doutor Pável Pereira Calado

Julho de 2007

Resumo

Com o crescimento do volume de informação em forma digital, aumenta a necessidade de usar técnicas de classificação de texto para recuperar informação relevante a partir de toda aquela que está disponível.

Para melhorar a qualidade do classificador, propõe-se a combinação de métodos de classificação diferentes. Nesta dissertação, mostra-se que a média da qualidade obtida pela combinação de k-NN com LSI, denominada k-NN-LSI, é superior à média da qualidade obtida por cada um dos métodos originais. Mostra-se também que a combinação de SVM com LSI, denominada SVM-LSI, funciona melhor do que qualquer dos métodos originais, para alguns dos conjuntos de dados usados neste trabalho. Tendo em conta que o SVM é apontado como o melhor método de classificação em vários estudos já publicados, é particularmente interessante que o SVM-LSI produza resultados ainda melhores em algumas situações.

Para diminuir o número de documentos de treino necessários, propõe-se a utilização de um classificador semi-supervisionado baseado em centróides que combina informação acerca de pequenas quantidades de documentos etiquetados com outros documentos por etiquetar. Utilizando um conjunto de dados sintético e três conjuntos de dados reais, mostra-se empiricamente que, se o modelo que o classificador criou inicialmente com os dados for suficientemente preciso, a utilização de documentos não-etiquetados melhora os resultados. Por outro lado, se o modelo inicial dos dados não for suficientemente preciso, a utilização de documentos não-etiquetados piora os resultados.

Apresenta-se também um estudo comparativo entre vários métodos de classificação bem conhecidos e as combinações de métodos propostas neste trabalho.

Abstract

As the volume of information in digital form increases, the use of *Text Categorization* techniques aimed at finding relevant information becomes more necessary.

To improve the quality of the classification, I propose the combination of different classification methods. The results show that k-NN-LSI, the combination of k-NN with LSI, presents an average Accuracy on the five datasets that is higher than the average Accuracy of each original method. The results also show that SVM-LSI, the combination of SVM with LSI, outperforms both original methods in some datasets. Having in mind that SVM is usually the best performing method, it is particularly interesting that SVM-LSI performs even better in some situations.

To reduce the number of labeled documents needed to train the classifier, I propose the use of a semi-supervised centroid-based method that uses information from small volumes of labeled data together with information from larger volumes of unlabeled data for text categorization. Using one synthetic dataset and three real-world datasets, I provide empirical evidence that, if the initial classifier for the data is sufficiently precise, using unlabeled data improves performance. On the other hand, using unlabeled data actually degrades the results if the initial classifier is not good enough.

The dissertation includes a comprehensive comparison between the classification methods that are most frequently used in the Text Categorization area and the combinations of methods proposed.

Palavras-chave

Classificação de Texto
Conjuntos de Dados
Medidas de Avaliação
Combinação de Métodos de Classificação
Classificação Semi-supervisionada
Classificação Incremental

Keywords

Text Categorization
Datasets
Evaluation Metrics
Combining Classification Methods
Semi-supervised Classification
Incremental Classification

Agradecimentos

First, I would like to thank Professor Arlindo Oliveira, my adviser, for the effort he put into guiding me throughout this work.

I would also like to thank Professor João Pavão Martins, my co-adviser, for his support during my work, in particular in these last months.

I would not have been able to endure these seven years without the love and support from my family. I thank my parents Aníbal and Isabel, and my sister Susana for always believing in me. Thanks also to my son Rui for postponing so many fun things until I finished my dissertation, and for his understanding when I was not able to give him the attention he deserves. And above all, I lovingly thank my husband João for all his support, many insightful discussions, comments on earlier versions of this dissertation, and help throughout this work.

Julho de 2007
Ana Margarida de Jesus Cardoso Cachopo

Contents

List of Figures

List of Tables

List of Algorithms

Glossary

Notation	Description	Page
TC	Text Categorization or Text Classification	1
IR	Information Retrieval	1
ML	Machine Learning	1
tfidf	Term frequency/inverse document frequency	13
td	Term distributions	14
Vector	Vector Method	17
k-NN	k-Nearest Neighbors	18
Naive Bayes	Naive Bayes	18
Centroid	Centroid-based Method	21
C-Rocchio	Centroid-based Rocchio	21
C-Average	Centroid-based Average	22
C-Sum	Centroid-based Sum	22
C-NormSum	Centroid-based Normalized Sum	22
LSI	Latent Semantic Indexing	23
SVD	Singular Value Decomposition	23
SVM	Support Vector Machines	24
Accuracy	Accuracy = MRR(1)	29
MRR	Mean Reciprocal Rank	31
EM	Expectation Maximization	36
Inc	Incremental	38
IREP	Information Retrieval Experimentation Package	54
k-NN-LSI	Combination of k-NN with LSI	80
SVM-LSI	Combination of SVM with LSI	80
C-EM	Semi-supervised classification using EM and a centroid-based method	92
C-Inc	Incremental semi-supervised classification using a centroid-based method	94

Notation

Notation	Description
t_i	Term i
$T = \{t_1, ..., t_p\}$	Set of p terms
$\|T\|$	Number of terms, generally p
d_j	Document j
$D = \{d_1, ..., d_r\}$	Set of r documents
$\|D\|$	Number of documents, generally r
$\overrightarrow{d_j}$	Vector representing a document
c_k	Class k
$C = \{c_1, ..., c_q\}$	Set of q classes
$\|C\|$	Number of classes, generally q
$\overline{c_k} = C - \{c_k\}$	All classes except c_k
$\overrightarrow{c_k}$	Vector representing the centroid of class c_k
D_{c_k}	Set of documents belonging to class c_k
$\|D_{c_k}\|$	Number of documents belonging to class c_k
$D_{\overline{c_k}} = D - D_{c_k}$	Set of documents not belonging to class c_k

Chapter 1

Introduction

This chapter gives an introduction to the area of Text Categorization, presents the contributions of this work, and describes the outline of this dissertation.

1.1 Text Categorization

Text Categorization (TC), also known as Text Classification, is the task of automatically classifying a set of text documents into different categories from a predefined set. If a document belongs to exactly one of the categories, it is a single-label classification task; otherwise, it is a multi-label classification task.

TC uses several tools from Information Retrieval (IR) and Machine Learning (ML) and has received much attention in the last years from both researchers in the academia and industry developers.

TC uses tools developed by IR researchers because it is a content-based document management task, sharing many characteristics with other IR tasks, such as text search, where the goal is to find the set of documents (or document passages) most relevant to a particular query. For example, documents used in classification

tasks are indexed using the same techniques as in IR; moreover, documents are compared and the similarity between them is measured using techniques originally developed for IR. The evaluation of classification tasks is often done using the same effectiveness measures as in IR.

TC is of interest also for ML researchers, because applications of TC are a challenging benchmark for their own techniques and methodologies. This is because TC applications use very high-dimensional feature spaces and very large amounts of data.

For industry developers, TC is important because of the large quantity of documents that need to be properly processed and classified. But, even more important, is the fact that automatic TC techniques have reached accuracy levels that rival the performance of trained professionals.

TC techniques are used in a variety of tasks, such as finding answers to similar questions, classifying news by subject or newsgroup, sorting spam from legitimate e-mail messages, organizing documents in different folders, etc. In each case, the goal of the categorization is to automatically assign the appropriate classification to each document that needs to be classified.

In general, in order to learn a classifier that is able to correctly classify unseen documents, it is necessary to train it with some pre-classified documents from each category, in such a way that the classifier is then able to generalize the model it has learned from the pre-classified documents and use that model to correctly classify the unseen documents. Several different types of classifiers have been studied in the IR field, and they can be used for TC by considering that the class(es) of the document to classify is(are) the class(es) of the most similar document(s) found. One important point is that the classifier should be effective, independently of the domain of the documents to be classified.

Performance evaluation can be done using training efficiency (how long it takes to learn the classifier using the training data), classification efficiency (how long it takes to classify a document using the classifier) and classification effectiveness (average correctness of the classifier). In order to perform this evaluation, a set of pre-classified documents is split into a training and a test set, that are then used to train and to evaluate the classifier, respectively.

In many classification tasks, however, it is a lot easier to find unclassified (also called *unlabeled*) than classified (also called *labeled*) documents, and often classification is a boring and time-consuming task that has to be done by humans. In such cases, one can try to improve the results obtained by the classifier by using, not only the labeled documents, but also some unlabeled documents that are available. Information about the unlabeled documents is generally incorporated into the classifier using a well-known statistical technique called Expectation Maximization (EM) and the task is then called *semi-supervised* classification.

Existing classifiers can be improved along different directions. They can be better at classifying documents by making less mistakes; they can be faster at learning the classifier or at classifying an unseen document; or they may need smaller volumes of labeled data to be trained on.

This work aims at finding better single-label classifiers along two of these directions: (1) by combining good classifiers in order to improve the quality of the results that they could achieve individually, and (2) by finding computationally efficient ways to use unlabeled documents to improve the classification model that would be possible to build using only the labeled documents, and at distinguishing the situations when the unlabeled documents are useful from when they are not.

1.2 Contributions

Before starting to work in single-label TC, it was necessary to build a computational framework that implemented several of the already known classifiers, and that could use some datasets to train and evaluate those classifiers. Afterwards, it was possible to study the combination of different classification methods, and ways of using unlabeled documents. I hope that future researchers can capitalize on this work and further develop some of the areas involved in this research.

Some achievements of this work are the following, in chronological order.

- Provide single-label datasets.

  I had access to a large volume of email messages sent to a bank's help desk and developed a classification scheme for these messages by manually classifying 1391 of them into 37 different categories.

  There are several standard collections used for evaluating TC tasks, where each document belongs to one or more categories. Because this work is concerned with the classification of documents into a single category, I edited existing standard multi-label datasets so that each document had exactly one label, and made these datasets publicly available from my homepage.

- Propose the use of the Mean Reciprocal Rank to evaluate the performance of classification methods on single-label classification tasks.

  The Mean Reciprocal Rank (MRR) is an evaluation metric that was used to evaluate the tasks in the "Question Answering" tracks of the TREC conferences. I argue that, although the concern of this work is with single-label classification tasks, the MRR is very well suited to evaluate the results presented, and propose its use as an extension to the commonly used Accuracy measure.

- Comparison of two term weighting alternatives for the text in the docu-

ments.

There are several ways to represent the importance of a particular term for a particular document, the most widely used in IR being *tfidf*. A more recent approach for weighting the terms in documents, that takes into account the statistical distribution of terms within classes, will be referred to as *td*. I compared these two term weighting alternatives for the text in the documents and showed that for six datasets and four classification methods, the traditional *tfidf* outperforms *td* in 17 of the 24 possible cases.

- Comparison of several centroid-based classification methods.

  Centroid-based methods are conceptually very simple: each class is represented by a "prototypical" document that summarizes the characteristics of the class. This simplicity makes them very easy to implement and the reduced amount of information that they keep (one "prototypical" document per class) makes them very fast, both in the training and in the test phases. There are several ways of determining the centroid of a class of documents. In this work I compared four of the most common and showed that the classification method where the centroid of a class corresponds to the normalized sum of the documents that belong to that class provides the best results for the six datasets. Moreover, results have shown that this method provides classification results very close to other state-of-the-art methods, namely Support Vector Machines.

- Computational framework.

  The Information Retrieval Experimentation Package, or IREP for short, is the computational framework that resulted from this work. IREP provides a good starting point for more developments, because it allows numerous combinations between datasets, representation alternatives, and classification methods.

The innovative part of this work concerns both the combination of different classification methods for improving the model of the data and the use of unlabeled documents to improve the classifier:

- I propose the combination of different classification methods, namely k-Nearest Neighbors and Support Vector Machines with Latent Semantic Indexing.

  The results obtained with this work show that k-NN-LSI, the combination of k-NN with LSI, usually shows a performance that is intermediate between the original two methods. Overall, k-NN-LSI presents an average Accuracy over the six datasets that is higher than the average Accuracy of each original method.

  Results also show that SVM-LSI, the combination of SVM with LSI, outperforms both original methods in some datasets. Having in mind that SVM usually is the best performing method in several published comparisons, it is particularly interesting that SVM-LSI performs even better in three of the six datasets.

- I propose the use of centroid-based methods for incorporating information about unlabeled documents for semi-supervised and incremental single-label text categorization. The results obtained using one synthetic dataset and three real world datasets show that, if the initial model of the data is good enough, using unlabeled data improves results, while it actually worsens results if the initial model is not good enough.

- I provide a comprehensive comparison between the classification methods that are most frequently used in the TC area and the combinations of methods proposed in this work.

1.3 Outline of the Dissertation

The remainder of this dissertation is organized in six chapters.

Chapter 2 describes the work that was previously done in the several areas of Text Categorization related to this research, namely Document Indexing, Classification Methods, Evaluation Metrics, Combinations Between Methods, Semi-supervised Classification, Incremental Classification, and Datasets.

Chapter 3 describes the experimental setup that was used for this work.

Chapter 4 describes the experiments that were performed to compare existing text classification methods and the results that were obtained.

Chapter 5 describes the combinations of methods that were proposed in this work and the results that were obtained with them.

Chapter 6 describes how centroid-based methods can be used to perform semi-supervised classification and how results improve by using them.

Finally, Chapter 7 concludes and mentions the work that I plan to do in the future.

Chapter 2

Related Work

This chapter describes previous work that was done in some areas related to Text Categorization.

It starts by describing indexing techniques for the terms in documents and some of the most widely used methods for classification. Then, it describes how these methods are usually evaluated. Afterwards, it describes how these methods can be combined, and how unlabeled documents can be used to improve evaluation results. Finally, it describes some data collections and why they are important in this area.

2.1 Text Categorization

The main goal of Text Categorization or Text Classification (TC) is to derive methods for the classification of natural language text [Sebastiani, 2002]. The objective is to automatically derive methods that, given a set of training documents $D = \{d_1, \ldots, d_r\}$ with known categories $C = \{c_1, \ldots, c_q\}$ and a new document q, which is usually called the *query*, will predict the query's category, that is, will

associate with q one or more of the categories in C.

The methods that are used in TC are generally the same that are used in the more general area of Information Retrieval (IR), where the goal is to find documents or passages within documents that are relevant, or related to, a particular query. By considering the document to classify as the query and the classes of the documents that are retrieved as the possible classes for the query, a method developed for IR can be used for TC tasks.

TC techniques are necessary to find relevant information in many different tasks that deal with large quantities of information in text form. Some of the most common tasks where these techniques are applied are: finding answers to similar questions that have been answered before; classifying news by subject or newsgroup; sorting spam from legitimate e-mail messages; finding Internet pages on a given subject, among othrs. In each case, the goal is to assign the appropriate category or label to each document that needs to be classified.

2.1.1 Single-Label vs Multi-label

Depending on the application, documents can be classified into one or more classes. For instance, a piece of news regarding how the Prime Minister spent his vacations may be classified both as politics and in the social column. This kind of classification is called *multi-label* classification, where any number $0 < n_j \leq |C|$ of classes may be assigned to each document $d_j \in D$.

In other situations, however, documents can have only one classification, for example, when distinguishing between spam and legitimate e-mail messages. This kind of classification is called *single-label* classification, where exactly one class $c_k \in C$ must be assigned to each document $d_j \in D$.

A special case of single-label classification is *binary* classification, in which,

given a category c_k , each $d_j \in D$ must be assigned either to c_k or to its complement $\overline{c_k}$. A problem of multi-label classification under $C = \{c_1, \ldots, c_q\}$ can be tackled as $|C|$ independent binary classification problems under $\{c_k, \overline{c_k}\}$, for $i = 1, \ldots, |C|$. In this case, a classifier for C is thus actually composed of $|C|$ binary classifiers.

The focus of this work is in the case where each document belongs to a single class, that is, single-label classification.

2.2 Document Term Weighting

Document indexing is the process of mapping a document into a compact representation of its content that can be interpreted by a classifier. The techniques used to index documents in TC are borrowed from Information Retrieval, where text documents are represented as a set of index terms which are weighted according to their importance for a particular document and for the collection in general [Salton and Lesk, 1968; Salton, 1971; Sebastiani, 2002; Yang and Liu, 1999]. That is, a text document d_j is represented by an n-dimensional vector $\overrightarrow{d_j}$ of index terms or keywords, where each index term corresponds to a word that appears at least once in the initial text and has a weight associated to it, which should reflect how important this index term is.

For efficiency reasons, some of the classification methods used in TC make simplifications that may not be totally justified but that have been experimentally validated. Some of these simplifications are:

1. They ignore the natural language structure of text. They do not try to fully "understand" a document, by semantically analyzing it, but they can use the structure that is easy to find (like HTML tags, for instance), even when they are processing large volumes of information. Besides being more effi-

cient than Natural Language Understanding techniques, this approach also has the advantage of not using domain-dependent techniques.

2. They assume that weights for the index terms are mutually independent. Although this simplification allows for a much easier treatment of the documents, weights for the index terms usually are not independent, because the fact that one of the index terms appears in the text may increase the probability of finding another term that is usually related to it, as in "computer network", for instance.

3. They ignore the order of words. In this "bag-of-words" approach, all the texts that are permutations of the same words are considered equal. This simplification is not always justified, but is necessary for efficiency reasons.

Regarding the problem of how to choose the terms that should be used to represent a document, several studies analyzed the effect of the representation of text documents on the results obtained by text classifiers. These studies have shown that structurally simple representations produced without linguistic or domain knowledge are as effective as others which are more complex [Lewis, 1992b; Lewis and Jones, 1996]. Moreover, the use of larger indexing units, such as frequently adjacent pairs or syntactically determined phrases, has not shown systematic patterns of improvement [Lewis, 1992a; Caropreso et al., 2001; Moschitti and Basili, 2004], which means that terms are usually made to coincide with single words, stemmed or not.

Regarding the problem of how to weight the terms in the documents, term weights can be binary-valued, indicating presence or absence of the term in the document; or real-valued, indicating the importance of the term in the document. There are multiple approaches for how real-valued weights can be computed. For example, Sable et al [Sable and Church, 2001] introduce a bin-based term weighting method intended for tasks where there is insufficient training data to estimate

a separate weight for each word. Debole et al [Debole and Sebastiani, 2004b] propose supervised term weighting, where information on the membership of training documents to categories be used to determine term weights. However, none of the more recent approaches consistently outperforms the popular term frequency / inverse document frequency [Salton and Buckley, 1988].

For the reasons explained above, text documents are usually represented as a set of index terms which are weighted according to their importance for a particular document and for the collection in general, where the words in the document correspond to the index terms. The importance of each term, that is, its weight, can be computed in several ways, and the next sections describe the popular *tfidf* and one more recent alternative.

2.2.1 Term Frequency / Inverse Document Frequency

In this case, which is the most usual in TC, the weight w_{ij} of a term t_i in a document $\overrightarrow{d_j}$ increases with the number of times that the term occurs in the document and decreases with the number of times the term occurs in the collection. This means that the importance of a term in a document is proportional to the number of times that the term appears in the document, while the importance of the term is inversely proportional to the number of times that the term appears in the entire collection.

This term-weighting approach is referred to as *term frequency/inverse document frequency* (*tfidf*) [Salton and Buckley, 1988].

Formally, w_{ij}, the weight of term t_i for document $\overrightarrow{d_j}$, is defined as:

$$w_{ij} = \frac{freq_{ij}}{max_l(freq_{lj})} \times \log \frac{|D|}{n_{t_i}} \tag{2.1}$$

where $freq_{ij}$ is the number of times that term t_i appears in document $\overrightarrow{d_j}$, $|D|$ is the total number of documents in the collection, and n_{t_i} is the number of documents where term t_i appears.

2.2.2 Term Distributions

A recent approach [Lertnattee and Theeramunkong, 2004] proposes a more sophisticated term weighting method than *tfidf*, based on term frequencies within a particular class and within the collection of training documents. This approach will be referred to as term distributions (*td*).

The weight of a term using term distributions is determined by combining three different factors, that depend on the average term frequency of term t_i in the documents of class c_k, represented as $\overline{tf}_{ik}$:

$$\overline{tf}_{ik} = \frac{\sum_{d_j \in c_k} tf_{ijk}}{|D_{c_k}|} \tag{2.2}$$

where D_{c_k} represents the set of documents that belong to class c_k, $|D_{c_k}|$ the number of documents belonging to class c_k, tf_{ijk} the frequency of term t_i in document $\overrightarrow{d_j}$ of class c_k, and $|C|$, which will be used in the next formulas, represents the number of classes in a collection.

The *inter-class standard deviation* of a term t_i is independent of class. Intuitively, a term with a high *icsd* distributes differently among classes and should have higher discriminating power for classification than the others. This factor promotes a term that exists in almost all classes but its frequencies for those classes are quite different. In this situation, the conventional factors *tf* and *idf* are not helpful.

$$icsd_i = \sqrt{\frac{\sum_k \left[\overline{tf}_{ik} - \frac{\sum_k \overline{tf}_{ik}}{|C|}\right]^2}{|C|}} \tag{2.3}$$

The *class standard deviation* term t_i in a class c_k depends on the different frequencies of the term in the documents of that class, and varies from class to class. This factor is important because different terms may appear with quite different frequencies among documents in the class. This difference can be compensated by using this deviation. A term with a high *csd* will appear in most documents in the class with quite different frequencies and should not be a good representative term of the class. A low *csd* of a term for a class may be triggered because either the occurrences of the term are nearly equal for all documents in the class or because the term rarely occurs in the class.

$$csd_{ik} = \sqrt{\frac{\sum_{d_j \in c_k} \left[tf_{ijk} - \overline{tf}_{ik} \right]^2}{|D_{c_k}|}} \tag{2.4}$$

The *standard deviation* of a term t_i depends on the frequency of that term in the documents in the collection and is independent of classes. Different terms may appear with quite different frequencies among documents in the collection. This difference can also be compensated by using this deviation. A term with a high *sd* will appear in several documents in the collection with quite different frequencies. A low *sd* of a term may be caused either because the occurrences of the term are nearly equal for all documents in the collection or because the term rarely occurs in the collection.

$$sd_i = \sqrt{\frac{\sum_k \sum_{d_j \in c_k} \left[tf_{ijk} - \frac{\sum_k \sum_{d_j \in c_k} tf_{ijk}}{\sum_k |D_{c_k}|} \right]^2}{\sum_k |D_{c_k}|}} \tag{2.5}$$

Considering term distributions, wtd_{ijk}, the term distributional weight of term t_i for document $\overrightarrow{d_j}$ in class c_k is defined as:

$$wtd_{ijk} = w_{ij} \times icsd_i^{\alpha} \times csd_{ik}^{\beta} \times sd_i^{\gamma} \tag{2.6}$$

where α, β and γ are used to adjust the relative weight of each factor and to indicate whether it is used as a multiplier or as a divisor for the term's *tfidf* weight, w_{ij}. Lertnattee et al [Lertnattee and Theeramunkong, 2004] performed several experiments and propose $\alpha = 0.5$, $\beta = -0.5$ and $\gamma = -0.5$ as the better combination of α, β and γ.

2.2.3 Document Length Normalization

Naturally, long documents contain more terms than short documents. Considering that the similarity between documents can be measured by how many terms they have in common, long documents will have more terms in common with other documents than short documents, so they will be more similar to other documents than short documents. To contrast this tendency, weights of terms for TC tasks are usually normalized so that document vectors have unitary length.

2.3 Classification Methods

This dissertation concerns methods for the classification of natural language text, that is, methods that, given a set of training documents with known categories and a new document, which is usually called the *query*, will predict the query's category.

In fact, what some of these methods do is to provide an ordering of the training documents by their similarity to the query. Based on the ordering of the training documents, it is possible to determine an ordered list of the classes that the document belongs to (by determining the training document's classes and removing duplicates). If it is a single-label problem, the class of the query is the first on the list; if it is a multi-label problem, the query belongs to the classes above a

certain similarity threshold, which needs to be tuned for each problem.

This section describes the methods that are compared in this dissertation. Baeza-Yates and Ribeiro-Neto [Baeza-Yates and Ribeiro-Neto, 1999] provide a description of most Information Retrieval methods, and Sebastiani [Sebastiani, 2002] contributed a more up-to-date survey of machine learning methods used for TC.

2.3.1 Vector Method

Historically, the Vector method [Salton and Lesk, 1968; Salton, 1971] was one of the first methods to be applied in Information retrieval tasks. In the Vector method, documents and queries are represented as a set of weighted index terms, that is, vectors in a p-dimensional space, where p is the total number of index terms.

Based on the weights of its terms, each document can be ranked by decreasing order of similarity to the query. The similarity of each document d_j to the query q is computed as the cosine of the angle formed by the vectors that represent each of them

$$sim(\overrightarrow{d_j}, \overrightarrow{q}) = \frac{\overrightarrow{d_j} \cdot \overrightarrow{q}}{||\overrightarrow{d_j}|| \times ||\overrightarrow{q}||} \tag{2.7}$$

The category of the query can be determined as the category of the most similar document found.

2.3.2 k-Nearest Neighbors

The initial application of k-Nearest Neighbors (k-NN) to *text categorization* was reported by Masand and colleagues [Creecy et al., 1996; Masand et al., 1992]. The

basic idea is to determine the category of a given query based not only on the document that is nearest to it in the document space, but on the categories of the k documents that are nearest to it. Having this in mind, the Vector method can be viewed as an instance on the k-NN method, where $k = 1$.

This work uses a vector-based, distance-weighted matching function, as did Yang [Yang, 1994; Yang and Liu, 1999], by calculating document's similarity like the Vector method. Then, it uses a voting strategy to find the query's class: each retrieved document contributes a vote for its class, weighted by its similarity to the query. The query's possible classifications will be ranked according to the votes they got in the previous step.

2.3.3 Naive Bayes

Bayesian or probabilistic classifiers [Lewis, 1998] have been widely used for text categorization. They use the joint probabilities of words and classes to estimate the probabilities of each class given a document.

Given a set of r document vectors $D = \{\overrightarrow{d_1}, \ldots, \overrightarrow{d_r}\}$, classified along a set C of q classes, $C = \{c_1, \ldots, c_q\}$, Bayesian classifiers estimate the probabilities of each class c_k given a document d_j as:

$$P(c_k|\overrightarrow{d_j}) = \frac{P(c_k)P(\overrightarrow{d_j}|c_k)}{P(\overrightarrow{d_j})} \tag{2.8}$$

In this equation, $P(\overrightarrow{d_j})$ is the probability that a randomly picked document has vector $\overrightarrow{d_j}$ as its representation, and $P(c_k)$ the probability that a randomly picked document belongs to c_k. Because the number of possible documents $\overrightarrow{d_j}$ is very high, the estimation of $P(\overrightarrow{d_j}|c_k)$ is problematic.

To simplify the estimation of $P(\overrightarrow{d_j}|c_k)$, Naive Bayes assumes that the probability of a given word or term is independent of other terms that appear in the same document. While this may seem an over simplification, in fact Naive Bayes presents results that are very competitive with those obtained by more elaborate methods. Moreover, because only words and not combinations of words are used as predictors, this naive simplification allows the computation of the model of the data associated with this method to be far more efficient than other non-naive Bayesian approaches. Using this simplification, it is possible to determine $P(\overrightarrow{d_j}|c_k)$ as the product of the probabilities of each term that appears in the document. So, $P(\overrightarrow{d_j}|c_k)$, where $\overrightarrow{d_j} = (w_{1j}, \ldots, w_{|T|j})$, may be estimated as:

$$P(\overrightarrow{d_j}|c_k) = \prod_{i=1}^{|T|} P(w_{ij}|c_k) \tag{2.9}$$

One of the best-known approaches to Naive Bayes classification is the binary independence classifier [Robertson and Jones, 1976], which uses binary-valued vector representations for documents. In this case, by abbreviating $P(w_{ix} = 1|c_k)$ by p_{ik}, the $P(w_{ij}|c_k)$ of Equation (2.9) may be written as

$$P(w_{ij}|c_k) = p_{ik}^{w_{ij}}(1 - p_{ik})^{1-w_{ij}} = \left(\frac{p_{ik}}{1 - p_{ik}}\right)^{w_{ij}}(1 - p_{ik}) \tag{2.10}$$

One may further observe that in TC the document space is partitioned into two categories, c_k and its complement $\overline{c_k}$, such that $P(\overline{c_k}|\overrightarrow{d_j}) = 1 - P(c_k|\overrightarrow{d_j})$. By plugging Equations (2.9) and (2.10) into Equation (2.8) and taking logs:

$$logP(c_k|\overrightarrow{d_j}) = logP(c_k) + \sum_{i=1}^{|T|} w_{ij} log\frac{p_{ik}}{1 - p_{ik}} + \sum_{i=1}^{|T|} log(1 - p_{ik}) - logP(\overrightarrow{d_j}) \tag{2.11}$$

$$log(1 - P(c_k|\overrightarrow{d_j})) = log(1 - P(c_k)) + \sum_{i=1}^{|T|} w_{ij} log \frac{p_{i\bar{k}}}{1 - p_{i\bar{k}}} + \sum_{i=1}^{|T|} log(1 - p_{i\bar{k}}) - log P(\overrightarrow{d_j}) \tag{2.12}$$

where $p_{i\bar{k}}$ means $P(w_{ix} = 1|\overline{c_k})$. It is possible to convert Equations (2.11) and (2.12) into a single equation by subtracting component-wise (2.12) from (2.11), thus obtaining

$$log \frac{P(c_k|\overrightarrow{d_j})}{1 - P(c_k|\overrightarrow{d_j})} = log \frac{P(c_k)}{1 - P(c_k)} + \sum_{i=1}^{|T|} w_{ij} log \frac{p_{ik}(1 - p_{i\bar{k}})}{p_{i\bar{k}}(1 - p_{ik})} + \sum_{i=1}^{|T|} log \frac{1 - p_{ik}}{1 - p_{i\bar{k}}} \tag{2.13}$$

Note that $\frac{P(c_k|\overrightarrow{d_j})}{1-P(c_k|\overrightarrow{d_j})}$ increases monotonically with $P(c_k|\overrightarrow{d_j})$. Also, $log \frac{P(c_k)}{1-P(c_k)}$ and $\sum_{i=1}^{|T|} log \frac{1-p_{ik}}{1-p_{i\bar{k}}}$ are constant for all documents and may thus be ignored. So, if the goal of a classifier is to rank documents in order of their probability of belonging to each class, then it is possible to approximate the value of $P(c_k|\overrightarrow{d_j})$ as:

$$P(c_k|\overrightarrow{d_j}) \approx \sum_{i=1}^{|T|} w_{ij} log \frac{P_{ik}(1 - P_{i\bar{k}})}{P_{i\bar{k}}(1 - P_{ik})} \tag{2.14}$$

where $|T|$ is the number of terms that exist in D, w_{ij} is the weight of term t_i in document d_j, P_{ik} is the probability that term t_i appears in class c_k, and $P_{i\bar{k}}$ is the probability that term t_i appears in classes different from c_k. The document belongs to the class which has the higher probability, $argmax_{c_k}(P(c_k|\overrightarrow{d_j}))$.

Naive Bayes is a very popular method in the TC area, and several authors have already presented results using it [Lewis and Ringuette, 1994; Joachims, 1998a; Koller and Sahami, 1997; Larkey and Croft, 1996; Robertson and Harding, 1984].

2.3.4 Centroid-based Methods

Centroid-based methods combine documents represented using the well known vector-space method [Salton, 1989], to find a representation for a "prototype" document that summarizes all the known documents for a given class, which is called the centroid.

Given a set of r document vectors $D = \{\overrightarrow{d_1}, \ldots, \overrightarrow{d_r}\}$, classified in one of a set C of q classes, $C = \{c_1, \ldots, c_q\}$, D_{c_k}, for $1 \leq k \leq q$, is used to represent the set of document vectors belonging to class c_k.

The centroid of a particular class c_k is represented by a vector $\overrightarrow{c_k}$, which is a combination of the document vectors $\overrightarrow{d_j}$ belonging to that class, sometimes combined with information about vectors of documents that are not in that class. There are several ways to calculate this centroid during the training phase, and several proposals for centroid-based methods exist in the literature. Each proposal uses one possible way of calculating the centroids. The most common proposals are:

- The *Rocchio* formula, where each centroid, $\overrightarrow{c_k}$, is represented by the average of all the document vectors for the positive training examples for class c_k, minus the average of all the vectors for the negative training examples, weighted by control parameters β and γ[1], respectively. This method will be referred to as C-Rocchio and the centroid of class c_k is determined as:

$$\overrightarrow{c_k} = \beta \cdot \frac{1}{|D_{c_k}|} \cdot \sum_{\overrightarrow{d_j} \in D_{c_k}} \overrightarrow{d_j} - \gamma \cdot \frac{1}{|D - D_{c_k}|} \cdot \sum_{\overrightarrow{d_j} \notin D_{c_k}} \overrightarrow{d_j} \tag{2.15}$$

  The application of this method to TC was first proposed by Hull [Hull, 1994] and it has been used in other works where the role of negative examples is

[1] I use β and γ here for consistency with previously published work, but these weights are independent of the ones that appeared in the definition of term distributions in section 2.2.2.

deemphasized, by setting β to a higher value than γ (usually $\beta = 16$ and $\gamma = 4$) [Cohen and Singer, 1999; Ittner et al., 1995; Joachims, 1997].

- The *average* formula [Han and Karypis, 2000; Shankar and Karypis, 2000], where each centroid, $\overrightarrow{c_k}$, is represented by the average of all the vectors for the positive training examples for class c_k. This method will be referred to as C-Average and the centroid of class c_k is determined as:

$$\overrightarrow{c_k} = \frac{1}{|D_{c_k}|} \cdot \sum_{\overrightarrow{d_j} \in D_{c_k}} \overrightarrow{d_j} \tag{2.16}$$

- The *sum* formula [Chuang et al., 2000], where each centroid, $\overrightarrow{c_k}$, is represented by the sum of all the vectors for the positive training examples for class c_k. This method will be referred to as C-Sum and the centroid of class c_k is determined as:

$$\overrightarrow{c_k} = \sum_{\overrightarrow{d_j} \in D_{c_k}} \overrightarrow{d_j} \tag{2.17}$$

- The *normalized sum* formula [Lertnattee and Theeramunkong, 2004; Tan et al., 2005a,b], where each centroid, $\overrightarrow{c_k}$, is represented by the sum of all the vectors for the positive training examples for class c_k, normalized so that it has unitary length. This method will be referred to as C-NormSum and the centroid of class c_k is determined as:

$$\overrightarrow{c_k} = \frac{1}{\| \sum_{\overrightarrow{d_j} \in D_{c_k}} \overrightarrow{d_j} \|} \cdot \sum_{\overrightarrow{d_j} \in D_{c_k}} \overrightarrow{d_j} \tag{2.18}$$

It is fairly obvious that these ways of calculating each class's centroid make centroid-based methods very efficient during the training phase, because there is little computation involved, unlike other methods, which build a more sophisticated model of the data.

Centroid-based methods also have the advantage that they are very easy to

modify in order to perform incremental learning during their training phase, as shall be shown in Section 6.3.

During the classification phase, each test document (or query) is represented by its vector, $\overrightarrow{d_j}$, and is compared to each of the class's centroids $\overrightarrow{c_k}$. Like for the Vector method, the document will be classified as belonging to the class to whose centroid it has the greatest cosine similarity:

$$sim(\overrightarrow{d_j}, \overrightarrow{c_k}) = \frac{\overrightarrow{d_j} \cdot \overrightarrow{c_k}}{||\overrightarrow{d_j}|| \times ||\overrightarrow{c_k}||} \tag{2.19}$$

Centroid-based methods are very efficient during the classification phase because time and memory spent are proportional to the number of classes that exist, rather than to the number of training documents as is the case for the Vector method and other related methods.

2.3.5 Latent Semantic Indexing

Matching documents and queries solely based on index terms can be misleading, because a document can be relevant for a query without having any terms in common with it.

The idea behind the Latent Semantic Indexing method (LSI) [Furnas et al., 1988; Deerwester et al., 1990] is to map each document and query vector into a lower dimensional space which is associated with concepts and retrieve the documents in this space. Arguably, retrieval effectiveness in this space will be better and it will also be computationally less costly, because it is a lower dimensional space.

LSI starts with a term-by-document rectangular matrix X which is decomposed by singular value decomposition (SVD) into the product of three other

matrices: $X = T_0 S_0 D_0$, such that T_0 and D_0 have orthonormal columns and S_0 is diagonal. T_0 and D_0 are the matrices of *left* and *right singular vectors* and S_0 is the diagonal matrix of *singular values*. If the singular values in S_0 are ordered by size (and the corresponding row and column permutations applied to T_0 and D_0), the first largest k may be kept and the remaining ones set to zero. The product of the resulting matrices is a matrix $\hat{X}$ which is only approximately equal to X and is of rank k. It can be shown that the new matrix $\hat{X}$ is the matrix of rank k which is closest in the least squares sense to X. Ideally, the value of k should be large enough to fit all the real structure in the data, but small enough so that the data is not overfit.

After these transformations, the result can still be represented geometrically by a spatial configuration in which the cosine between vectors representing a document and a query corresponds to their similarity.

As in the Vector method, documents can now be ranked according to their similarity to the query, and the category of the query is the category of the most similar document.

2.3.6 Support Vector Machines

The Support Vector Machines (SVM) method or large margin classifier was introduced by Vapnik [Vapnik, 1995; Cortes and Vapnik, 1995] and was first applied for *text categorization* by Joachims [Joachims, 1998a, 1999a]. A thorough description can also be found in other sources [Cristianini and Shawe-Taylor, 2000; Burges, 1998].

The SVM method is a method for efficiently training linear classifiers. This technique is based on recent advances in statistical learning theory. They map the documents into a high dimensional feature space, and try to learn a separat-

ing hyperplane, that provides the widest margins between two different types of documents. SVM use Lagrange multipliers to translate the problem of finding this hyperplane into an equivalent quadratic optimization problem for which efficient algorithms exist, and which are guaranteed to find the global optimum:

$$\text{minimize} \qquad -\sum_{i=1}^{n} \alpha_i + \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j \overrightarrow{d_i} \cdot \overrightarrow{d_j} \tag{2.20}$$

$$\text{such that} \qquad \sum_{i=1}^{n} \alpha_i y_i = 0 \qquad \text{and} \qquad \forall_i \alpha_i \geq 0 \tag{2.21}$$

The set of coefficients α_i^* resulting from the optimization process can then be used to construct the hyperplane that correctly classifies all the training examples with the maximum margin:

$$\overrightarrow{w} \cdot \overrightarrow{d} = \sum_{i=1}^{n} \alpha_i^* y_i (\overrightarrow{d_i} \cdot \overrightarrow{d}) \qquad \text{and} \qquad b = \frac{1}{2} (\overrightarrow{w} \cdot \overrightarrow{d_\circ} + \overrightarrow{w} \cdot \overrightarrow{d_\bullet}) \tag{2.22}$$

This equation shows that the resulting weight vector of the hyperplane is constructed as a linear combination of the training examples. Only examples for which the coefficient α_i^* is grater than zero contribute. These are called the *support vectors*, because they have minimum distance to the hyperplane. Figure 2.1 illustrates these ideas.

For sets of documents that are not linearly separable, the SVM method uses *convolution functions* (or kernels, $K(d_i, d_j)$) instead of the internal product between the two vectors. These kernels transform the initial feature space into another one, by means of a non-linear mapping Φ, where the transformed documents are linearly separable and the SVM method is able to find the hyperplane that separates the documents with the widest margin. In fact, it is not necessary to explicitly calculate $\Phi(d_i) \cdot \Phi(d_j)$, if the kernel function can be used to directly calculate this value. To use a kernel function, one simply substitutes every occurrence of

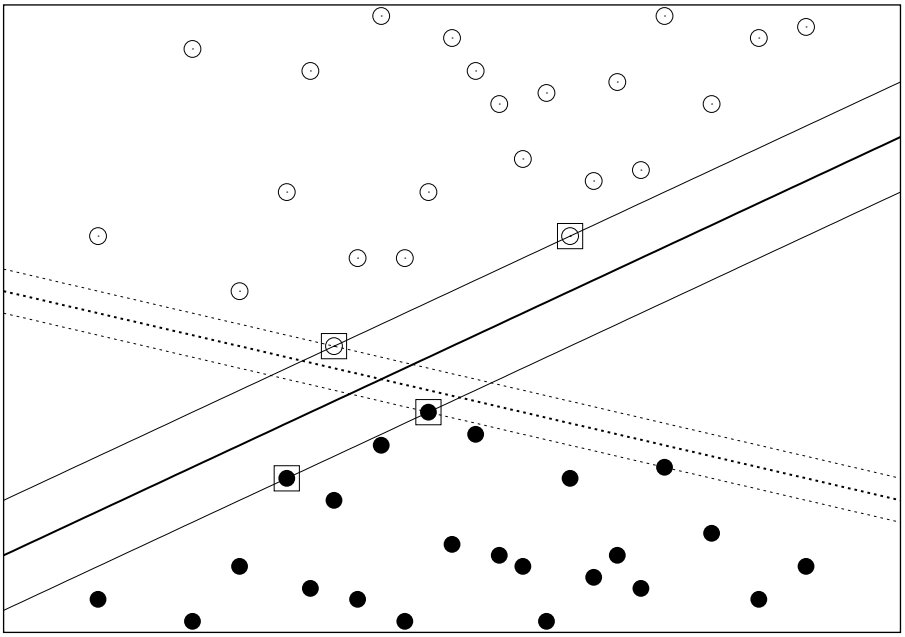

Figure 2.1: Example of a two class, linearly separable problem and two possible separation hyperplanes with corresponding margins. The decision hyperplane chosen by the SVM method is the bold solid line, which corresponds to the largest possible separation margins. The squares indicate the corresponding support vectors.

the inner product in Equations (2.20) and (2.22) with the desired kernel function. These now become:

$$\text{minimize} \qquad -\sum_{i=1}^{n}\alpha_i + \frac{1}{2}\sum_{i,j=1}^{n}\alpha_i\alpha_j y_i y_j * K(d_i, d_j) \tag{2.23}$$

$$\text{such that} \qquad \sum_{i=1}^{n}\alpha_i y_i = 0 \qquad \text{and} \qquad \forall_i \alpha_i \geq 0 \tag{2.24}$$

and the resulting classifier is given by:

$$\overrightarrow{w}\cdot\overrightarrow{d} = \sum_{i=1}^{n}\alpha_i^* y_i K(d_i, d) \qquad \text{and} \qquad b = \frac{1}{2}(K(w, d_\circ) + K(W, \overrightarrow{d_\bullet})) \tag{2.25}$$

By including several classifiers, these ideas can easily be generalized for collections with more than two classes of documents, which is usually done in a one-against-one or one-against-all approach [B. Schölkopf, 1999]. In the one-

against-one approach, if there are q classes, build $q(q-1)/2$ classifiers, using the documents from each combination of two different classes; in the one-against-all approach, build q classifiers, one for each class, using the examples from that class and merging all the others to form the other "class". In this second cases, use a voting strategy to determine the class of a query document. The one-against-one approach builds more classifiers, but each classifier has less examples to train on than the one-against-all approach. The one-against-one approach has been shown to be more suitable for practical use [Hsu and Lin, 2001].

As argued by Joachims [Joachims, 1998b], the SVM method presents two important advantages for TC: (1) term selection is often not needed, as SVM tend to be fairly robust to overfitting and can scale up to large datasets; (2) there is no need to perform parameter tuning on a validation set, because there is a theoretically motivated, default choice of parameter settings, which has also been shown to provide the best effectiveness.

Extensive experimental comparisons [Joachims, 1998b; Yang and Liu, 1999; Cardoso-Cachopo and Oliveira, 2003; Liu et al., 2005] are unanimous in reporting that, among the several classification methods available today, the SVM method is highly competitive in classification Accuracy and can therefore be considered the state-of-the art in text categorization.

2.4 Evaluation Metrics

Evaluating the performance of computational systems is often done in terms of the resources (time and space) they need to operate, assuming that they perform the task that they are supposed to.

Text Classification systems are supposed to classify a query document, by as-

sociating with it an ordered list of categories to which the query belongs. Obviously, it is not enough to classify a document as belonging to any set of categories in a reasonable amount of time. The categories should also be the "right" ones, that is, the ones that the document in fact belongs to. Measures based on Precision and Recall, which take into account if the predicted categories are the right ones, like F1 or PRBP, have been widely used to compare the performance of TC methods [Sebastiani, 2002].

Precision is defined as the fraction of the retrieved documents that are relevant, and can be viewed as a measure of the system's soundness, that is:

$$Precision = \frac{\#\text{Relevant Retrieved Documents}}{\#\text{Retrieved Documents}} \tag{2.26}$$

Recall is defined as the fraction of the relevant documents that is actually retrieved, and can be viewed as a measure of the system's completeness, that is:

$$Recall = \frac{\#\text{Relevant Retrieved Documents}}{\#\text{Relevant Documents}} \tag{2.27}$$

As Berger et al [Berger et al., 2000] already pointed out, these measures are not adequate for evaluating tasks where a single answer is required, in particular because Recall does not make sense when considering only a single answer. The point is that, if a document belongs to a class, it can be correctly classified or not, which means that the contingency tables with False Positives and False Negatives used to calculate Precision and Recall are not adequate. So, to evaluate single-label TC tasks, these measures are not adequate.

2.4.1 Accuracy

Accuracy, which is defined as the percentage of correctly classified documents, is generally used to evaluate single-label TC tasks (see, for instance, [Nigam et al., 2000; Han and Karypis, 2000; Chuang et al., 2000; Han et al., 2001; Lertnattee and Theeramunkong, 2004; Sebastiani, 2005]). Usually, Accuracy is represented as a real value between 0 and 1.

$$Accuracy = \frac{\text{\#Correctly classified documents}}{\text{\#Total documents}} \tag{2.28}$$

As Calado already pointed out [Calado, 2004], it can be shown that, in single-label classification tasks,

$$Accuracy = microaveraged\ F1 = microaveraged\ Precision = microaveraged\ Recall$$

This is because each document can be correctly classified or not, so the number of False Positives is the same as the number of False Negatives.

One of the problems of the Accuracy evaluation measure that is frequently pointed out is that data collections are skewed and that, because of this, it is relatively easy to find a good classifier that simply predicts the most frequent class in the training set. It is possible to overcome this problem by providing, along with each dataset, the results that this kind of dumb classifier would achieve, and by evaluating other classifiers having this value in mind as a baseline.

Ling et al [Ling et al., 2003] proved that the evaluation measure Area Under the ROC Curve (AUC) is statistically consistent and more discriminating than Accuracy, but only for balanced two-class datasets. To the best of my knowledge, there is no further work that extends these results to datasets with unbalanced class distribution and multiple classes, as is the case for the datasets used in this

dissertation.

2.4.2 Mean Reciprocal Rank

Text Categorization systems are supposed to classify a query document, by associating with it an ordered list of categories to which the query belongs. In single-label TC tasks, the goal is to get the right category, and to get it as close to the top of the list as possible. In this case, a measure that takes the rank of the first correct answer in the list of possible categories into account will be more informative than simple Accuracy, which only takes into account if the first category returned by the system was right or wrong.

The Mean Reciprocal Rank (MRR), is a measure that was used to evaluate submissions to the "Question Answering Track" of the TREC conferences, originally defined by Voorhees [Voorhees, 1999]. The idea is that its value is higher if the rank of the first correct answer is lower. The MRR has several advantages:

1. It is closely related to the average precision measure used extensively in Information Retrieval.

2. It is bounded between 0 (worst) and 1 (best), inclusive, and averages well.

3. A system is penalized for not retrieving any correct category for a query document, but not unduly so.

4. It is intuitive and easy to calculate.

The MRR can be calculated for each individual query document as the reciprocal of the rank at which the first correct category was returned, or 0 if none of the first n choices contained the correct category. The score for a sequence of classification queries, considering the first n choices, is the mean of the individual

query´s reciprocal ranks. So:

$$MRR(n) = \frac{\sum_{i=1}^{\#\text{Total queries}}((\frac{1}{rank_i}) \text{ or } 0)}{\#\text{Total queries}} \tag{2.29}$$

where $rank_i$ is the rank of the first correct category for query i, considering the first n categories returned by the system.

By considering MRR(1), that is, by looking just at the first answer, the results are the same as with Accuracy.

$$Accuracy = \frac{\#\text{Correctly classified queries}}{\#\text{Total queries}} = MRR(1) \tag{2.30}$$

$MRR(n)$ with $n > 1$ will increase as long as one is willing to be more permissive, that is, to analyze more answers given by the system. That is, if one is willing to look not at only the first answer given by the classification system but at the first five, how much will the results improve? And what if one looks at the first ten?

So, MRR is in fact a generalization of Accuracy, where it is possible to decide how "sloppy" a system can be. If "hard" classification is intended, use $MRR(1)$, which is Accuracy; if "soft" classification is intended, use $MRR(n)$, where n expresses how "soft" the classification can be. The "softness" to consider depends on the application. For example, if one is building a system that helps a human operator answer new email messages based on answers previously given to similar messages, one might expect the operator to be willing to look at some additional answers besides the first, if the likelihood of finding a more similar response is likely to improve. The point is that MRR provides a simple and intuitive generalization of the standard Accuracy measure for single-label classification tasks, and that additional information can be obtained by looking at more

than one $MRR(n)$ value.

Shah and Croft [Shah and Croft, 2004] have also argued that for some applications achieving high precision in the top document ranks is very important and use as evaluation measure the Mean Reciprocal Rank of the first relevant result.

2.4.3 Micro and Macro Averaging

The measures above can be calculated over the entire collection, which is called *micro-averaging*, or for each class and then across all the classes, which is called *macro-averaging*.

In micro-averaging, each document counts the same for the average, and small classes have a small impact on final results; in macro-averaging, first the average for each class is determined, and then each class counts the same for the final average.

This difference is particularly important when the collection is skewed, that is, when there are classes with very different numbers of documents.

2.4.4 Statistical Significance

According to Hull [Hull, 1993], an evaluation study is not complete without some measurement of the significance of the differences between retrieval methods, and statistical significance tests provide this measurement. These tests can be extremely useful because they provide information about whether observed differences in evaluation scores are really meaningful or simply due to chance. It is not possible to make such a distinction for an individual query, but when evaluation measures are averaged over a number of queries, one can obtain an estimate of the error associated with that measure and significance tests become applicable.

The preliminary assumption, or null hypothesis H_0, will be that all the retrieval methods being tested are equivalent in terms of performance. The significance test will attempt to disprove this hypothesis by determining a p-value, a measurement of the probability that the observed difference could have occurred by chance. Prior to the experiment, a significance level α is chosen, and if the p-value is less than α, one can conclude that the search methods are significantly different. A smaller value of α (which is often set to 0.05) implies a more conservative test. Alternatively, the p-value can merely be viewed as an estimate of the likelihood than two methods are different. All significance tests make the assumption that the queries are independent (i.e. not obviously related). However, slight violations of this assumption are generally not a problem.

Several alternatives exist for evaluating the statistical significance of the results obtained to compare several IR methods, but the most widely used is the t-test. Sanderson and Zobel [Sanderson and Zobel, 2005] argue that the t-test is highly reliable (more so than the sign or Wilcoxon test), and is far more reliable than simply showing a large percentage difference in effectiveness measures between IR systems. They show that both a relative improvement in measured effectiveness and statistical significance are required for confidence in results.

Cormack and Lynam [Cormack and Lynam, 2007] also examine the validity and power of the t-test, Wilcoxon test, and sign test in determining whether or not the difference in performance between two IR systems is significant. They show that these tests have good power, with the t-test proving superior overall.

2.5 Combinations of Methods for Text Categorization

The fundamental idea behind the combination of different classifier methods is the idea of creating a more accurate classifier via some combination of the outputs

of the contributing classifiers. This idea is based on the intuition that, because different methods classify data in different ways, the appropriate combination might improve the results obtained by each individual method.

Classifiers can be combined in different ways, the most common of which will be described in the next paragraphs, along with some of the research work that was done regarding that type of combination.

The simplest way to create a combined classifier is to use several classification methods and a procedure that selects the best method to use in different situations. This can be viewed as a simplified instance of the next methodology, where the best classifier for a particular situation weights 1 and the others 0.

The second way to create a combined classifier is to use several classification methods and a procedure that takes into account the results from each method and weights them to produce the final result.

Hull et al [Hull et al., 1996] compare combination strategies for document filtering and find that simple averaging strategies improve performance.

Larkey and Croft [Larkey and Croft, 1996] combine k-NN, Relevance Feedback, and a Bayesian classifier and show that a combination of classifiers produced better results than any single classifier.

Lam and Lai [Lam and Lai, 2001] employ a meta-learning phase using document feature characteristics, which are derived from the training document set and capture some inherent category-specific properties of a particular category. Their system can automatically recommend a suitable algorithm for each category based on these characteristics. Their results show that their system demonstrates a better performance than existing methods.

Al-Kofahi et al [Al-Kofahi et al., 2001] use various sources of information

about the documents to produce a combined classifier that improves the results obtained by each individual classifier.

Tsay et al [Tsay et al., 2003] study the development of multiple classifier systems in Chinese TC by the strengths of well-known classifiers and show that their approach significantly improves the classification accuracy of individual classifiers for Chinese TC as well as for web page classification.

Bennett et al [Bennett et al., 2005, 2002] introduce a probabilistic method for combining classifiers that considers the context-sensitive reliabilities of contributing classifiers, using variables that provide signals about the performance of classifiers in different situations. They show that their approach provides robust combination schemes across a variety of performance measures.

Bi et al [Bi et al., 2007] and Bell et al [Bell et al., 2005] investigate the combination of four machine learning methods for TC using Dempster´s rule of combinations and show that the best combined classifier can improve the performance of the individual classifiers, and Dempster´s rule of combination outperforms majority voting in combining multiple classifiers.

Finally, a combined classifier can be generated by applying each classification method to the results of the previous ones, such as in boosting procedures.

Weiss et al [Weiss et al., 1999] use the Reuters collection to show that adaptive resampling techniques can improve decision-tree performance and that relatively small, pooled local dictionaries are effective. They apply these techniques to online banking applications to enhance automated e-mail routing.

Schapire and Singer [Schapire and Singer, 2000] introduce the use of a Machine Learning technique called boosting to TC. The main idea of boosting is to combine many simple and moderately inaccurate categorization rules into a single, highly accurate categorization rule. The simple rules are trained sequen-

tially; conceptually, each rule is trained on the examples which were most difficult to classify by the preceding rules. They show, using a number of evaluation measures, that their system´s performance is generally better than individual algorithms, sometimes by a wide margin.

2.6 Semi-supervised Text Categorization

A characteristic common to many TC applications is that it is expensive to classify data for the training phase, while it is relatively inexpensive to find unlabeled data. In other situations, only a small portion of the document space is available initially, and new documents arrive incrementally. A paradigmatic example of this is the classification of web pages: while it is very easy to find unclassified pages in the web, it is slow and labor intensive to get a set of classified pages; moreover, pages in the web are always changing.

It has been shown that the use of large volumes on unlabeled data in conjunction with small volumes of labeled data can greatly improve the performance of some TC methods [Nigam et al., 2000]. The combination of the information contained in the labeled and unlabeled data is done using Expectation-Maximization (EM) [Dempster et al., 1977].

EM is a class of iterative algorithms for maximum likelihood estimation of hidden parameters in problems with incomplete data. In TC, the labels of the unlabeled documents are considered as unknown and EM is used to estimate these (unknown) labels. The general algorithm for the incorporation of unlabeled data using EM is presented in Algorithm 2.1.

EM has been used to combine labeled and unlabeled data for classification in conjunction with several different methods: Shahshahani and Landgrebe [Shahsha-

Inputs: A set of labeled documents, and a set of unlabeled documents.

Initialization step:

- Build an initial classifier from the labeled documents only.

Estimation step:

- Use the current classifier to estimate the class of each unlabeled document.

Maximization step:

- Re-estimate the classifier, given the class predicted for each unlabeled document in the previous step.

Iterate:

- Iterate the estimation and maximization steps until convergence.

Outputs: A classifier that takes the unlabeled documents into account and that, given a new document, predicts its class.

Algorithm 2.1: General algorithm for the incorporation of unlabeled data using EM.

hani and Landgrebe, 1994] use a mixture of Gaussians; Miller and Uyar [Miller and Uyar, 1997] use mixtures of experts; McCallum and Nigam [McCallum and Nigam, 1998] use pool-based active learning; and Nigam [Nigam et al., 2000] uses Naive Bayes. EM has also been used with k-means [MacQueen, 1967], but for clustering rather than for classification, under the name of constrained k-means [Banerjee et al., 2005; Bilenko et al., 2004].

An alternative approach to semi-supervised classification that does not use EM is Transductive Support Vector Machines [Joachims, 1999b], where the SVM method is extended with the ability to use unlabeled documents to improve the results that would be achieved using only the labeled documents. Several authors have reported that this is also a very promising technique [Fung and Mangasarian, 2001; Sindhwani and Keerthi, 2006].

2.7 Incremental Text Categorization

Incremental learning is used in situations where only a small portion of the document space is available initially, or where the documents are expected to evolve over time. According to Sebastiani [Sebastiani, 2002, page 24], "online (a.k.a. incremental) methods build a classifier soon after examining the first training document, and incrementally refine it as they examine new ones. This may be an advantage in the applications where the training set is not available in its entirety from the start, or in which the meaning of the category may change in time."

Algorithm 2.2 details the general algorithm for the incremental incorporation of unlabeled data.

The incremental approach is very suited for tasks that require continuous learning, because the available data changes over time. Once more, a paradig-

Inputs: A set of labeled documents, and a set of unlabeled documents U.

Initialization step:

- Build an initial classifier from the labeled documents only.

Iterate:
For each unlabeled document $d_j \in U$:

- Use the current classifier to classify d_j.
- Update the model of the data obtained using this classifier with the new document d_j classified in the previous step.

Outputs: A classifier that takes the unlabeled documents into account and that, given a new document, predicts its class.

Algorithm 2.2: General algorithm for the incremental incorporation of unlabeled data.

matic example of this is the classification of web pages, because new web pages appear on a daily basis and the ones that already exist are constantly being updated.

2.8 Datasets

Datasets are collections of pre-classified documents. They are essential to develop and evaluate a TC system, that is, to train the system and then to test how well it behaves, when given a new document to classify.

A dataset consists of a set of documents, along with the category or categories that each document belongs to. In a first step, called the *training phase*, some of these documents (called the *training documents*) are used to train the TC system, by allowing it to learn a model of the data. Afterwards, in a step called the *test phase*, the rest of the documents (called the *test documents*) are used to test the TC system, to see how well the system behaves when classifying previously unseen documents.

To allow a fair comparison between several TC systems, it is desirable that they are tested in equivalent settings. With this goal in mind, several data collections were created and made public, generally with a standard train/test split, so that the results obtained by the different systems can be correctly compared.

Some of the publicly available collections are more used than others. In the TC field, and in the single-label sub-field in particular, the most commonly used collections are the 20-Newsgroups collection, the Reuters-21578 collection, and the Webkb collection.

2.8.1 The 20-Newsgroups Collection

The 20-Newsgroups collection is a set of approximately 20,000 newsgroup documents, partitioned (nearly) evenly across the 20 different newsgroups. The articles are typical postings and thus have headers including subject lines, signature files, and quoted portions of other articles.

2.8.2 The Reuters-21578 Collection

All the documents contained in the Reuters-21578 collection appeared on the Reuters newswire and were manually classified by personnel from Reuters Ltd. This collection is very skewed, with documents very unevenly distributed among different classes. The Mod Apté train/test split is generally used for classification tasks [Sebastiani, 2002].

2.8.3 The Webkb Collection

The Webkb collection contains webpages from computer science departments collected by the World Wide Knowledge Base (Web->Kb) project of the CMU text

learning group in 1997. For each of the different classes, the collection contains pages from four universities (Cornell, Texas, Washington and Wisconsin), and other miscellaneous pages collected from other universities.

2.8.4 The Cade Collection

The documents in the Cade collection correspond to web pages extracted from the CADÊ Web Directory, which points to Brazilian web pages classified by human experts.

2.8.5 Pre-Processing

It is widely accepted that the way that documents and queries are represented influences the quality of the results that can be achieved. With this fact in mind, there are several proposals that aim at improving retrieval results.

The main aim of pre-processing the data is to reduce the problem's dimensionality by controlling the size of the system's vocabulary (the number of different index terms). In some situations, aside from reducing the complexity of the problem, this pre-processing will also make the data more uniform in a way that improves performance.

Some of the pre-processing techniques routinely used in TC are:

- Discard words too short or too long, based on the assumption that they probably will not be meaningful.
- Remove numbers and non-letter characters, or substitute them for a special token.
- Case and special character unification. Special character unification is needed

in some languages which contain special characters, because some people use them as they type and some do not.

- Stemming, that is, reducing words to their morphological root, so that the number of different terms in the documents is reduced.

2.9 Summary and Conclusions

Text Categorization is a research area that has provided efficient, effective, and working solutions that have been used in a variety of application domains.

Two of the reasons for this success have been the involvement of the Machine Learning community in TC, which has resulted in the use of the very latest Machine Learning technology in TC applications, and the availability of standard data collections, which has encouraged research by providing a setting in which different research efforts can be compared to each other, so that the best methods can be discovered.

Currently, TC research is pointing in several interesting directions. One of them is to try to improve existing classifiers, by improving their effectiveness, or by making them faster to train or to test.

Another is the attempt to find better representations for text. While the bag of words model is still the most widely used text representation model, some researchers still think that a text is something more than a mere collection of tokens, and that it is worth to continue trying to find more sophisticated models to represent text in the documents [Koster and Seutter, 2003].

A further direction is investigating the scalability properties of TC systems, that is, understanding whether the systems that have proven the best in terms of effectiveness alone can deal with very large numbers of categories [Yang et al.,

2003].

Finally, researchers are realizing that labeling examples for training a text classifier when labeled examples do not previously exist is expensive. As a result, there is increasing attention in TC to semi-supervised Machine Learning methods, that is, methods that can train with a small set of labeled examples and use unlabeled examples to improve their model of the data [Nigam et al., 2000].

This dissertation is concerned with the first (improving the effectiveness of existing classifiers) and last (requiring less labeled documents) of these directions.

Chapter 3

Experimental Setup

This chapter presents the experimental setup used for the experiments performed during this work, namely the classification methods, the evaluation metrics and the datasets that were used.

3.1 Classification Methods

These experiments used existing implementations for some of the Text Classification methods that were compared. These implementations are freely available, and can be obtained from the authors.

A Sourceforge project called IGLU[1] was used for the Vector method described in Section 2.3.1. IGLU aims at being a software platform suitable for testing Information Retrieval methods. At the time of this writing, only the Vector method is implemented.

FAQO — Frequently Asked Questions Organizer [Caron, 2000] was used for the LSI method described in section 2.3.5. FAQO is an application that was de-

[1] Available at http://sourceforge.net/projects/iglu-java.

signed to help the technical support team at Unidata (University Corporation for Atmospheric Research) in the task of answering questions posed by the users of their software. It uses LSI to find similarities between the user's questions and questions that were previously answered by Unidata's personnel. As a result, FAQO shows a ranked list of previous questions and answers that are most similar to the present one. FAQO is an open source project released under the GPL license.

A library called LIBSVM [Chang and Lin, 2001] was used for the SVM method described in Section 2.3.6. LIBSVM, which is an integrated software for Support Vector classification, already supports classification over several different classes, returning, for each document, the (one) class that it belongs to. To calculate the MRR used in this work, however, a ranked list of possible classes for each document is needed, as the one that is returned by the other methods used in this work. So, to determine the class of a given document, a "voting strategy" was implemented, where a document's possible classes are ranked according to the number of votes that they had in a one-against-one approach, as Chang and Lin did [Chang and Lin, 2001]. In this work, a linear kernel was used.

The Naive Bayes method was implemented as described in Section 2.3.3.

For the k-NN method described in Section 2.3.2, I implemented a "voting strategy", where the possible classes of a document are voted on by the documents that belong to that class. Cosine similarity was used as the weight for each vote, and only the 10 nearest documents were considered.

The implementation of the four different centroid-based methods was based on the implementation of the Vector method, and each centroid was determined as described in Section 2.3.4.

3.2 Term Weighting

The *tfidf* term weighting approach described in Section 2.2.1 was already implemented in IGLU, so it was used directly.

The *td* term weighting approach was implemented according to the description in Section 2.2.2, and used exponent factors $\alpha = 0.5$, $\beta = -0.5$, and $\gamma = -0.5$ in the experiments presented in this dissertation. In previous experiments, I combined different values for α, β, and γ and concluded that the ones that provided the best results were the ones used here.

3.3 Evaluation Metrics

As already explained in Section 2.4.2, MRR(1) is the same as Accuracy, the standard evaluation measure for single-label classification tasks. Results are usually reported using three values for MRR (MRR(1), MRR(5), and MRR(10)) instead of only Accuracy because this way it is possible to convey more information. Namely, if one is willing to look not only at the first answer given by the classification system but at the first five, how much will the results improve? And what if one looks at the first ten? For this reason, results are reported using MRR(1), MRR(5), and MRR(10) for the six datasets.

3.4 Datasets

This section describes the work regarding datasets for single-label TC and provides a complete description of the datasets used for these experiments. Except for the Bank collection, all datasets are publicly available from my homepage.[2]

[2]Available at http://www.gia.ist.utl.pt/~acardoso/datasets/.

The creation of these datasets was necessary in order to have different settings to test the rest of this work. It also benefits other researchers in the single-label TC field because it provides easy access to several preprocessed and different-sized datasets for their own experiments.

These experiments use the six datasets that are described here and the standard train/test split for each dataset, so that results can be directly compared to other works that use the same datasets and splits, and because this option allows for more combinations of the numbers of labeled and unlabeled documents, that will be necessary in Chapter 6. When possible, I also use 5-fold cross-validation to test whether the differences observed in the results are statistically significant. Except where explicitly stated otherwise, *tfidf* term weighting is used.

In order to check the validity of the results obtained, 5-fold cross-validation is also performed for most of the methods comparisons.

3.4.1 Creating a New Dataset from the Bank Collection

The Bank collection consists of messages sent by the clients of a financial institution to their help-desk and the answers that the clients got from the help desk assistants. All these documents are in Portuguese.

I had access to the collection of answered messages under a non-disclosure agreement, developed a classification that allows the separation of messages according to the type of request that was made by the client, and classified the messages according to it. This collection contains one class for each type of message that can be answered automatically and one class that comprises all the messages that need human intervention. The complete collection has 1391 classified messages and their respective answers. There are 37 different classes, containing from 5 to 346 messages each. This collection was randomly split into two thirds of the

documents for training and the remaining third for testing. The distribution of documents per class for this particular split, which shall be referred to as Bank37, is presented in Table A.1.

3.4.2 Providing Other Single-label Datasets

Several works on single-label TC used the 20-Newsgroups, the Reuters-21578, and the Webkb standard collections as single-label, by ignoring documents that are crossposted to different newsgroups (in 20-Newsgroups), or that have more than one topic (in Reuters-21578), or considering only some of the available classes (in Webkb). The problem with this approach is that different authors have slightly different datasets, and so comparisons will not be completely accurate [Shankar and Karypis, 2000; Han and Karypis, 2000; Han et al., 2001; Nigam et al., 2000; Lertnattee and Theeramunkong, 2004; Ramakrishnan et al., 2005; Nigam et al., 1999].

Having this in mind, I made available on the web a set of datasets for single-label text categorization. The website contains files with the terms obtained by pre-processing these collections, and was made available for three main reasons:

- To allow an easier comparison among different algorithms. Many papers in this area use these collections but report slightly different numbers of terms for each of them. By having exactly the same terms, the comparisons made using these files will be more reliable.
- To ease the work of people starting out in this field. Because these files contain less information than the original ones, they can have a simpler format and thus will be easier to process. The most common pre-processing steps are also provided for some datasets.
- To provide single-label datasets, which the original collections were not.

Group	Group
alt.atheism	rec.sport.hockey
comp.graphics	sci.crypt
comp.os.ms-windows.misc	sci.electronics
comp.sys.ibm.pc.hardware	sci.med
comp.sys.mac.hardware	sci.space
comp.windows.x	soc.religion.christian
misc.forsale	talk.politics.guns
rec.autos	talk.politics.mideast
rec.motorcycles	talk.politics.misc
rec.sport.baseball	talk.religion.misc

Table 3.1: Usenet groups for the 20-Newsgroups collection.

3.4.2.1 20-Newsgroups

The 20-Newsgroups collection was downloaded from Jason Rennie's page and the "Bydate" version was used, because it already had a standard train/test split. This is a collection of approximately 20,000 newsgroup messages, partitioned (nearly) evenly across the 20 different newsgroups mentioned in Table 3.1. Although already cleaned-up, this collection still had several attachments, many PGP keys and approximately 4% of the articles are cross-posted. After removing them and the messages that became empty because of it, the distribution of training and test messages for each newsgroup is presented in Table A.6. From now on, this particular version of the 20-Newsgroups collection shall be referred to as the 20Ng dataset.

3.4.2.2 Reuters-21578

The Reuters-21578 collection was downloaded from David Lewis's page and the standard "Mod Apté" train/test split was used. The documents in this collection appeared on the Reuters newswire in 1987 and were manually classified by personnel from Reuters Ltd.

Due to the fact that the class distribution for these documents is very skewed,

two sub-collections are usually considered for text categorization tasks [Debole and Sebastiani, 2004a]:

- R10 – The set of documents belonging to the 10 classes with the highest number of positive training examples.

- R90 – The set of documents belonging to the 90 classes with at least one positive training and testing example.

Besides being very skewed, many of the documents in this collection are classified as having no topic at all or with more than one topic. Table A.3 shows the distribution of the documents per number of topics. The separation between training, test, and other documents is done considering the standard Mod Apté split.

Because the goal of this work is to consider single-label datasets, all the documents with less than or with more than one topic were eliminated. After this, some of the classes in R10 and R90 were left with no training or test documents. Considering only the documents with a single topic and the classes which still have at least one training and one test example, there are 8 of the 10 most frequent classes and 52 of the original 90. Following Sebastiani´s convention, these sets will be called R8 and R52. From R10 to R8 the classes "corn" and "wheat", which are intimately related to the class "grain", disapeared and this last class lost many of its documents. Similar transformations occurred from R90 to R52.

The distribution of training and test documents per class for R8 is presented in Table A.4 and the distribution of training and test documents per class for R52 is presented in Table A.5.

3.4.2.3 Webkb

The Webkb collection (also called 4 Universities Data Set) was downloaded from the homepage of the World Wide Knowledge Base (WebKb) project of the CMU Text Learning Group.

This collection contains webpages collected from computer science departments of various universities in 1997. The webpages were manually classified into seven categories and contain pages from four universities plus some miscellaneous pages collected from other universities.

As Nigam et Al. [Nigam et al., 2000], the classes "Department" and "Staff" were discarded because there were only a few pages from each university. The class "Other" was also discarded, because pages were very different among the examples for this class.

Because there is no standard train/test split for this collection, and in order to be consistent with the previous collections, two thirds of the documents were randomly chosen for training and the remaining third for testing. The distribution of training and test webpages for each class is presented in Table A.2. From now on, this particular version of the Webkb collection shall be referred to as he Web4 dataset.

3.4.2.4 Cade

The documents in the Cade collection correspond to a subset of web pages extracted from the CADÊ Web Directory, which points to Brazilian web pages classified by human experts. This directory is available at Cade's Homepage, in Brazilian Portuguese.

A preprocessed version of this collection, which is part of project Gerindo, was

made available by Marco Cristo, from Universidade Federal de Minas Gerais, in Brazil.

Two thirds of the documents were randomly chosen for training and the remaining third for testing. The distribution of training and test documents per class for this particular split, which shall be referred to as the Cade12 dataset, is presented in Table A.7.

3.4.3 Statistics of the Datasets

This work uses six different datasets, Bank37, 20Ng, R8, R52, Web4 and Cade12, that originated in five different collections with different kinds of contents, Bank, 20-Newsgroups, Reuters-21578, Webkb and Cade. Both R8 and R52 originated from Reuters-21578.

The datasets have different sizes, ranging from 1391 documents for Bank37, to 40983 documents for Cade12. Also, even though 20Ng is relatively well balanced in the number of documents per class, all the others are very skewed. For example, R52 has classes with as little as three documents, whereas the class "earn", with 3923 documents, contains almost half the total number of documents. Table 3.2 presents the numbers of documents for the six datasets: number of training documents, number of test documents, total number of documents, number of documents in the smallest class, and number of documents in the largest class. The detailed distribution of training and test documents per class for each dataset can be found in Appendix A.

Furthermore, different collections use different languages: 20-Newsgroups, Reuters-21578, and Webkb are in English; Bank is in Portuguese; and Cade is in Brazilian Portuguese. This is important because, if a result holds for all the collections, it probably is language-independent (considering these two languages).

Dataset	Train Docs	Test Docs	Total Docs	Smallest Class	Largest Class
Bank37	928	463	1391	5	346
20Ng	11293	7528	18821	628	999
R8	5485	2189	7674	51	3923
R52	6532	2568	9100	3	3923
Web4	2803	1396	4199	504	1641
Cade12	27322	13661	40983	625	8473

Table 3.2: Numbers of documents for the six datasets: number of training documents, number of test documents, total number of documents, number of documents in the smallest class, and number of documents in the largest class.

In order to complete the evaluations based on the Accuracy and MRR measures, Table 3.3 contains the values of MRR(1), MRR(5), and MRR(10) for each of the datasets that can be achieved by a dumb classifier, that is, a classifier that totally disregards the test documents and always predicts that the class of the query is the most frequent class in the training set. Observe that, because R8 and R52 are very skewed, the Dumb Classifier achieves relatively high values. Having this in mind, it is to be expected that a good classifier for these datasets achieves significantly higher values. Also, note that, because Web4 only has four different classes, MRR(5), and MRR(10) are the same for this dataset.

3.5 Computational Framework — IREP

IREP —*Information Retrieval Experimentation Package*— is a computational framework that integrates the implementations of several classification methods, term weighting schemes, and evaluation measures and uses them with the datasets that are provided.

IREP was built in Java and has a number of features:

- It is able to read and pre-process the documents and pass them on to the different classification methods. IREP adapts the format of the documents to

Dataset	MRR	Dumb Classifier
Bank37	1	0.2505
	5	0.3116
	10	0.3336
20Ng	1	0.0530
	5	0.1208
	10	0.1546
R8	1	0.4947
	5	0.6887
	10	0.6978
R52	1	0.4217
	5	0.5870
	10	0.5975
Web4	1	0.3897
	5	0.6277
	10	0.6277
Cade12	1	0.2083
	5	0.3869
	10	0.4205

Table 3.3: Values of MRR(1), MRR(5), and MRR(10) achieved by the Dumb Classifier for the six datasets.

the format required by each method implementation, guaranteeing that all the methods use exactly the same data. IREP also has filters that implement the pre-processing steps described in Appendix A.6 and the term-weighting schemes described in Section 2.2.

- It is able to use the same evaluation measures with all the classification methods. The results provided by each method are analyzed by IREP, which then computes the evaluation measures required.

- It is easy to incorporate new classification methods. Each classifier must provide two Java methods, one that builds the classifier based on the training data and one that, given the classifier and a document to classify, returns a list with the document´s possible classes. Based on these two Java methods, IREP is able to train, to test, and to evaluate any new classifier.

- It is possible to combine the results obtained by different classifiers. IREP can be used to combine different classifiers by creating a new classifier that

calls the Java methods of each original classifier in a specific order.

- It is easy to change the parameters used by each classifier. IREP has a top-level method that reads and interprets the parameters that can be used by each classification method, and passes them on to the respective method at run time.

- It is able to repeatedly train and test the classifiers and provide the results in an understandable manner. IREP can be called from (for example) a UNIX shell with the required parameters and the results are written to a text file.

IREP is thus a very configurable computational tool that can be used to experiment with existing classifiers and easily extended to include new ones.

Chapter 4

Performance of Existing Text Classification Methods

This chapter presents the results obtained in experimental tests that were conducted in order to access the efficiency of different Text Categorization (TC) methods.

4.1 Comparing Centroid-based Methods

This section reports the results of some experiments, which show that, of the several centroid-based methods that are generally used in the literature, the one that calculates the centroid of a class as the normalized sum of the documents that belong to the class is the one that provides the best results.

A thorough comparison between several centroid-based methods and term weighting schemes is available [Cardoso-Cachopo and Oliveira, 2006], but this work uses more datasets (Bank37, Web4 and Cade12 were not used in previous experiments) and the results are reported in a different way.

The centroid-based methods used in this comparison are the ones described in Section 2.3.4, and the correspondence between the names and respective equations is obvious:

- In C-Rocchio the centroid of a class is the centroid of the positive examples minus the centroid for the negative examples, weighted by control parameters β and γ, respectively (see Equation (2.15)).
- In C-Average the centroid of a class is the average of all the positive examples for that class (see Equation (2.16)).
- In C-Sum the centroid of a class is the sum of all the positive examples for that class (see Equation (2.17)).
- In C-NormSum the centroid of a class is the sum of all the positive training examples for that class, normalized so that it has unitary length (see Equation (2.18)).

Figure 4.1 shows six charts with the values of MRR(1), MRR(5), and MRR(10) for the six datasets, for each of the centroid-based methods. Table 4.1 contains the values that were used to plot the charts. Table 4.2 contains average Accuracy values for the centroid-based methods.

By looking at the charts, it is easy to see that, for all datasets, C-NormSum is the centroid-based method that provides the best results. For the R8 dataset, for example, Accuracy is 0.7172 for the C-Sum method, but it improves to 0.9356 for the C-NormSum method. Table 4.2 supports the same conclusion, showing that the average Accuracy over all the datasets is significantly higher for C-NormSum than for the other centroid-based methods.

Except for the R52 dataset, C-Sum is the worst performing centroid-based method.

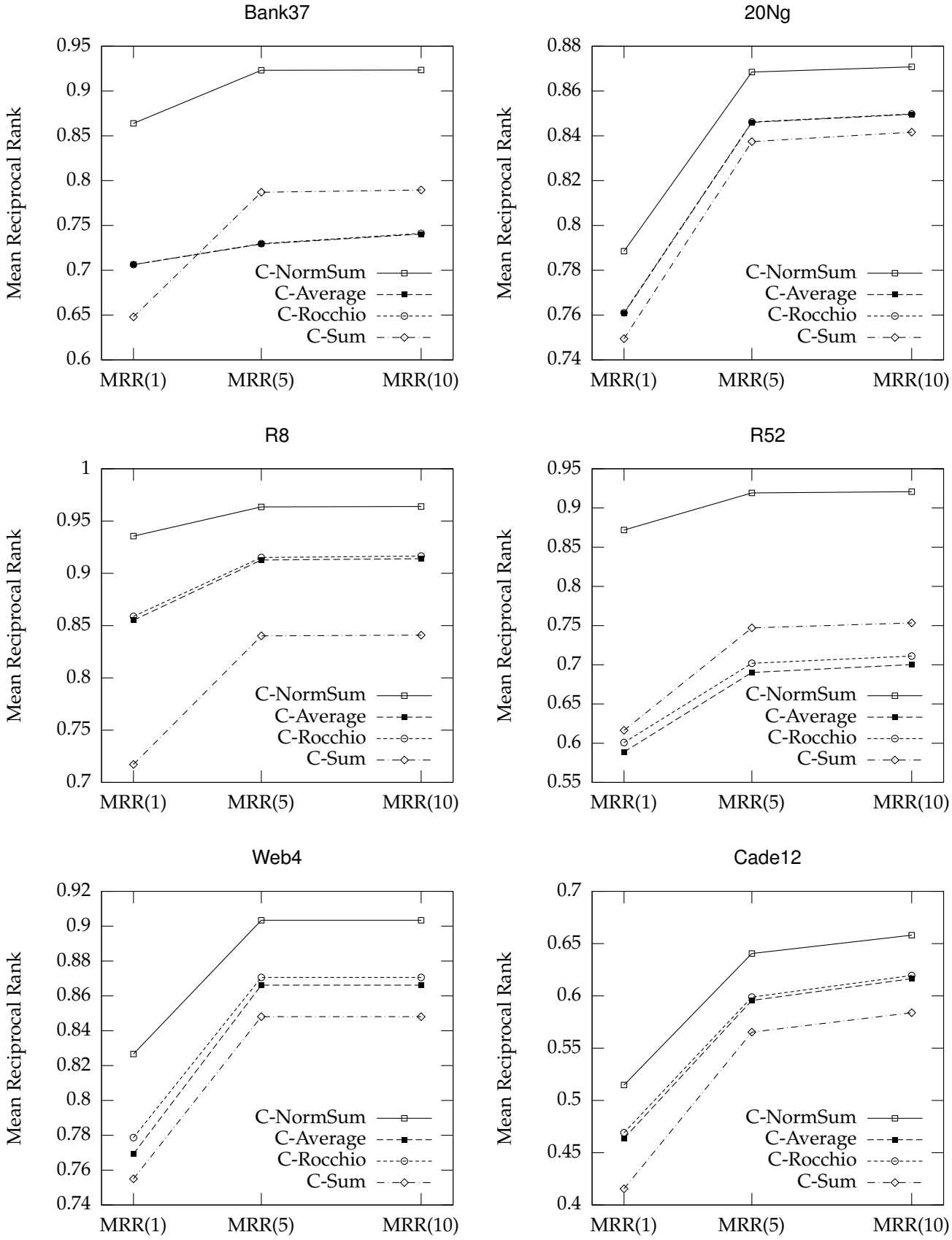

Figure 4.1: MRR(1), MRR(5), and MRR(10) for the six datasets, for each of the centroid-based methods C-NormSum, C-Rocchio, C-Average and C-Sum.

Dataset	MRR	C-NormSum	C-Average	C-Rocchio	C-Sum	Dumb
Bank37	1	0.8639	0.7063	0.7063	0.6479	0.2505
	5	0.9231	0.7293	0.7297	0.7870	0.3116
	10	0.9234	0.7404	0.7412	0.7896	0.3336
20Ng	1	0.7885	0.7608	0.7610	0.7495	0.0530
	5	0.8685	0.8460	0.8462	0.8374	0.1208
	10	0.8708	0.8496	0.8498	0.8416	0.1546
R8	1	0.9356	0.8552	0.8588	0.7172	0.4947
	5	0.9635	0.9127	0.9151	0.8402	0.6887
	10	0.9639	0.9140	0.9165	0.8409	0.6978
R52	1	0.8719	0.5888	0.6009	0.6164	0.4217
	5	0.9192	0.6902	0.7019	0.7470	0.5870
	10	0.9207	0.7003	0.7111	0.7534	0.5975
Web4	1	0.8266	0.7693	0.7787	0.7550	0.3897
	5	0.9034	0.8662	0.8706	0.8481	0.6277
	10	0.9034	0.8662	0.8706	0.8481	0.6277
Cade12	1	0.5148	0.4637	0.4691	0.4155	0.2083
	5	0.6405	0.5956	0.5988	0.5653	0.3869
	10	0.6580	0.6164	0.6195	0.5839	0.4205

Table 4.1: Values of MRR(1), MRR(5), and MRR(10) for the six datasets, for each of the centroid-based methods C-NormSum, C-Rocchio, C-Average and C-Sum. The values obtained by the Dumb Classifier are also included for comparison purposes.

	Accuracy				
Dataset	C-NormSum	C-Average	C-Rocchio	C-Sum	Dumb
Bank37	0.8639	0.7063	0.7063	0.6479	0.2505
20Ng	0.7885	0.7608	0.7610	0.7495	0.0530
R8	0.9356	0.8552	0.8588	0.7172	0.4947
R52	0.8719	0.5888	0.6009	0.6164	0.4217
Web4	0.8266	0.7693	0.7787	0.7550	0.3897
Cade12	0.5148	0.4637	0.4691	0.4155	0.2083
Average	0.8002	0.6907	0.6958	0.6503	0.3030

Table 4.2: Values of Accuracy for each of the six datasets, for the centroid-based methods C-NormSum, C-Average, C-Rocchio, C-Sum, and average Accuracy over all the datasets for each method. Values for the Dumb Classifier are included for comparison purposes.

For all datasets, the results obtained with C-Average are very similar to those obtained with C-Rocchio, because the lines representing their results are always very close together. This result can be verified by comparing average Accuracy for both these methods in Table 4.2. This means that incorporating in the centroid that represents a given class information about all the documents that do not belong to that class does not improve results.

In all cases, results improve significantly between MRR(1) and MRR(5), but they show a small improvement from MRR(5) to MRR(10). This means that, if one is interested in building a system that can suggest more than one classification for each document, it is worth looking at the first five answers given by the system, but looking at the first ten will not provide a significant improvement.

Note also that the range in the Y axis representing MRR is quite different for the different datasets. While for Bank37, 20Ng, R8, R52 and Web4 it is possible to achieve relatively high Accuracy results, for Cade12 classification Accuracy barely reaches 0.5148. Nevertheless, all these centroid-based methods provide significant improvements in Accuracy relatively to the ones obtained by the Dumb Classifier, as can be easily observed in Tables 4.1 and 4.2.

In order to verify if the results obtained previously are statistically significant, 5-fold cross-validation tests were performed using all the datasets and the results were compared with the results obtained by each method, using paired t-tests. Table 4.3 contains the values of Accuracy for each of the six datasets, for the centroid-based methods, for each of the 5 folds, and average Accuracy over all the folds for each dataset and method.

Because the differences in the results across all the datasets were statistically significant ($p < 0.05$), it is possible to conclude that, for the four centroid-based methods that were used,

C-NormSum $\neq$ C-Average ($p = 0.0000002$)

		Accuracy			
Dataset	Fold	C-NormSum	C-Average	C-Rocchio	C-Sum
Bank37	1	0.8705	0.7014	0.7014	0.6439
	2	0.8705	0.7122	0.7122	0.6295
	3	0.8022	0.6835	0.6835	0.5827
	4	0.8705	0.6799	0.6799	0.6079
	5	0.8495	0.6559	0.6559	0.6344
	mean	0.8526	0.6866	0.6866	0.6197
20Ng	1	0.8568	0.8262	0.8262	0.8002
	2	0.8488	0.8116	0.8114	0.7936
	3	0.8387	0.8172	0.8170	0.8002
	4	0.8385	0.8135	0.8130	0.7978
	5	0.8470	0.8197	0.8186	0.8011
	mean	0.8460	0.8177	0.8172	0.7986
R8	1	0.8963	0.8318	0.8416	0.7458
	2	0.9036	0.8384	0.8469	0.7355
	3	0.9055	0.8248	0.8339	0.7674
	4	0.9114	0.8306	0.8404	0.7349
	5	0.9075	0.8371	0.8450	0.7550
	mean	0.9049	0.8326	0.8415	0.7477
R52	1	0.8495	0.5659	0.5846	0.6335
	2	0.8571	0.6121	0.6258	0.6429
	3	0.8489	0.6038	0.6203	0.6533
	4	0.8401	0.5835	0.5956	0.6165
	5	0.8456	0.5967	0.6088	0.6396
	mean	0.8482	0.5924	0.6070	0.6371
Web4	1	0.8486	0.7902	0.7914	0.7330
	2	0.8250	0.7964	0.7976	0.7548
	3	0.8071	0.7810	0.7869	0.7607
	4	0.8393	0.7905	0.7929	0.7512
	5	0.8298	0.7821	0.7857	0.7726
	mean	0.8300	0.7880	0.7909	0.7545
Cade12	1	0.5192	0.4786	0.4813	0.4269
	2	0.5008	0.4569	0.4592	0.4119
	3	0.5034	0.4602	0.4640	0.4042
	4	0.5154	0.4663	0.4710	0.4136
	5	0.4965	0.4460	0.4500	0.4044
	mean	0.5071	0.4616	0.4651	0.4122
Average	mean	0.7981	0.6965	0.7014	0.6616

Table 4.3: Values of Accuracy for each of the six datasets, for the centroid-based methods C-NormSum, C-Average, C-Rocchio, C-Sum, for each of the 5 folds, and average Accuracy over all the datasets for each method.

C-NormSum $\neq$ C-Rocchio ($p = 0.0000002$)

C-NormSum $\neq$ C-Sum ($p = 0.0000000$)

C-Average $\neq$ C-Rocchio ($p = 0.0000204$)

C-Average $\neq$ C-Sum ($p = 0.0001051$)

C-Rocchio $\neq$ C-Sum ($p = 0.0000093$)

Using the inequalities above, the average across all the datasets in Table 4.3, and the fact that all the inequalities are statistically significant, it is possible to find a total ordering for the quality of the centroid-based methods:
C-NormSum $>$ C-Rocchio $>$ C-Average $>$ C-Sum.

Because the C-NormSum is the best centroid-based method, I shall consider this one from now on, whenever I refer to a centroid-based method.

4.2 Comparing Classification Methods

This section reports the results of the experiments that have shown that the centroid-based method C-NormSum provides results that are almost as good as those obtained by the state-of-the-art SVM method.

Figure 4.2 shows six charts with the values of MRR(1), MRR(5), and MRR(10) for the six datasets, for each of the classification methods. Table 4.4 contains the values that were used to plot the charts. Table 4.5 contains average Accuracy values for the various methods.

From the charts, it is obvious that for all datasets except Cade12, the best method is SVM. This means that the algorithms underlying the SVM method generally work very well on very different datasets, except for the webpages in the Cade12 dataset.

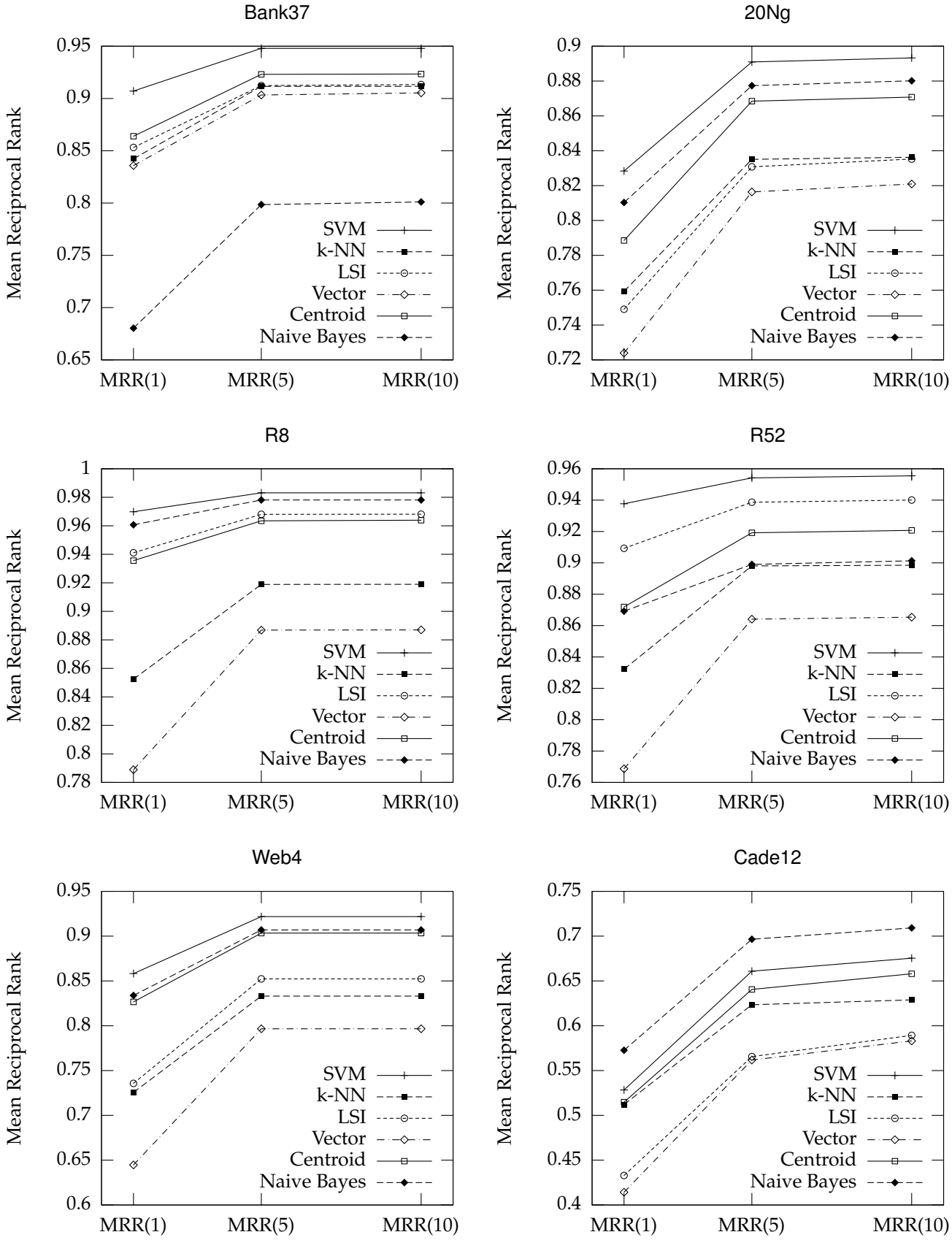

Figure 4.2: MRR(1), MRR(5), and MRR(10) for the six datasets, for each of the classification methods Vector, k-NN, LSI, SVM and Centroid.

Dataset	MRR	Centroid	SVM	N-Bayes	k-NN	LSI	Vector	Dumb
Bank37	1	0.8639	0.9071	0.6803	0.8423	0.8531	0.8359	0.2505
	5	0.9231	0.9479	0.7985	0.9116	0.9123	0.9033	0.3116
	10	0.9234	0.9479	0.8011	0.9116	0.9133	0.9054	0.3336
20Ng	1	0.7885	0.8284	0.8103	0.7593	0.7491	0.7240	0.0530
	5	0.8685	0.8910	0.8774	0.8351	0.8308	0.8164	0.1208
	10	0.8708	0.8933	0.8801	0.8363	0.8353	0.8209	0.1546
R8	1	0.9356	0.9698	0.9607	0.8524	0.9411	0.7889	0.4947
	5	0.9635	0.9831	0.9781	0.9189	0.9681	0.8868	0.6887
	10	0.9639	0.9831	0.9781	0.9190	0.9681	0.8870	0.6978
R52	1	0.8719	0.9377	0.8692	0.8322	0.9093	0.7687	0.4217
	5	0.9192	0.9542	0.8990	0.8981	0.9387	0.8641	0.5870
	10	0.9207	0.9555	0.9013	0.8985	0.9400	0.8654	0.5975
Web4	1	0.8266	0.8582	0.8352	0.7256	0.7357	0.6447	0.3897
	5	0.9034	0.9218	0.9099	0.8331	0.8523	0.7966	0.6277
	10	0.9034	0.9218	0.9099	0.8331	0.8523	0.7966	0.6277
Cade12	1	0.5148	0.5284	0.5727	0.5120	0.4328	0.4142	0.2083
	5	0.6405	0.6609	0.6965	0.6234	0.5656	0.5617	0.3869
	10	0.6580	0.6754	0.7092	0.6289	0.5892	0.5831	0.4205

Table 4.4: Values of MRR(1), MRR(5), and MRR(10) for the six datasets, for each of the classification methods Centroid, SVM, Naive Bayes, k-NN, LSI, and Vector. The values obtained by the Dumb Classifier are also included for comparison purposes.

	Accuracy						
Dataset	Centroid	SVM	N-Bayes	k-NN	LSI	Vector	Dumb
Bank37	0.8639	0.9071	0.6803	0.8423	0.8531	0.8359	0.2505
20Ng	0.7885	0.8284	0.8103	0.7593	0.7491	0.7240	0.0530
R8	0.9356	0.9698	0.9607	0.8524	0.9411	0.7889	0.4947
R52	0.8719	0.9377	0.8692	0.8322	0.9093	0.7687	0.4217
Web4	0.8266	0.8582	0.8352	0.7256	0.7357	0.6447	0.3897
Cade12	0.5148	0.5284	0.5727	0.5120	0.4328	0.4142	0.2083
Average	0.8002	0.8383	0.7881	0.7540	0.7702	0.6961	0.3030

Table 4.5: Values of Accuracy for each of the six datasets, for each of the classification methods Centroid, SVM, Naive Bayes, k-NN, LSI, and Vector, and average Accuracy over all the datasets for each method. The values obtained by the Dumb Classifier are also included for comparison purposes.

For the Cade12 dataset, the best method is Naive Bayes.

The centroid-based method is usually second or third best, and it is always on the "top half" of the chart. This means that, independently of the subject of the dataset, it is expected to show a good performance.

The LSI method performs quite well for the datasets based on the Reuters-21578 collection, R8 and R52, and also for Bank37.

The k-NN method shows a bad performance for all datasets, always among the two or three worst methods.

As expected, the simple Vector method provides a base-line performance for almost all the datasets. The only dataset for which it is not the worst performing method is the Bank37 dataset, where Naive Bayes performs even worse.

By considering average Accuracy values over all the datasets, one can see that the Centroid method is the second best, following the state-of-the-art SVM method. After considering that the Centroid method is very fast, in both training and test phases, as I shall show in the next section, I decided to investigate this method further. Another important advantage of the centroid-based methods is that they require a very small amount of memory to build the model of the data (only one vector to represent each class) during the training and test phases, while others need to have all the training documents in memory at the same time to build the model of the data.

Once more, results improve significantly between MRR(1) and MRR(5), but they show a small improvement from MRR(5) to MRR(10). While for Bank37, R8, R52 and 20Ng it is possible to achieve relatively high Accuracy results, for Cade12 the maximum achievable Accuracy is 0.5727. All these methods provide significant improvements in Accuracy relatively to the ones obtained by the

Dumb Classifier, as can be easily observed in Table 4.4[1].

In order to verify if the results obtained previously are statistically significant, I performed 5-fold cross-validation tests using all the datasets and compared the results obtained by each method using paired t-tests. Table 4.6 contains the values of Accuracy for each of the classification methods Centroid, SVM, Naive Bayes, k-NN, LSI, and Vector, for each of the 5 folds, and average Accuracy over all the folds for each dataset and method. It was not possible to use 5-fold cross-validation for LSI with the Cade12 dataset because FAQO, the tool that was used to implement LSI did not support the required amount of training documents. For comparisons involving LSI only the other five datasets were used.

This time not all results were statistically significant ($p < 0.05$):

C-NormSum $\neq$ SVM ($p = 0.0000000$)

C-NormSum $\neq$ Naive Bayes ($p = 0.3969039$)

C-NormSum $\neq$ k-NN ($p = 0.1473818$)

C-NormSum $\neq$ LSI ($p = 0.0619330$)

C-NormSum $\neq$ Vector ($p = 0.0000000$)

SVM $\neq$ Naive Bayes ($p = 0.0006746$)

SVM $\neq$ k-NN ($p = 0.0000000$)

SVM $\neq$ LSI ($p = 0.0000000$)

SVM $\neq$ Vector ($p = 0.0000000$)

Naive Bayes $\neq$ k-NN ($p = 0.2245246$)

Naive Bayes $\neq$ LSI ($p = 0.0619624$)

Naive Bayes $\neq$ Vector ($p = 0.0032988$)

k-NN $\neq$ LSI ($p = 0.0000343$)

k-NN $\neq$ Vector ($p = 0.0000001$)

[1] I include the values of MRR(1), MRR(5), and MRR(10) obtained by the Dumb Classifier in the table to make comparisons easier. I do not include them in the charts because they are too low, compared to the others, and that would cause the lines for the other methods to be too close together.

				Accuracy			
Dataset	Fold	Centroid	SVM	N-Bayes	k-NN	LSI	Vector
	1	0.8705	0.9245	0.7050	0.8741	0.8597	0.8489
	2	0.8705	0.8921	0.6727	0.8561	0.8381	0.8525
Bank37	3	0.8022	0.8669	0.6439	0.8309	0.8273	0.7878
	4	0.8705	0.9137	0.6691	0.8669	0.8741	0.8525
	5	0.8495	0.9104	0.7025	0.7957	0.8423	0.8100
	mean	0.8526	0.9015	0.6786	0.8448	0.8483	0.8304
	1	0.8568	0.9139	0.8916	0.8818	0.8701	0.8778
	2	0.8488	0.9187	0.8789	0.8677	0.8624	0.8791
20Ng	3	0.8387	0.9152	0.8831	0.8714	0.8696	0.8738
	4	0.8385	0.9083	0.8786	0.8642	0.8608	0.8632
	5	0.8470	0.9201	0.8903	0.8707	0.8704	0.8714
	mean	0.8460	0.9153	0.8845	0.8712	0.8666	0.8731
	1	0.8963	0.9681	0.9511	0.8957	0.9557	0.8383
	2	0.9036	0.9700	0.9440	0.8847	0.9550	0.8371
R8	3	0.9055	0.9759	0.9557	0.9134	0.9603	0.8397
	4	0.9114	0.9661	0.9459	0.8853	0.9505	0.8515
	5	0.9075	0.9739	0.9531	0.8899	0.9492	0.8450
	mean	0.9049	0.9708	0.9500	0.8938	0.9541	0.8423
	1	0.8495	0.9456	0.8769	0.8709	0.9115	0.8165
	2	0.8571	0.9505	0.8780	0.8681	0.9159	0.8379
R52	3	0.8489	0.9544	0.8797	0.8797	0.9275	0.8335
	4	0.8401	0.9593	0.8885	0.8874	0.9286	0.8363
	5	0.8456	0.9440	0.8736	0.8610	0.9143	0.8132
	mean	0.8482	0.9508	0.8793	0.8734	0.9196	0.8275
	1	0.8486	0.8772	0.8355	0.7294	0.7723	0.6508
	2	0.8250	0.8619	0.8488	0.7321	0.7702	0.6690
Web4	3	0.8071	0.8619	0.8083	0.7595	0.7726	0.6702
	4	0.8393	0.8750	0.8464	0.7321	0.7690	0.6440
	5	0.8298	0.8726	0.8369	0.7476	0.7655	0.6464
	mean	0.8300	0.8697	0.8352	0.7402	0.7699	0.6561
	1	0.5192	0.5487	0.5961	0.5264		0.4306
	2	0.5008	0.5312	0.5827	0.5106		0.4043
Cade12	3	0.5034	0.5388	0.5910	0.5137		0.4118
	4	0.5154	0.5380	0.5858	0.5160		0.4145
	5	0.4965	0.5220	0.5724	0.5080		0.3994
	mean	0.5071	0.5357	0.5856	0.5149		0.4121
Average	mean	0.7981	0.8573	0.8022	0.7897	0.8717	0.7402

Table 4.6: Values of Accuracy for each of the six datasets, for each of the classification methods Centroid, SVM, Naive Bayes, k-NN, LSI, and Vector, for each of the 5 folds, and average Accuracy over all the datasets for each method.

LSI $\neq$ Vector ($p = 0.0000009$)

Using the inequalities above, the average across all the datasets in Table 4.6, and the fact that only some of the inequalities are statistically significant, it is possible to find a partial ordering for the quality of the classification methods: SVM $>$ Centroid $\approx$ Naive Bayes $\approx$ k-NN $\approx$ LSI$>$ Vector.

4.3 Comparing Execution Times for the Methods

This section reports the results of the experiments that have shown that centroid-based methods are very fast, both in the training and in the test phase.

Besides the quality of the results that a TC method yields, another important aspect to consider when evaluating TC methods is the time and memory they require to execute.

Figure 4.3 shows six charts, representing the time that each method takes to execute, in both the training and the test phases, for each dataset. The X axis represents the time spent for training, while the Y axis represents the time spent for tests, both in seconds, in logarithmic scales. These times are the average obtained over five runs for each method and each dataset. Table 4.7 contains the values that were used to plot the charts. The values reported for the centroid-based method are for C-NormSum, but they are very similar for all the other centroid-based methods.

By looking at the X axis of Figure 4.3, it is possible to compare the time that each method takes for training. As expected, Naive Bayes is the fastest method. Following this method are Vector and k-NN, because they simply read the training documents and save the term/document matrix for future use. The other

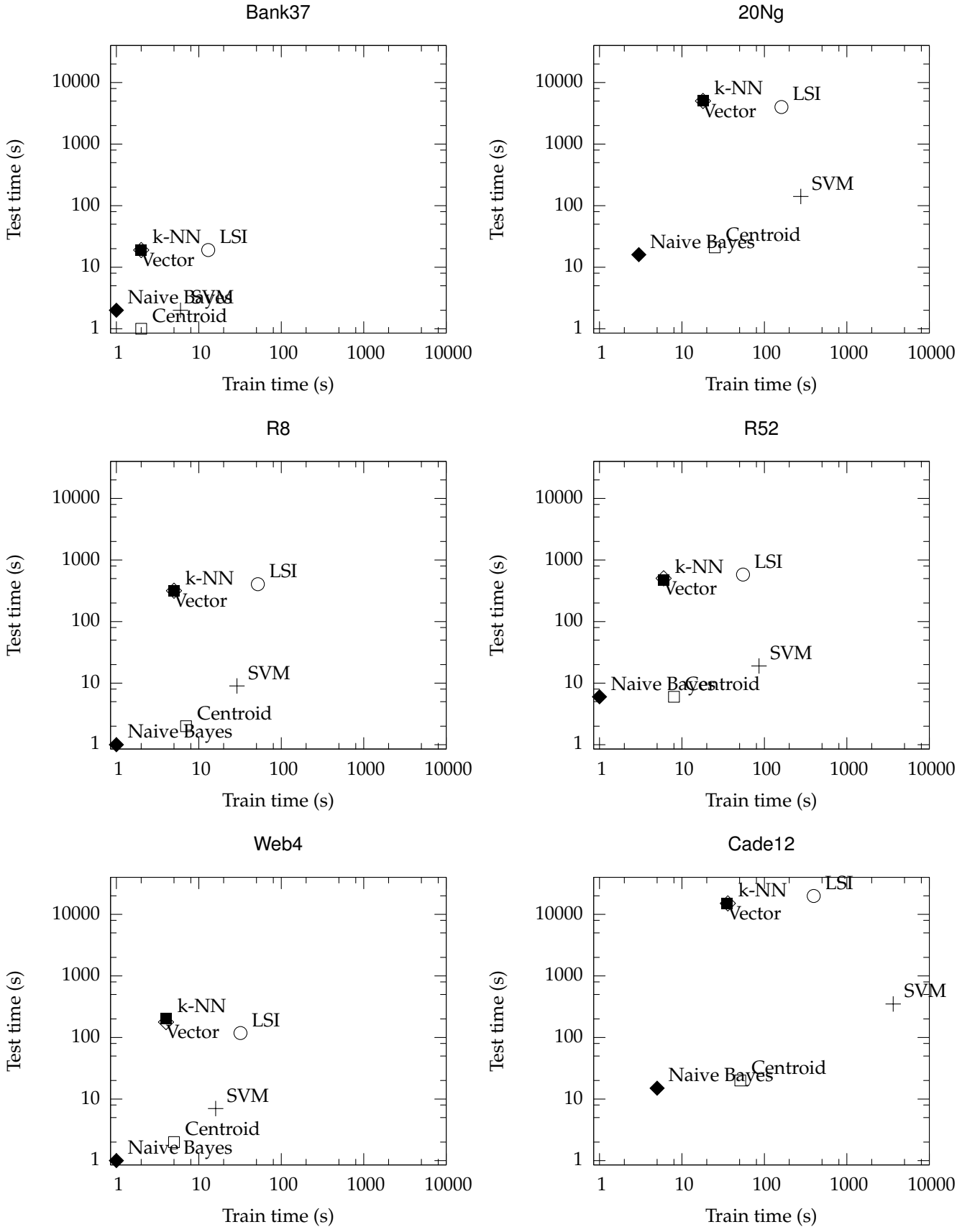

Figure 4.3: Training and Test time spent by each method for each dataset, in seconds, in a logarithmic scale.

Dataset	Time (s)	Centroid	SVM	N-Bayes	k-NN	LSI	Vector
Bank37	Train	2	6	1	2	13	2
	Test	1	2	2	19	19	19
	Total	3	8	3	21	32	21
20Ng	Train	25	276	3	18	161	18
	Test	21	141	16	5049	4002	5016
	Total	47	417	19	5067	4163	5034
R8	Train	7	29	1	5	52	5
	Test	2	9	1	313	405	317
	Total	9	38	2	318	456	322
R52	Train	8	86	1	6	55	6
	Test	6	19	6	474	580	504
	Total	14	105	7	479	635	510
Web4	Train	5	16	1	4	32	4
	Test	2	7	1	202	118	178
	Total	7	23	1	206	150	182
Cade12	Train	51	3644	5	35	396	36
	Test	20	350	15	15188	19898	15088
	Total	71	3994	20	15223	20294	15124

Table 4.7: Training, Test, and Total time spent by each method for each dataset, in seconds.

methods need to build this matrix and then determine a new model of the data. The simplest transformation is done by the Centroid method, because it only calculates the centroid for each class, which is why it is only slightly slower than the other two methods in the training phase. The methods that take longer for training are SVM and LSI, because they are the ones that apply the most complicated transformation to the training documents.

To compare the time that each method takes during the test phase, observe the Y axis of Figure 4.3. Once more, the fastest method is Naive Bayes, very closely followed by the Centroid method. This should be expected, because in the Naive Bayes case it is necessary to calculate a probability for each class and each training document, while in the Centroid case the operation that is required for each test document is a cosine similarity with each class's centroid. The third fastest is SVM. The slowest are Vector and k-NN, because they have to compare each test document to each training document, making test time proportional to

the number of documents in the training set. Using an index structure would speed up these last two methods, but they would still be considerably slower than the Naive Bayes and the centroid-based methods.

Overall, the centroid-based method provides a significant reduction in time and memory required during the test phase relatively to SVM, k-NN and LSI, because these requirements are proportional to the number of classes instead of the number of documents.

Note that none of the methods was particularly optimized for speed, and that probably it is possible to find faster implementations for these classification methods. From this section, however, it is obvious that centroid-based methods are orders of magnitude faster than other classification methods (except Naive Bayes). This was to be expected, not only because the representation of each class is very simple (a single vector), which leads to the use of very small amounts of memory, but also because the operations required to determine each centroid are very simple, and, hence very fast.

4.4 Comparing Term Weighting Schemes

This section reports the results of the experiments that have shown that the traditional *tfidf* term weighting scheme is in general better than the more recently proposed term distributions *td* [Lertnattee and Theeramunkong, 2004].

Figure 4.4 shows six charts with the values of MRR(1), MRR(5), and MRR(10) for the six datasets, for each of the methods, using *tfidf* and *td* term weighting. Table 4.8 contains the values that were used to plot the charts.

By looking at Figure 4.4 it is easy to see that, except for the Centroid method, *td* term weighting always worsens results. The only exception is k-NN for the

Dataset/weighting	MRR	Centroid	SVM	k-NN	Vector
Bank37/*tfidf*	1	0.8639	0.9071	0.8423	0.8359
	5	0.9231	0.9479	0.9116	0.9033
	10	0.9234	0.9479	0.9116	0.9054
Bank37/*td*	1	0.8704	0.9028	0.8445	0.8164
	5	0.9268	0.9419	0.9105	0.8915
	10	0.9271	0.9430	0.9109	0.8932
20Ng/*tfidf*	1	0.7885	0.8284	0.7593	0.7240
	5	0.8685	0.8910	0.8351	0.8164
	10	0.8708	0.8933	0.8363	0.8209
20Ng/*td*	1	0.8038	0.8203	0.7487	0.6603
	5	0.8779	0.8848	0.8236	0.7682
	10	0.8799	0.8875	0.8263	0.7747
R8/*tfidf*	1	0.9356	0.9698	0.8524	0.7889
	5	0.9635	0.9831	0.9189	0.8868
	10	0.9639	0.9831	0.9190	0.8870
R8/*td*	1	0.9351	0.9534	0.7853	0.7346
	5	0.9645	0.9739	0.8819	0.8557
	10	0.9646	0.9740	0.8819	0.8562
R52/*tfidf*	1	0.8719	0.9377	0.8322	0.7687
	5	0.9192	0.9542	0.8981	0.8641
	10	0.9207	0.9555	0.8985	0.8654
R52/*td*	1	0.8843	0.9167	0.7574	0.7029
	5	0.9290	0.9405	0.8504	0.8225
	10	0.9297	0.9415	0.8512	0.8249
Web4/*tfidf*	1	0.8266	0.8582	0.7256	0.6447
	5	0.9034	0.9218	0.8331	0.7966
	10	0.9034	0.9218	0.8331	0.7966
Web4/*td*	1	0.8474	0.8281	0.7364	0.6282
	5	0.9164	0.9027	0.8420	0.7851
	10	0.9164	0.9027	0.8420	0.7851
Cade12/*tfidf*	1	0.5148	0.5284	0.5120	0.4142
	5	0.6405	0.6609	0.6234	0.5617
	10	0.6580	0.6754	0.6289	0.5831
Cade12/*td*	1	0.4545	0.3899	0.3695	0.2874
	5	0.5891	0.5347	0.5037	0.4567
	10	0.6104	0.5597	0.5107	0.4847

Table 4.8: Values of MRR(1), MRR(5), and MRR(10) for each dataset and each method, using *tfidf* and *td* term weighting.

Bank37 and Web4 datasets, where MRR(1) shows a very small improvement from 0.8423 to 0.8445 and from 0.7256 to 0.7364, respectively.

The only classification method for which *td* usually improves results is the Centroid method, exactly the one that was used to propose this term weighting scheme [Lertnattee and Theeramunkong, 2004]. Even for the Centroid method, *td* worsens results for the Cade12 and R8 datasets. This is probably due to the

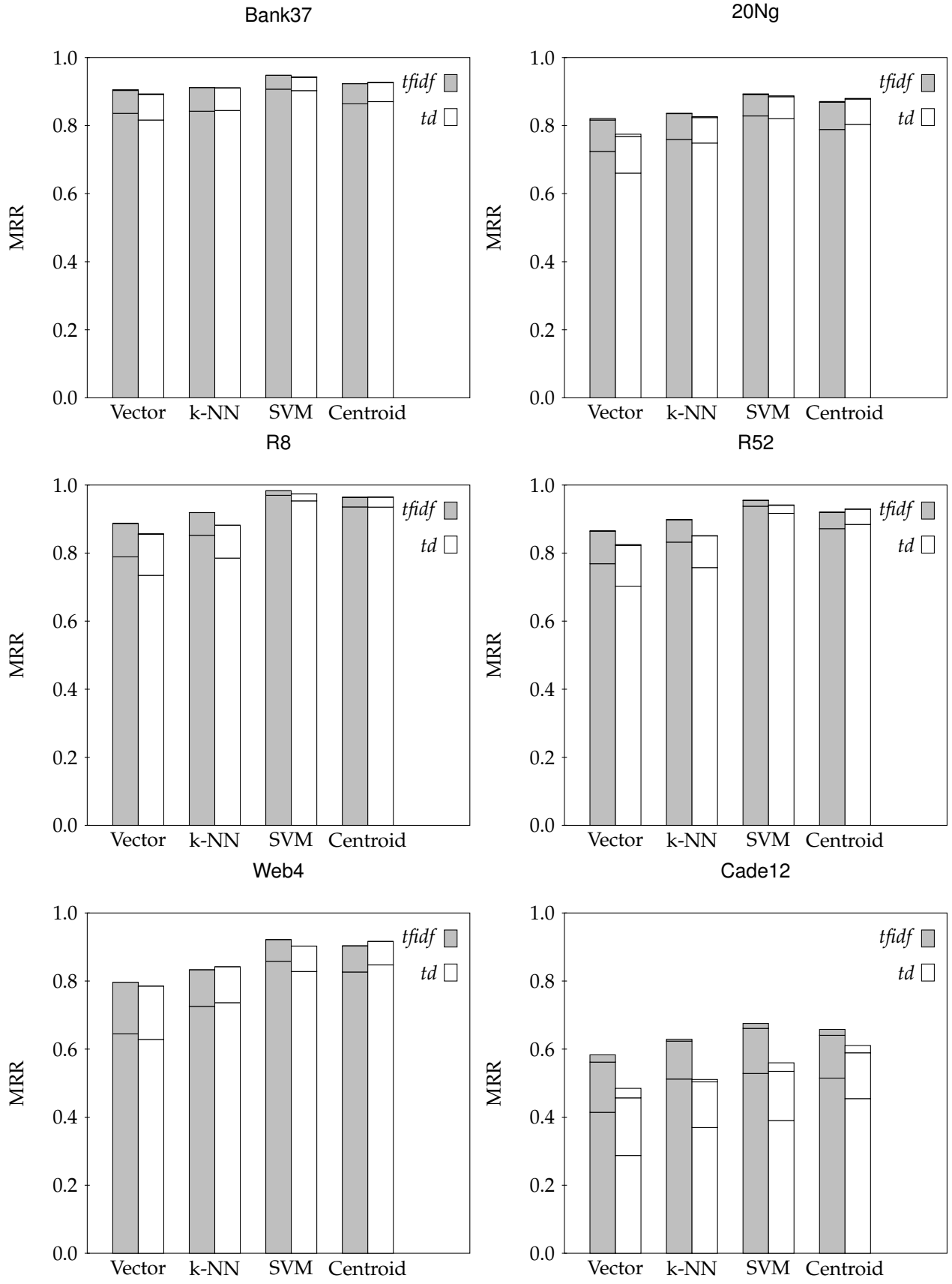

Figure 4.4: MRR(1), MRR(5), and MRR(10) for each dataset and method, using *tfidf* and *td* term weighting.

fact that these collections are skewed and so term distributions, which depend on the classes, are bad predictors.

Besides the fact that the *td* term weighting scheme only marginally improves results for one method and for some datasets relatively to the traditional *tfidf* term weighting scheme, the *td* approach also has the disadvantage that there are three different weights that can be combined and that need to be optimized. I performed extensive testing combining different values for α, β and γ for the different classification methods and determined that, on average, the best results were obtained when the three weights were set to zero, that is, when *tfidf* term weighting was used. Based on my previous experiments and on the results presented in this section, I conclude that the traditional *tfidf* term weighting scheme is in general better than the more recently proposed term distributions *td* and will use *tfidf* in the rest of my work.

Once more, it is easy to observe that: Accuracy or MRR(1) accounts for a good part of the correct answers given by the system, independently of the classification method and of the dataset that is considered; that results improve significantly between MRR(1) and MRR(5); and that they show a very small improvement from MRR(5) to MRR(10).

4.5 Summary of Experimental Results

Regarding the different centroid-based methods, it was shown that C-NormSum always outperforms other centroid-based approaches for these datasets. Average Accuracy over the six datasets improves from 0.6958 for C-Rocchio, the second best centroid-based method, to 0.8002 when using C-NormSum.

Compared to the Vector and k-NN methods, C-NormSum significantly im-

proves Accuracy for these six datasets, while at the same time approximately maintaining training time and reducing test time by several orders of magnitude.

The results of this work also showed that C-NormSum, a computationally simple and fast method, can obtain results similar to the more sophisticated and computationally expensive state-of-the-art SVM method. This makes the centroid-based method a promising method to perform further research. The difference in average Accuracy over the six datasets is from 0.8002 using C-NormSum to 0.8383 using SVM.

Overall, the C-NormSum method presents a good trade-off between time spent during the training and test phases and the quality of the results obtained. This method also has the advantage of being amenable to changes in the training set, unlike the SVM method.

It was also shown that the traditional *tfidf* term weighting approach remains very effective, even when compared to *td*, a more recent and computationally demanding term weighting approach, because over the six datasets and the four methods that were tested, *tfidf* provided better results in 17 of the 24 possible combinations. Both term weighting approaches achieved similar results in 2 cases.

Chapter 5

Combinations Between Classification Methods

This chapter describes the underlying ideas regarding the combination of classification methods and the experiments that were performed with them.

The combination of k-NN with LSI was first proposed in 2003 [Cardoso-Cachopo and Oliveira, 2003]. The combination of SVM with LSI was first proposed in 2007 [Cardoso-Cachopo and Oliveira, 2007a].

5.1 Document Representation and Classification Strategy

This section describes how different classification methods use different representations for the training documents of a dataset and different classification strategies in order to classify the test documents.

Figure 5.1 shows a graphical representation of the transformations applied to

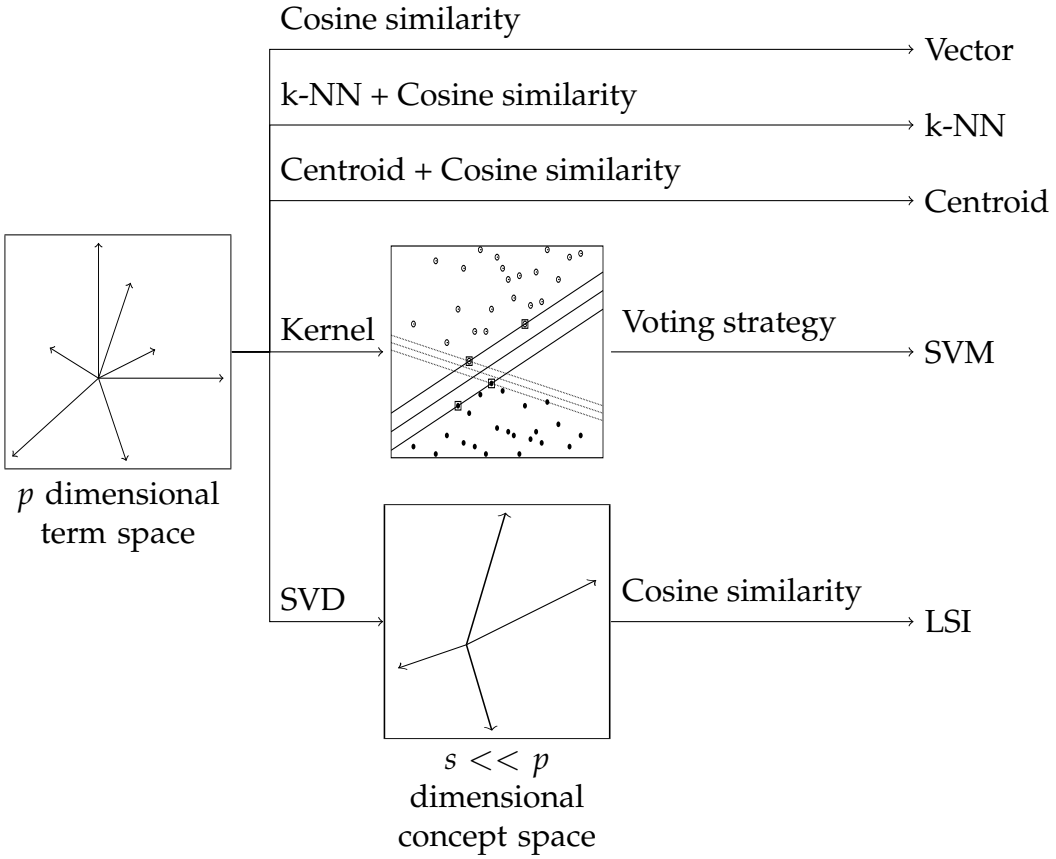

Figure 5.1: Graphical representation of the transformations applied to the initial term space representing the training documents and of the classification strategy used by each classification method.

the initial term space representing the training documents and the classification strategy used by each classification method.

Starting at the left, there is the initial representation of the r training documents of the dataset as p-dimensional vectors in a p-dimensional space, where r is the number of training documents of the dataset and p is the number of different terms considered for the dataset. This is usually represented by a $p \times r$ term/document matrix.

Given another p-dimensional vector representing the query document—that is, the document that needs to be classified—there are several options:

- Apply cosine similarity directly to the query and each of the training doc-

uments; the class of the query is the class of the most similar document — Vector method.

- Apply cosine similarity directly to the query and each of the training documents, select the k most similar documents, apply a voting strategy, where each document "votes" for its class weighted by its similarity to the query; the class of the query is the most voted class — k-NN method.

- Determine the centroid of the documents belonging to each class; the class of the query is determined as the class of the most similar centroid — centroid-based method.

- Apply a kernel function so that training documents are represented in a high dimensional feature space, where each class is linearly separable from the others; apply a voting strategy, where possible classes are ranked according to the number of votes that they had in a one-against-one classification approach; the class of the query is the class which got more votes — SVM method.

- Apply singular value decomposition to the original term/document matrix, transforming the original space into a much smaller concept space; apply to the query document the same transformation applied to the matrix; the class of the transformed query is the class of the most similar transformed document — LSI method.

By considering these representations and classification strategies, I propose the combinations between methods described in the following section.

5.2 Combinations Between Methods

This section describes the rationale behind the combination of the different representations and classification strategies in order to obtain new classification methods, that ideally will perform better than the original ones.

Figure 5.2 shows a graphical representation of the transformations applied to the initial term space representing the training documents and the classification strategy used by the combinations of methods, corresponding to the combination of k-NN and SVM with LSI.

The difference to the previous approaches is that now, instead of applying their transformations to the initial term/document matrix, the methods to combine with LSI apply their transformations to the concept space that was previously obtained using Singular Value Decomposition. Then, given another p-dimensional vector representing the query document, choose one of the options:

- Apply to the query document the same transformation as the one applied to the initial term/document matrix; apply cosine similarity to the transformed query and to each of the transformed training documents; select the k most similar documents; apply a voting strategy, where each transformed document "votes" for its class, weighted by its similarity to the transformed query; the class of the query is the most voted class — k-NN-LSI method.

- Apply a kernel function to the transformed concept matrix, so that concepts are represented in a high dimensional feature space, where each class is linearly separable from the others; apply to the query document the same transformation applied to the initial term/document matrix; apply a voting strategy, where possible classes are ranked according to the number of votes that they had in a one-against-one classification approach; the class of the query is the class which got more votes — SVM-LSI method.

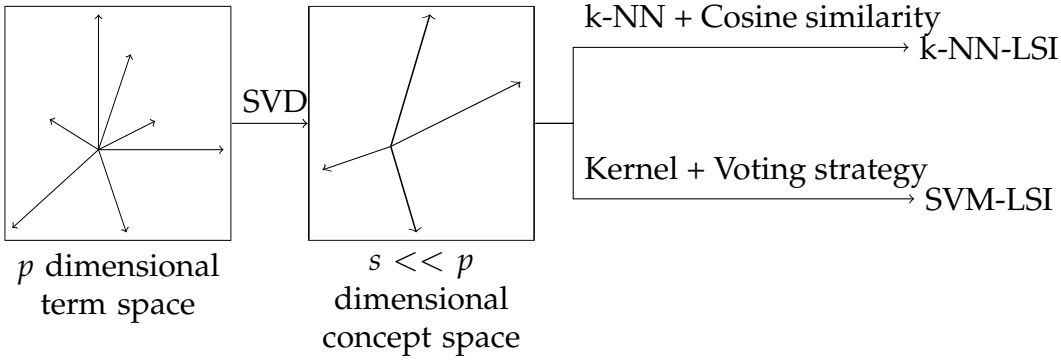

Figure 5.2: Graphical representation of the transformations applied to the initial term space representing the training documents and of the classification strategy used by the combinations of methods.

Dataset	MRR	k-NN	LSI	k-NN-LSI
Bank37	1	0.8423	0.8531	0.8488
	5	0.9116	0.9123	0.9099
	10	0.9116	0.9133	0.9106
20Ng	1	0.7593	0.7491	0.7557
	5	0.8351	0.8308	0.8263
	10	0.8363	0.8353	0.8271
R8	1	0.8524	0.9411	0.9488
	5	0.9189	0.9681	0.9691
	10	0.9190	0.9681	0.9691
R52	1	0.8322	0.9093	0.9100
	5	0.8981	0.9387	0.9365
	10	0.8985	0.9400	0.9367
Web4	1	0.7256	0.7357	0.7908
	5	0.8331	0.8523	0.8744
	10	0.8331	0.8523	0.8744
Cade12	1	0.5120	0.4328	0.4881
	5	0.6234	0.5656	0.5962
	10	0.6289	0.5892	0.6019

Table 5.1: Values of MRR(1), MRR(5), and MRR(10) for the six datasets, for k-NN, LSI and k-NN-LSI.

5.3 Comparing k-NN-LSI with k-NN and LSI

This section reports the results of the experiments performed to analyze the combination of k-NN with LSI.

Figure 5.3 shows six charts with the values of MRR(1), MRR(5), and MRR(10) for the six datasets, for the LSI, k-NN and k-NN-LSI methods. Table 5.1 contains

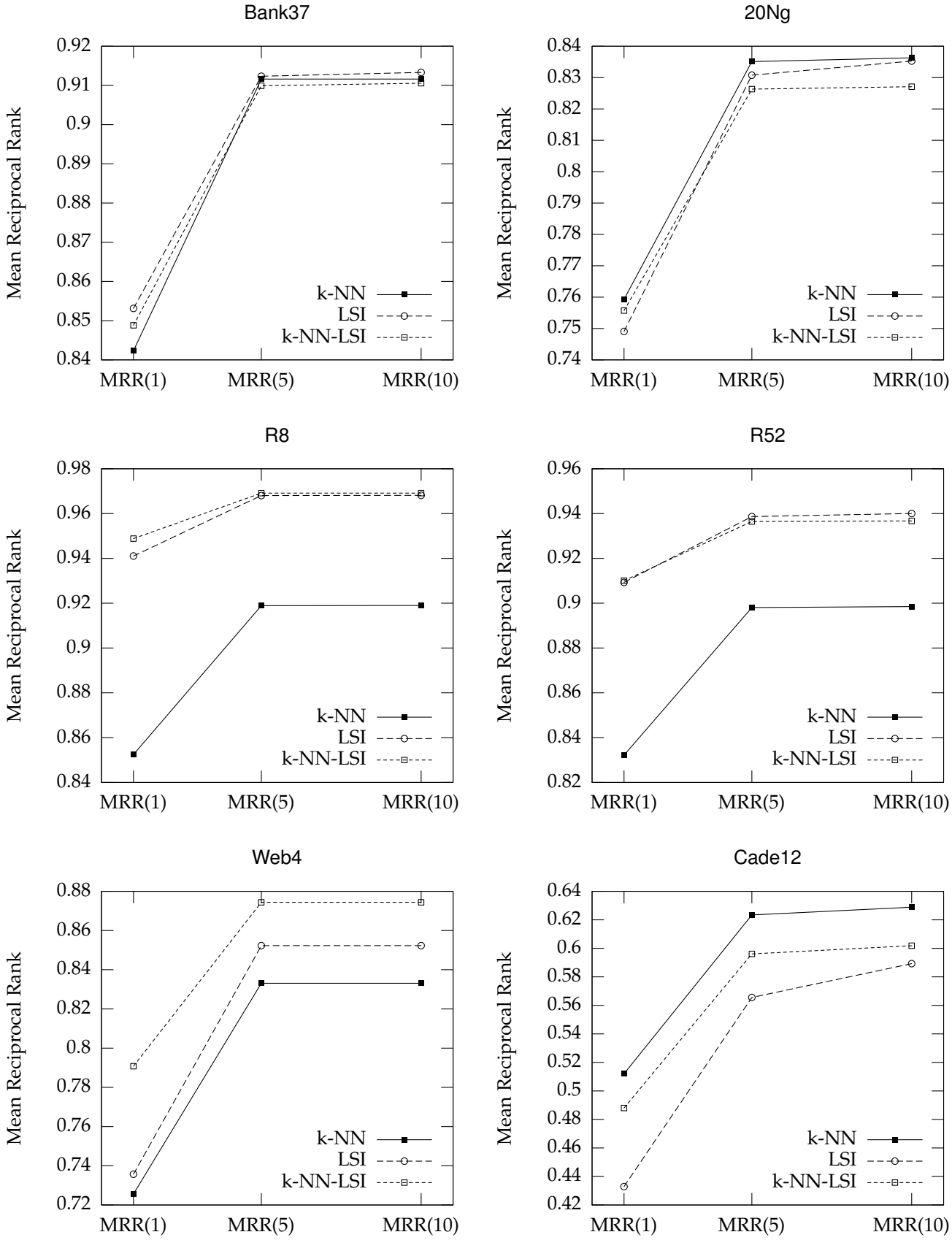

Figure 5.3: MRR(1), MRR(5), and MRR(10) for the six datasets, for k-NN, LSI and k-NN-LSI.

	Accuracy		
Dataset	k-NN	LSI	k-NN-LSI
Bank37	0.8423	0.8531	0.8488
20Ng	0.7593	0.7491	0.7557
R8	0.8524	0.9411	0.9488
R52	0.8322	0.9093	0.9100
Web4	0.7256	0.7357	0.7908
Cade12	0.5120	0.4328	0.4881
Average	0.7540	0.7702	0.7904

Table 5.2: Values of Accuracy for each of the six datasets, for k-NN, LSI and k-NN-LSI, and average Accuracy over all the datasets for each method.

the values that were used to plot the charts.

First, it is important to note that, from the two original methods, there is not one that always outperforms the other. For the four smaller datasets, Bank37, R8, R52 and Web4, LSI performs better than k-NN, whereas for 20Ng and Cade12 it is k-NN that provides better results. This is probably because LSI is very effective at finding the "concepts" in the smaller datasets, but, for the other datasets, which consist in newsgroup messages (that can quote others) and web pages (that can be copies of others), finding the most similar document is more effective.

For R8 and Web4, the combination of k-NN with LSI is the best performing method. For all the other datasets, it is second best, independently of which of the original methods shows a better performance, and its performance is closer to the one achieved by the best method.

As can be seen in Table 5.2, when considering average Accuracy over the six datasets, k-NN-LSI is the best performing method.

In order to verify if the results obtained previously are statistically significant, I performed 5-fold cross-validation tests using all the datasets and compared the results obtained by each method using paired t-tests. Table 5.3 contains the values of Accuracy for each of the classification methods k-NN, LSI, and k-NN-LSI, for each of the 5 folds, and average Accuracy over the five folds and five datasets for

each method. It was not possible to use 5-fold cross-validation for LSI with the Cade12 dataset because FAQO, the tool that was used to implement LSI, did not support the required amount of training documents.

This time, not all results were statistically significant:
k-NN-LSI $\neq$ k-NN ($p = 0.0003036$)
k-NN-LSI $\neq$ LSI ($p = 0.2345841$)

Using the inequalities above, the average across the datasets in Table 5.3, and the fact that only some of the inequalities are statistically significant, it is possible to find a partial ordering for the quality of the classification methods:
k-NN-LSI $\approx$ LSI and
k-NN-LSI $>$ k-NN.

However, for 20Ng and Web4, k-NN-LSI $>$ LSI, $p = 0.0004887$ and $p = 0.0012195$, respectively.

5.4 Comparing SVM-LSI with SVM and LSI

This section reports the results of my experiments regarding the combination of SVM with LSI. The results show that, for some datasets, this combination outperforms the SVM method.

Figure 5.4 shows six charts with the values of MRR(1), MRR(5), and MRR(10) for the six datasets, for the LSI, SVM and SVM-LSI methods. Table 5.4 contains the values that were used to plot the charts.

For all datasets, the worst performing method is LSI. One should note that, except for the Cade12 dataset, even LSI has high Accuracy values. SVM is the best for the datasets in English, R8, R52 and 20Ng. SVM-LSI is the best for

		Accuracy		
Dataset	Fold	k-NN	LSI	k-NN-LSI
Bank37	1	0.8741	0.8597	0.8633
	2	0.8561	0.8381	0.8669
	3	0.8309	0.8273	0.8237
	4	0.8669	0.8741	0.8813
	5	0.7957	0.8423	0.8100
	mean	0.8448	0.8483	0.8491
20Ng	1	0.8818	0.8701	0.8544
	2	0.8677	0.8624	0.8409
	3	0.8714	0.8696	0.8443
	4	0.8642	0.8608	0.8411
	5	0.8707	0.8704	0.8393
	mean	0.8712	0.8666	0.8440
R8	1	0.8957	0.9557	0.9635
	2	0.8847	0.9550	0.9524
	3	0.9134	0.9603	0.9616
	4	0.8853	0.9505	0.9537
	5	0.8899	0.9492	0.9531
	mean	0.8938	0.9541	0.9569
R52	1	0.8709	0.9115	0.9203
	2	0.8681	0.9159	0.9209
	3	0.8797	0.9275	0.9231
	4	0.8874	0.9286	0.9291
	5	0.8610	0.9143	0.9154
	mean	0.8734	0.9196	0.9218
Web4	1	0.7294	0.7723	0.8200
	2	0.7321	0.7702	0.8071
	3	0.7595	0.7726	0.7929
	4	0.7321	0.7690	0.7952
	5	0.7476	0.7655	0.7952
	mean	0.7402	0.7699	0.8021
Average	mean	0.8447	0.8717	0.8748

Table 5.3: Values of Accuracy for each of the six datasets, for k-NN, LSI and k-NN-LSI, for each of the five folds, and average Accuracy over all the datasets for each method.

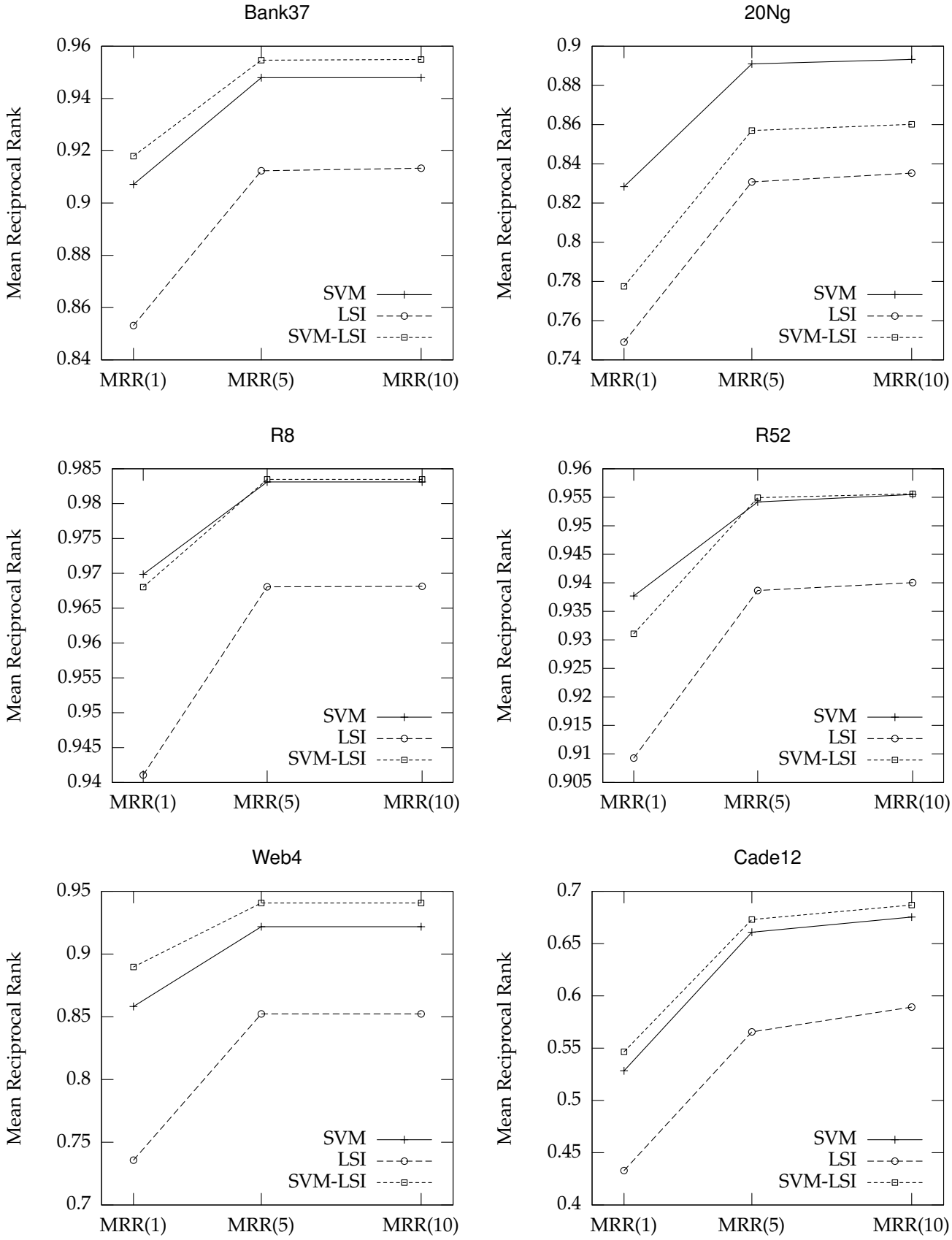

Figure 5.4: MRR(1), MRR(5), and MRR(10) for the six datasets, for SVM, LSI and SVM-LSI.

Dataset	MRR	SVM	LSI	SVM-LSI
Bank37	1	0.9071	0.8531	0.9222
	5	0.9479	0.9123	0.9568
	10	0.9479	0.9133	0.9571
20Ng	1	0.8284	0.7491	0.7775
	5	0.8910	0.8308	0.8570
	10	0.8933	0.8353	0.8602
R8	1	0.9698	0.9411	0.9680
	5	0.9831	0.9681	0.9835
	10	0.9831	0.9681	0.9835
R52	1	0.9377	0.9093	0.9311
	5	0.9542	0.9387	0.9549
	10	0.9555	0.9400	0.9556
Web4	1	0.8582	0.7357	0.8897
	5	0.9218	0.8523	0.9407
	10	0.9218	0.8523	0.9407
Cade12	1	0.5284	0.4328	0.5459
	5	0.6609	0.5656	0.6727
	10	0.6754	0.5892	0.6866

Table 5.4: Values of MRR(1), MRR(5), and MRR(10) for the six datasets, for SVM, LSI and SVM-LSI.

the datasets in Portuguese, Bank37 and Cade12, and also for the English web-pages of computer science departments, Web4 . Considering that SVM was the best performing method in my previous comparison of classification methods, it is particularly interesting that its combination with LSI performs even better in some situations.

As can be seen in Table 5.5, considering average Accuracy over the six datasets, SVM-LSI is slightly better than SVM.

In order to verify if the results obtained previously are statistically significant, I performed 5-fold cross-validation tests using all the datasets and compared the results obtained by each method using paired t-tests. Table 5.6 contains the values of Accuracy for each of the classification methods SVM, LSI, and SVM-LSI, for each of the 5 folds, and average Accuracy over the five folds and five datasets for each method. It was not possible to use 5-fold cross-validation for LSI with the Cade12 dataset because FAQO, the tool that was used to implement LSI, did not

	Accuracy		
Dataset	SVM	LSI	SVM-LSI
Bank37	0.9071	0.8531	0.9222
20Ng	0.8284	0.7491	0.7775
R8	0.9698	0.9411	0.9680
R52	0.9377	0.9093	0.9311
Web4	0.8582	0.7357	0.8897
Cade12	0.5284	0.4328	0.5459
Average	0.8383	0.7702	0.8385

Table 5.5: Values of Accuracy for each of the six datasets, for SVM, LSI, and SVM-LSI, and average Accuracy over all the datasets for each method.

support the required amount of training documents.

Not all results were statistically significant:

SVM-LSI $\neq$ SVM ($p = 0.1867986$)

SVM-LSI $\neq$ LSI ($p = 0.0001911$)

Using the inequalities above, the average across the datasets in Table 5.6, and the fact that only some of the inequalities are statistically significant, it is possible to find a partial ordering for the quality of the classification methods:

SVM-LSI $\approx$ SVM and

SVM-LSI $>$ LSI.

However, once more for 20Ng and Web4, SVM-LSI $>$ SVM, $p = 0.0000032$ and $p = 0.0.0007549$, respectively.

5.5 Summary and Conclusions

The results of the experiments with the combinations of methods have shown that:

- k-NN-LSI, the combination of k-NN with LSI, usually shows a performance

		Accuracy		
Dataset	Fold	SVM	LSI	SVM-LSI
Bank37	1	0.9245	0.8597	0.9353
	2	0.8921	0.8381	0.9101
	3	0.8669	0.8273	0.9029
	4	0.9137	0.8741	0.9281
	5	0.9104	0.8423	0.8996
	mean	0.9015	0.8483	0.9152
20Ng	1	0.9139	0.8701	0.8544
	2	0.9187	0.8624	0.8467
	3	0.9152	0.8696	0.8526
	4	0.9083	0.8608	0.8451
	5	0.9201	0.8704	0.8547
	mean	0.9153	0.8666	0.8507
R8	1	0.9681	0.9557	0.9654
	2	0.9700	0.9550	0.9603
	3	0.9759	0.9603	0.9668
	4	0.9661	0.9505	0.9655
	5	0.9739	0.9492	0.9713
	mean	0.9708	0.9541	0.9659
R52	1	0.9456	0.9115	0.9297
	2	0.9505	0.9159	0.9484
	3	0.9544	0.9275	0.9484
	4	0.9593	0.9286	0.9577
	5	0.9440	0.9143	0.9374
	mean	0.9508	0.9196	0.9443
Web4	1	0.8772	0.7723	0.9142
	2	0.8619	0.7702	0.9000
	3	0.8619	0.7726	0.8798
	4	0.8750	0.7690	0.9012
	5	0.8726	0.7655	0.9107
	mean	0.8697	0.7699	0.9012
Average	mean	0.9216	0.8717	0.9154

Table 5.6: Values of Accuracy for each of the six datasets, for SVM, LSI, and SVM-LSI, for each of the five folds, and average Accuracy over all the datasets for each method.

that is intermediate between the performances of the two original methods. Overall, k-NN-LSI presents an average Accuracy over the six datasets that is higher than the average Accuracy of each original method.

- SVM-LSI, the combination of SVM with LSI, outperforms both original methods SVM and LSI in some datasets. Since SVM is usually the best performing method, it is particularly interesting that its combination with LSI performs even better in some situations.

Considering the present results obtained by the combinations of methods, it is worth to continue exploring this line of research, namely regarding the combinations between the number of dimensions that is considered in the LSI method and the kernel function that is used by the SVM method.

Chapter 6

Incorporation of Unlabeled Data using Centroid-based Methods

This chapter describes how a centroid-based method can be used to combine large volumes of unlabeled data with small volumes of labeled data, in order to improve Accuracy, while at the same time needing less labeled documents for the training phase [Cardoso-Cachopo and Oliveira, 2007b].

6.1 Reasons to Choose a Centroid-based Method

The choice to use a centroid-based method to combine large volumes of unlabeled data with small volumes of labeled data was done for the following reasons:

- Centroid-based methods provide a very good Accuracy, even when compared to the state-of-the-art method in text classification, SVM.
- Centroid-based methods are very fast, both for training and for testing.
- Centroid-based methods require very small amounts of memory to build

the model of the data.

- Centroid-based methods allow for the comparison of batch and incremental updates of the model of the data.

- Centroid-based methods are easy to implement and to modify in order to incorporate information about the unlabeled data.

6.2 Incorporating Unlabeled Data with EM

It has been shown that the use of large volumes of unlabeled data in conjunction with small volumes of labeled data can greatly improve the performance of some TC methods [Nigam et al., 2000]. The combination of the information contained in the labeled and unlabeled data is done using Expectation-Maximization (EM).

EM is a class of iterative algorithms for maximum likelihood estimation of hidden parameters in problems with incomplete data. In this case, I consider that the labels of the unlabeled documents are unknown and use EM to estimate these (unknown) labels.

I propose the combination of EM with a centroid-based method for TC, which works according to the algorithm detailed in Algorithm 6.1, after choosing one of the Equations (2.15) to (2.18) to calculate each class's centroid. This classification method will be referred to as C-EM.

6.3 Incrementally Incorporating Unlabeled Data

The incremental approach is well suited for tasks that require continuous learning, because the available data changes over time. Centroid-based methods are

Inputs: A set of labeled document vectors, L, and a set of unlabeled document vectors U.

Initialization step:

- For each class c_j appearing in L, set D_{c_j} to the set of documents in L belonging to class c_j.
- For each class c_j, calculate the class's centroid $\overrightarrow{c_j}$, using one of the Equations (2.15) to (2.18).

Estimation step:

- For each class c_j appearing in L, set U_{c_j} to the empty set.
- For each document vector $\overrightarrow{d_i} \in U$:
  - Let c_k be the class to whose centroid $\overrightarrow{d_i}$ has the greatest cosine similarity, calculated using Equation (2.7).
  - Add $\overrightarrow{d_i}$ to the set of document vectors labeled as c_k, i.e., set U_{c_k} to $U_{c_k} \cup \{\overrightarrow{d_i}\}$.

Maximization step:

- For each class c_j, calculate $\overrightarrow{c_{j_{new}}}$, using $D_{c_k} \cup U_{c_k}$ as the set of documents labeled as c_k, for each class c_k.

Iterate:

- If, for some j, $\overrightarrow{c_j} \neq \overrightarrow{c_{j_{new}}}$, then set $\overrightarrow{c_j}$ to $\overrightarrow{c_{j_{new}}}$ and repeat from the "Estimation step" forward.

Outputs: For each class c_j, the centroid $\overrightarrow{c_j}$.

Algorithm 6.1: Algorithm for the incorporation of unlabeled data combining EM with a centroid-based method, C-EM.

particularly suitable for this kind of approach because they are very fast, both for training and for testing, and can be applied to very large domains like the web.

Unlike the traditionally used perceptron-based method [Schütze et al., 1995; Wiener et al., 1995], centroid-based methods trivially generalize to multiple classes. In the case of single-label TC, there is no need to fine tune a threshold for deciding when a document belongs to a class, because it will belong to the class represented by the most similar centroid.

In terms of computational efficiency, centroid-based methods are very fast, because updating a centroid-based method can be easily achieved, provided that some additional information is kept in the model for each centroid.

Algorithm 6.2 details the algorithm for the incremental update of a general centroid-based method. This classification method will be referred to as C-Inc.

The next paragraphs describe how the model is updated for each of the centroid-based methods presented in Section 2.3.4. In each case, the goal is to update the model with a new document, $\overrightarrow{d_{new}}$, classified as belonging to class c_j.

The simplest case is for the *sum* method (Equation (2.17)), where the method is updated by calculating a new value for centroid $\overrightarrow{c_j}$ using the following attribution:

$$\overrightarrow{c_j} \leftarrow \overrightarrow{c_j} + \overrightarrow{d_{new}} \tag{6.1}$$

To simplify the incremental update of the *average* method (Equation (2.16)), I maintain in the model also the number of documents, n_j, which were used to

Inputs: A set of labeled document vectors, L, and a set of unlabeled document vectors U.

Initialization step:

- For each class c_k appearing in L, determine the class´s centroid $\overrightarrow{c_k}$, using one of the Equations for the centroids and considering only the labeled documents.

Iterate: For each unlabeled document $d_j \in U$:

- Classify d_j according to its similarity to each of the centroids.
- Update the centroids with the new document d_j classified in the previous step.

Outputs: For each class c_k, the centroid $\overrightarrow{c_k}$.

Algorithm 6.2: Algorithm for the incremental incorporation of unlabeled data using a centroid-based method, C-Inc.

calculate each centroid $\overrightarrow{c_j}$. With this information, the model is updated according to the following attributions:

$$\overrightarrow{c_j} \leftarrow\!\!\rightarrow \frac{(\overrightarrow{c_j}.n_j) + \overrightarrow{d_{new}}}{n_j + 1} \qquad \text{and then} \qquad n_j \leftarrow\!\!\rightarrow n_j + 1 \tag{6.2}$$

For the *normalized sum* method (Equation (2.18)), I maintain, with each normalized centroid $\overrightarrow{c_j}$, also the non-normalized centroid, $\overrightarrow{nnc_j}$, so that updating the model can be performed by the following attributions:

$$\overrightarrow{nnc_j} \leftarrow\!\!\rightarrow \overrightarrow{nnc_j} + \overrightarrow{d_{new}} \qquad \text{and then} \qquad \overrightarrow{c_j} \leftarrow\!\!\rightarrow \frac{\overrightarrow{nnc_j}}{\| \overrightarrow{nnc_j} \|} \tag{6.3}$$

Finally, the most complex case is for the *Rocchio* method (Equation (2.15)), because each new document forces the update of every centroid in the model. In this case, I maintain, for each centroid, $\overrightarrow{c_j}$, two vectors: the sum of the positive examples, $\overrightarrow{pos_j}$, and the sum of the negative examples, $\overrightarrow{neg_j}$. Using these two vectors, Equation (2.15) can be rewritten as

$$\overrightarrow{c_j} = \beta \cdot \overrightarrow{pos_j} - \gamma \cdot \overrightarrow{neg_j} \tag{6.4}$$

Updating the model can be achieved by first updating the appropriate vectors:

$$\overrightarrow{pos_j} \leftarrow\!\!\rightarrow \overrightarrow{pos_j} + \overrightarrow{d_{new}} \tag{6.5}$$

$$\text{for each } i \neq j, \qquad \overrightarrow{neg_i} \leftarrow\!\!\rightarrow \overrightarrow{neg_i} + \overrightarrow{d_{new}} \tag{6.6}$$

and then, calculating all the new centroids according to Equation (6.4).

In this work I experimentally show how incrementally updating the model (that is, the centroids) with the unlabeled documents during the training phase

influences the Accuracy of a centroid-based method.

6.4 Comparing Semi-Supervised and Incremental TC

This section provides empirical evidence that, when using centroid-based methods to perform single-label TC, incorporating information about unlabeled data improves results if the initial model of the data is good enough, while it may actually worsen results if the initial model was not good enough.

This is done, first using one synthetic dataset, and then using four real-world datasets to confirm the results obtained previously.

This chapter reports results using Accuracy and not several values for MRR because it is necessary to vary the number of labeled documents per class, and using only Accuracy the charts are easier to understand.

6.4.1 Synthetic Dataset

The synthetic dataset was created with several goals in mind. First, it had to be a dataset that was simple, and whose properties were well known. It was also supposed to allow the generation of as many "documents" as were needed for the experiments. Ultimately, it was necessary to prove that the effect of using unlabeled data depends not only on the classification method that is used, but also on the quality of the dataset.

What will be called the synthetic dataset corresponds to four different mixtures of Gaussians, in one dimension, thus representing, in fact, four different datasets.[1] The data points belonging to each Gaussian distribution are randomly

[1]Similar experiments were performed with Gaussians of up to 5000 dimensions. Because all the Gaussians corresponded to high-dimensional spheres in the n-dimensional space, the results

generated according to the well known Gaussian probability distribution function:

$$g(x) = \frac{e^{-(x-\mu)^2/2\sigma^2}}{\sigma\sqrt{2\pi}} \tag{6.7}$$

In each mixture of Gaussians, each Gaussian distribution corresponds to a different class. I used different ratios between parameters μ (mean) and σ (standard deviation or width of the Gaussian) to simulate problems with varying difficulties. Figure 6.1 depicts the four different mixtures of Gaussians, that is, the four datasets with varying difficulties, that were used.

Is is easy to see that, as the ratio $\frac{\mu}{\sigma}$ decreases, the problem is more difficult, because the overlap between the Gaussian distributions increases. This means that it is more difficult to decide which of the two distributions originated a randomly generated point. In particular, the limit for the Accuracy of the optimal classifier can be obtained as the value of the cumulative distribution function of the Gaussian at the point of intersection.

6.4.2 Using Unlabeled Data with the Synthetic Dataset

To prove that the effect of using unlabeled data depends not only on the classification method that is used, but also on the quality of the dataset, I randomly generated four different two-class datasets, each according to two different one-dimensional Gaussian distributions, so that each dataset posed a different difficulty level, known in advance. With each dataset, I used the same classification methods, and in the end I compared the results.

So that the experiments would not depend on one particular ordering of the

that were obtained were similar to the ones reported here.

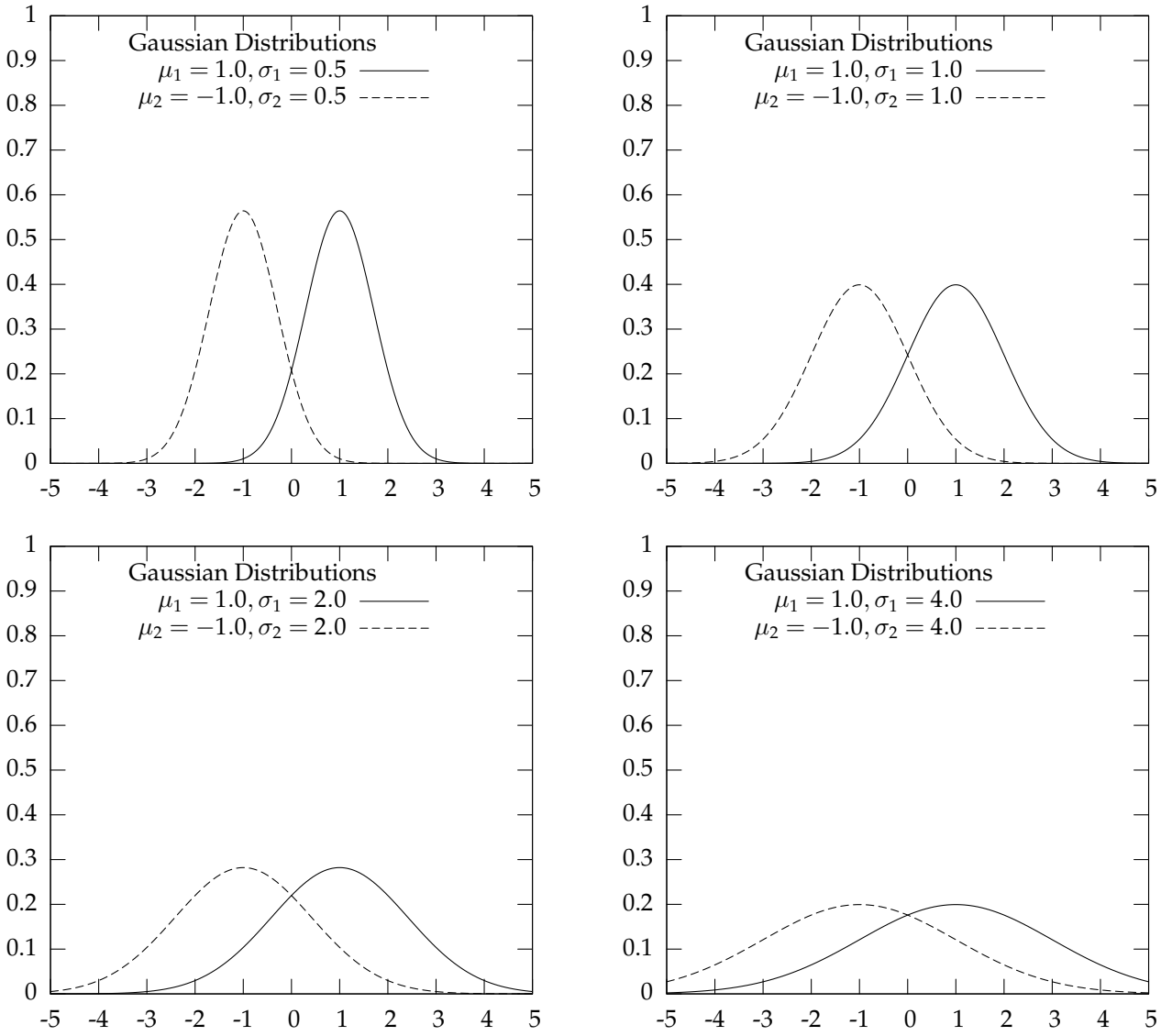

Figure 6.1: Combinations of Gaussians used as the synthetic dataset.

dataset, I repeated the following steps 500 times for each dataset:

1. Randomly generate from 1 to 20 labeled training documents per class (in this case, per Gaussian Distribution).

2. Based on the training documents alone, calculate each class's centroid, that is, the average of the numbers corresponding to the training documents.

3. Randomly generate 5000 test documents. These should be approximately half from each class.

4. Determine Accuracy for the centroid-based method.

5. Randomly generate 5000 unlabeled documents.

6. Using C-EM / C-Inc, update each class's centroid.

7. Determine Accuracy for C-EM / C-Inc, using the same test documents as for the centroid-based method alone.

This can be summed-up in Algorithm 6.3.

```
For each dataset, repeat 500 times
 For i = 1 to 20
  Randomly generate i training docs per class
  Calculate each class's centroid
  Randomly generate 5000 test docs
  Determine Accuracy for the centroid-based method
  Randomly generate 5000 unlabeled docs
  Using EM / Inc, update the centroids
  Determine Accuracy for EM / Inc
```

Algorithm 6.3: Algorithm for using unlabeled data with the synthetic dataset.

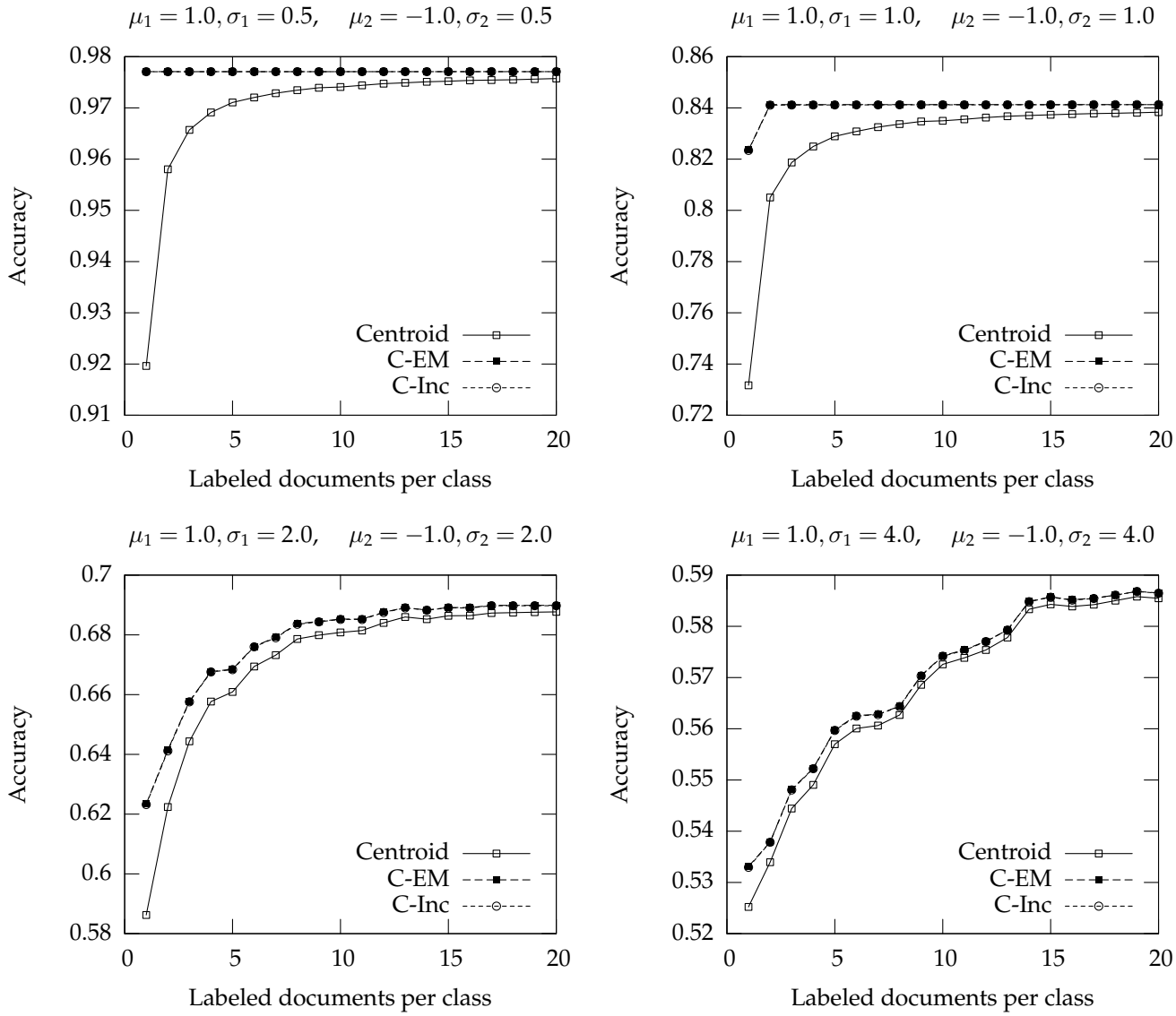

Figure 6.2: Accuracy for the Gaussians dataset, as a function of the number of labeled documents per class that were used, for Centroid, C-EM and C-Inc.

Figure 6.2 presents the mean Accuracy values for the 500 tests for each dataset, as a function of the number of labeled documents per class that were used.

Observe that there are some features that are common, independently of the difficulty level of the dataset:

- As expected, more labeled training documents per class improve results. This observed improvement, however, decreases as the number of labeled documents increases.

- As the number of labeled training documents per class increases, the effect of using unlabeled documents decreases. This means that, having unlabeled data is always good, and it is better when there is less labeled data.
- Both methods (C-EM and C-Inc) of updating the centroids of the classes give the same results, because both lines are the same. This happens in this case because there are many unlabeled documents and their structure is very simple (they are real numbers), so the order by which they are incorporated in the model does not make a difference in the results obtained. In this setting, it is better to incrementally update the centroids, because the results are the same and this method is computationally faster.

As for the features that depend on the difficulty level of the dataset:

- As the difficulty level of the dataset increases, the Accuracy that can be achieved decreases (note the different ranges in the Y axis).
- As the difficulty level of the dataset increases, the difference between the lower and higher Accuracy level that can be achieved (the lower and higher values in the Y axis) decreases. This means that the advantage of having unlabeled data decreases.
- When only one labeled document per class is available, 5000 unlabeled documents led to an improvement in Accuracy from 0.9196 to 0.9771, for the easier dataset, while for the most difficult dataset improvement is much smaller, only from 0.5252 to 0.5331.
- As a general rule, the effect of using 5000 unlabeled documents to update the model of the data decreases as the difficulty level of the dataset increases.

All these observations lead to the conclusion that the effect of using unlabeled

data depends not only on the classification method that is used, but also on the quality of the dataset that is considered. If the initial model of the labeled data was already able to achieve a high Accuracy, using unlabeled data will help improve the results, and it will help more if the initial model was better.

6.4.3 Using Unlabeled Data with the Real World Datasets

To analyze the effect of using information from unlabeled data with the real-world datasets, I used the R8, 20Ng, Web4 and Cade12 datasets. As already pointed out, it was possible to find good classifiers for the first three datasets, but not for the fourth. The Bank37 and the R52 datasets were not used in these experiments because the number of labeled documents for some of the classes was too low.

The steps followed for testing these datasets were similar to the ones followed in the previous section, and can be summed-up in Algorithm 6.4. Due to their reduced size, for the R8 and Web4 datasets I could only use 1000 unlabeled documents. For the R8 dataset, it was possible to select a maximum of 40 labeled documents per class.

```
For each dataset, consider its train/test split
For each dataset, repeat 5 times
 For i in {1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 15, 20, 30, 40, 50, 60, 70}
  Randomly select i labeled training docs per class
  Calculate each class's centroid using Equation (2.18)
  Determine Accuracy for the centroid-based method
  Randomly select 5000 (or 1000) unlabeled docs
           from the remaining training docs
  Using EM / Inc, update the centroids
  Determine Accuracy for EM / Inc
```

Algorithm 6.4: Algorithm for using unlabeled data with the real-world datasets.

Figure 6.3 presents the charts for the mean Accuracy values for the 5 runs for each dataset. Tables 6.1, 6.2, 6.3, and 6.4 contain the values that were used to plot the charts. These tables also contain values for other methods using small numbers of labeled documents per class, which will be discussed afterwards.

Observe that, independently of the dataset that is used, the lines are steeper on the left side (where there are less labeled documents per class) than they are in the right side. This means that more labeled training documents per class improve results, and that this improvement decreases as the number of labeled documents per class increases.

The rest of the observations depend on the dataset that is used:

- For R8, which allowed the achievement of a high Accuracy using all 5485 training documents with Centroid and SVM, Accuracy varies from 0.5530 with one labeled document per class to 0.9375 with 40 labeled documents per class for the Centroid method. It is worth noting that Accuracy starts over 0.67 with a single labeled document per class for both C-EM and C-Inc,

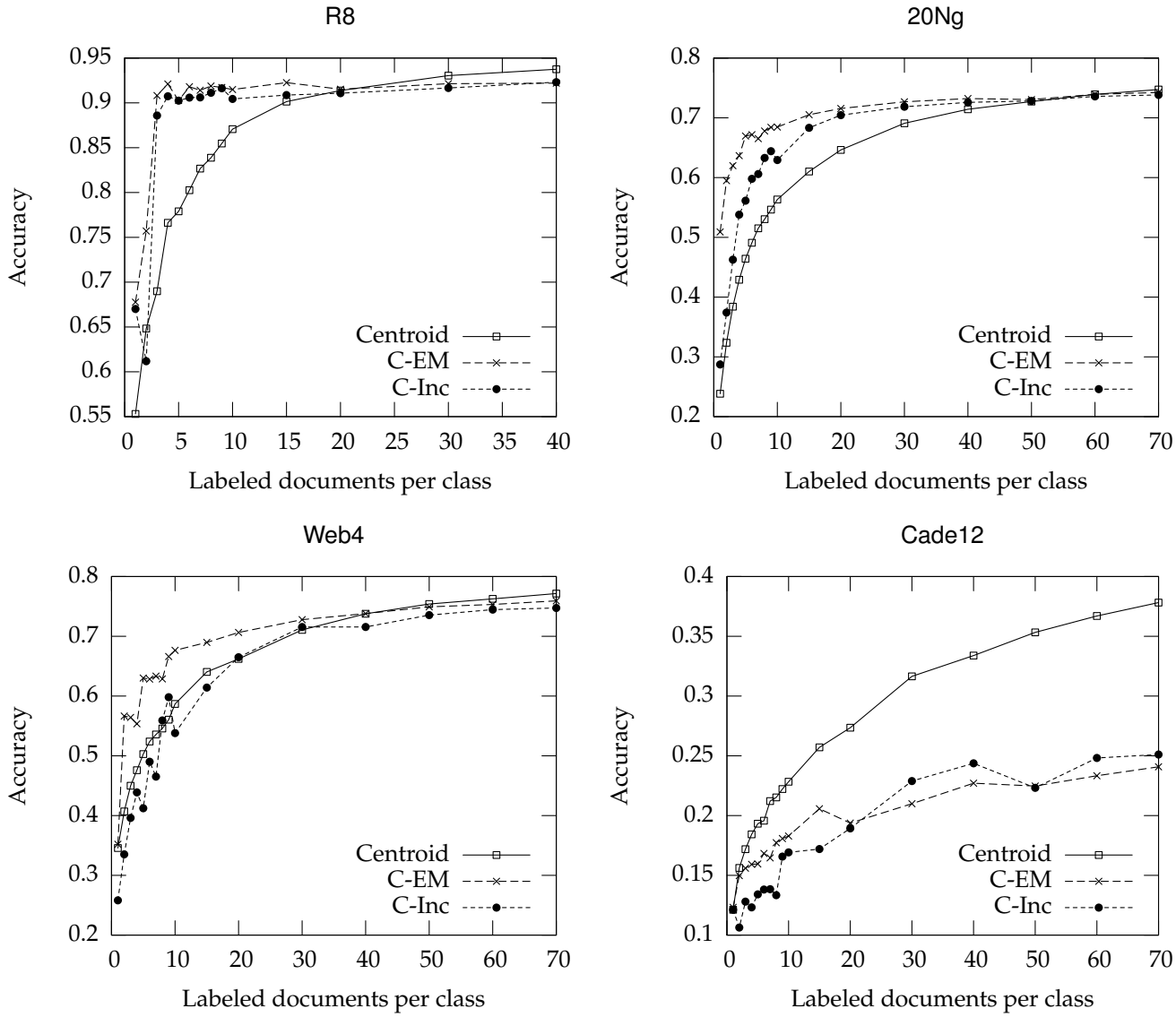

Figure 6.3: Accuracy for the four real world datasets, as a function of the number of labeled documents per class that were used, for Centroid, C-EM and C-Inc.

				R8				
Lab/class	Centroid	C-EM	C-Inc	SVM	N-Bayes	k-NN	LSI	Vector
1	0.5530	0.6777	0.6701	0.5536	0.5033	0.5533	0.5433	0.5530
2	0.6483	0.7571	0.6118	0.6796	0.5866	0.6437	0.6597	0.6222
3	0.6899	0.9082	0.8859	0.7556	0.6386	0.6775	0.6980	0.6368
4	0.7662	0.9211	0.9074	0.8312	0.7103	0.7422	0.7596	0.6949
5	0.7790	0.9027	0.9025	0.8439	0.7314	0.7543	0.7741	0.7060
6	0.8026	0.9180	0.9059	0.8500	0.7648	0.7718	0.7854	0.7087
7	0.8267	0.9141	0.9061	0.8763	0.7924	0.7944	0.8018	0.7386
8	0.8388	0.9188	0.9113	0.8930	0.8099	0.8027	0.8153	0.7381
9	0.8547	0.9172	0.9163	0.9047	0.8270	0.8135	0.8234	0.7434
10	0.8707	0.9149	0.9044	0.9098	0.8531	0.8234	0.8261	0.7432
15	0.9014	0.9226	0.9087	0.9235	0.8949	0.8344	0.8364	0.7604
20	0.9138	0.9152	0.9108	0.9320	0.9126	0.8438	0.8405	0.7612
30	0.9304	0.9214	0.9166	0.9391	0.9261	0.8410	0.8514	0.7641
40	0.9375	0.9220	0.9231	0.9461	0.9365	0.8463	0.8645	0.7646
All	0.9356	—	—	0.9698	0.9607	0.8524	0.9411	0.7889

Table 6.1: Accuracy values for the R8 dataset, as a function of the number of labeled documents per class that were used, and using all the training documents, for Centroid, C-EM, C-Inc, SVM, Naive Bayes, k-NN, LSI and Vector.

				20Ng				
Lab/class	Centroid	C-EM	C-Inc	SVM	N-Bayes	k-NN	LSI	Vector
1	0.2383	0.5090	0.2873	0.2383	0.1139	0.2382	0.2448	0.2383
2	0.3236	0.5949	0.3742	0.3231	0.1413	0.3222	0.3083	0.3037
3	0.3840	0.6199	0.4626	0.3775	0.1882	0.3765	0.3553	0.3387
4	0.4290	0.6367	0.5379	0.4186	0.2135	0.4150	0.3852	0.3681
5	0.4642	0.6695	0.5614	0.4535	0.2368	0.4481	0.4080	0.3855
6	0.4910	0.6718	0.5979	0.4820	0.2709	0.4709	0.4303	0.4042
7	0.5151	0.6652	0.6059	0.5040	0.3031	0.4944	0.4500	0.4228
8	0.5304	0.6779	0.6329	0.5158	0.3104	0.5053	0.4585	0.4283
9	0.5464	0.6841	0.6442	0.5298	0.3261	0.5175	0.4713	0.4386
10	0.5634	0.6846	0.6294	0.5436	0.3547	0.5291	0.4812	0.4487
15	0.6102	0.7052	0.6833	0.5924	0.4254	0.5621	0.5168	0.4715
20	0.6463	0.7156	0.7045	0.6302	0.4656	0.5933	0.5507	0.4994
30	0.6909	0.7268	0.7186	0.6772	0.5423	0.6277	0.5917	0.5346
40	0.7143	0.7322	0.7258	0.7036	0.5766	0.6510	0.6162	0.5557
50	0.7272	0.7306	0.7282	0.7149	0.6150	0.6633	0.6380	0.5704
60	0.7393	0.7393	0.7358	0.7290	0.6410	0.6721	0.6443	0.5823
70	0.7474	0.7424	0.7383	0.7392	0.6546	0.6779	0.6506	0.5928
All	0.7885	—	—	0.8284	0.8103	0.7593	0.7491	0.7240

Table 6.2: Accuracy values for the 20Ng dataset, as a function of the number of labeled documents per class that were used, and using all the training documents, for Centroid, C-EM, C-Inc, SVM, Naive Bayes, k-NN, LSI and Vector.

				Web4				
Lab/class	Centroid	C-EM	C-Inc	SVM	N-Bayes	k-NN	LSI	Vector
1	0.3457	0.3520	0.2582	0.3487	0.3649	0.3437	0.3517	0.3457
2	0.4070	0.5666	0.3351	0.4284	0.4665	0.4007	0.4322	0.4039
3	0.4499	0.5642	0.3961	0.4756	0.4582	0.4437	0.4466	0.4248
4	0.4759	0.5536	0.4388	0.4681	0.5123	0.4719	0.4769	0.4486
5	0.5029	0.6302	0.4122	0.4954	0.5540	0.4920	0.4903	0.4617
6	0.5239	0.6284	0.4901	0.5285	0.5633	0.5030	0.4928	0.4643
7	0.5357	0.6334	0.4652	0.5298	0.5685	0.5201	0.4964	0.4655
8	0.5457	0.6287	0.5592	0.5421	0.5834	0.5281	0.5074	0.4712
9	0.5605	0.6662	0.5981	0.5537	0.5977	0.5334	0.5117	0.4732
10	0.5868	0.6765	0.5380	0.5795	0.6183	0.5586	0.5215	0.4877
15	0.6404	0.6898	0.6142	0.6221	0.6632	0.5971	0.5398	0.5030
20	0.6622	0.7064	0.6650	0.6493	0.6923	0.6076	0.5554	0.5085
30	0.7107	0.7278	0.7155	0.7016	0.7231	0.6284	0.5679	0.5212
40	0.7377	0.7380	0.7156	0.7330	0.7477	0.6513	0.5840	0.5474
50	0.7540	0.7493	0.7354	0.7520	0.7580	0.6619	0.5970	0.5514
60	0.7626	0.7533	0.7446	0.7229	0.7688	0.6676	0.6086	0.5683
70	0.7715	0.7593	0.7474	0.7398	0.7784	0.6754	0.6285	0.5781
All	0.8266	—	—	0.8582	0.8352	0.7256	0.7357	0.6447

Table 6.3: Accuracy values for the Web4 dataset, as a function of the number of labeled documents per class that were used, and using all the training documents, for Centroid, C-EM, C-Inc, SVM, Naive Bayes, k-NN, LSI and Vector.

				Cade12				
Lab/class	Centroid	C-EM	C-Inc	SVM	N-Bayes	k-NN	LSI	Vector
1	0.1212	0.1232	0.1212	0.1233	0.1372	0.1327	0.1325	0.1212
2	0.1562	0.1498	0.1063	0.1588	0.1501	0.1607	0.1561	0.1534
3	0.1719	0.1560	0.1280	0.1752	0.1559	0.1707	0.1655	0.1722
4	0.1843	0.1592	0.1232	0.1807	0.1575	0.1840	0.1665	0.1775
5	0.1932	0.1595	0.1341	0.1869	0.1633	0.1934	0.1745	0.1809
6	0.1957	0.1683	0.1381	0.1945	0.1683	0.1931	0.1755	0.1806
7	0.2121	0.1646	0.1384	0.2014	0.1738	0.2044	0.1846	0.1853
8	0.2152	0.1773	0.1334	0.2061	0.1728	0.2074	0.1873	0.1855
9	0.2222	0.1808	0.1657	0.2127	0.1831	0.2155	0.1925	0.1904
10	0.2283	0.1829	0.1692	0.2198	0.1930	0.2194	0.1951	0.1955
15	0.2571	0.2057	0.1720	0.2455	0.2084	0.2395	0.2062	0.2055
20	0.2735	0.1937	0.1893	0.2684	0.2220	0.2511	0.2174	0.2136
30	0.3165	0.2099	0.2289	0.3101	0.2504	0.2734	0.2332	0.2278
40	0.3339	0.2271	0.2437	0.3310	0.2589	0.2926	0.2402	0.2387
50	0.3533	0.2247	0.2232	0.3455	0.2736	0.3086	0.2496	0.2464
60	0.3670	0.2333	0.2482	0.3602	0.2708	0.3156	0.2632	0.2572
70	0.3782	0.2408	0.2511	0.3683	0.2875	0.3264	0.2679	0.2604
All	0.5148	—	—	0.5284	0.5727	0.5120	0.4329	0.4142

Table 6.4: Accuracy values for the Cade12 dataset, as a function of the number of labeled documents per class that were used, and using all the training documents, for Centroid, C-EM, C-Inc, SVM, Naive Bayes, k-NN, LSI and Vector.

and stays over 0.9 with as little as 4 labeled documents per class for both these methods. In fact, for Centroid, Accuracy is higher using 40 labeled documents per class than it was using all training documents as labeled. This is probably because this dataset is very skewed.

- For 20Ng, which also allowed high Accuracy values using all 11293 training documents with Centroid and SVM, Accuracy varies from 0.2383 with one labeled document per class to 0.7474 with 70 labeled documents per class. Using C-EM, Accuracy starts at 0.5 with a single labeled document per class, and 15 labeled documents per class are enough to achieve 0.7 Accuracy. For this dataset, C-EM consistently outperforms C-Inc, and both these methods perform better than Centroid for small numbers of labeled documents per class.

- For Web4, the Accuracy of the Centroid method steadyly improves from 0.35 to 0.77 as the number of labeled documents per class goes from 1 to 70. For this dataset, using C-EM to incorporate the unlabeled documents improves performance for small numbers of labeled documents per class, but using C-Inc has the opposite effect.

- For Cade12, which already had poor Accuracy values using all 27322 training documents with Centroid and SVM, Accuracy varies from 0.1212 with one labeled document per class to 0.3782 with 70 labeled documents per class (note the different ranges in the Y axis). For this dataset, using unlabeled data worsens results, whether C-EM or C-Inc is used.

- For R8, 20Ng, and Web4, using unlabeled data improves results, and the improvement is larger for smaller numbers of labeled documents per class, while for Cade12, using unlabeled data worsens results. This observation allows me to experimentally confirm my initial intuition that it is only worth it to use unlabeled data if the initial model for the labeled data already pro-

vided good results.

- For R8, 20Ng, and Web4, as the number of labeled training documents per class increases, the effect of using unlabeled documents decreases. Like for the synthetic dataset, having unlabeled data is good, and it is better when there is less labeled data.

- For all datasets, the two methods (C-EM and C-Inc) of updating the centroids of the classes using unlabeled documents give different results, because now the order by which the centroids are updated is different. Moreover, the difference in the results decreases as the number of labeled documents increases. This happens because when more labeled documents are available, the initial model is better, and therefore the centroids are less moved by either one of the updating techniques. Generally, using C-EM to update the centroids yields better results than using C-Inc.

All these observations confirm the previous conclusion that the effect of using unlabeled data depends not only on the classification method that is used, but also on the difficulty of the dataset that is considered. For the datasets for which it is possible to achieve a good classification Accuracy (R8, 20Ng, and Web4), using unlabeled data helped improve results, while for the dataset for which the initial results were not so good (Cade12), using unlabeled data actually made the results even worse.

I should note that the results presented here cannot be directly compared to those in [Nigam et al., 2000] because the test settings were very different: (1) the partition that was done to the 20Ng dataset was different, (2) the Reuters dataset here is treated as single-labeled and in [Nigam et al., 2000] was treated as multi-labeled, and (3) I use the standard mod Apté train/test partitions for this dataset and that work does not.

Another aspect that should be considered is how other methods, aside from Centroid, behave with very small numbers of labeled documents per class. In order to allow such a comparison, Figures 6.4 and 6.5 present the charts for the mean Accuracy values for the 5 runs for each dataset and each method. The values used to plot the charts were already presented in Tables 6.1, 6.2, 6.3, and 6.4.

By observing these figures and tables, it is possible to draw some conclusions:

- As expected, for all datasets and all methods, performance increases as the number of labeled documents per class increases.

- For R8, 20Ng, and Web4 the best performing method when there are small numbers of labeled documents available is C-EM, which means that, in these cases, it is worth using unlabeled data. Under the same circumstances, the best performing method for Cade12 is Centroid, largely outperforming Naive Bayes, which is the best performing method when all the training documents are used for this dataset.

- Of the methods that do not make use of the unlabeled documents, there is not one that always outperforms the others: for R8, the best method is SVM; for 20Ng (and Cade12!), the best is Centroid; and for Web4, the best is Naive Bayes.

Nigam [Nigam et al., 2000] already showed that unlabeled documents can be used to improve the performance of a classifier based on the Naive Bayes method when there are small portions of labeled documents available. Here, I showed that the unlabeled documents can also be used to improve a Centroid based classifier, and that the obtained classifier provided the best results for small numbers of labeled documents, on three of the four datasets analyzed.

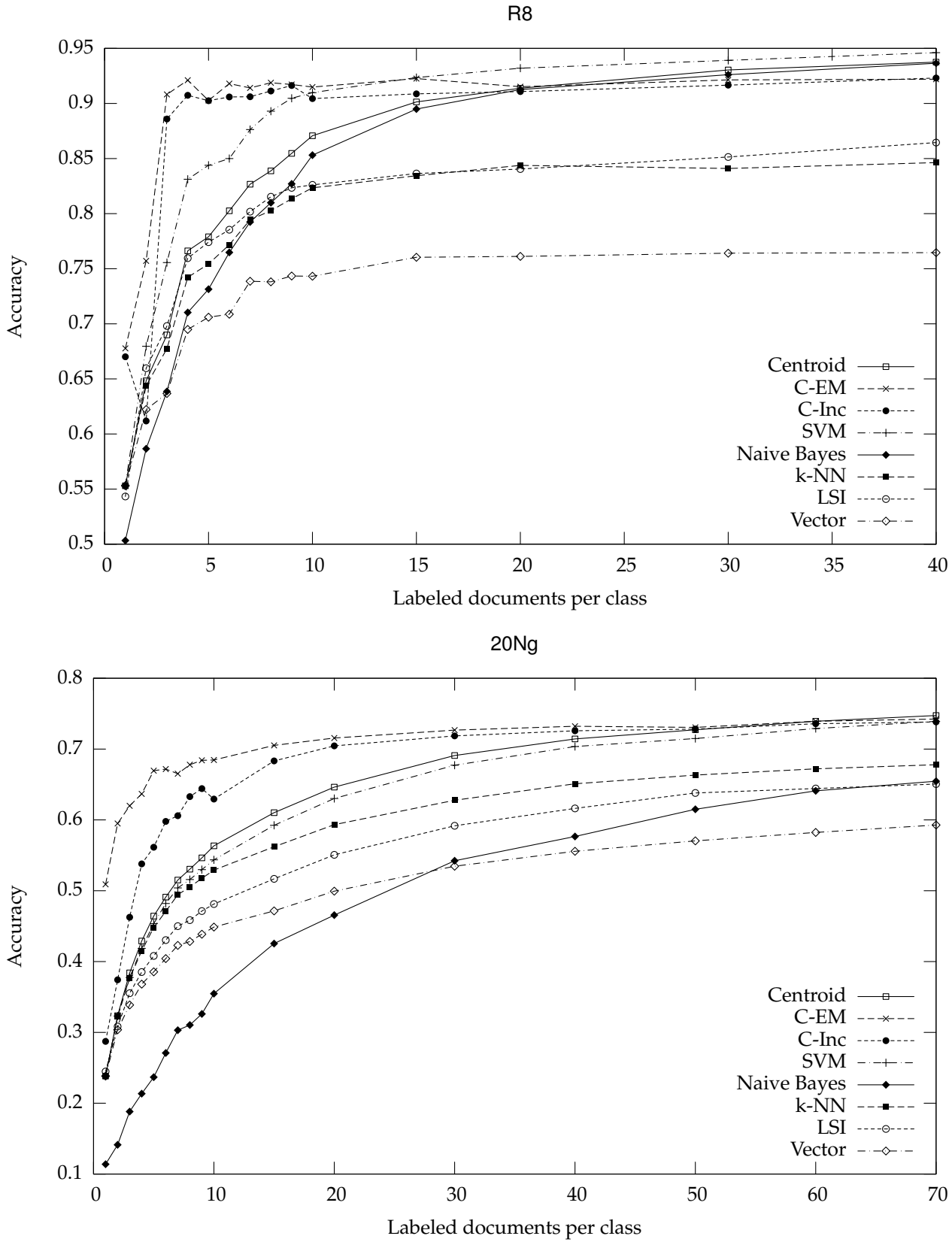

Figure 6.4: Accuracy for two of the real world datasets, R8 and 20Ng, as a function of the number of labeled documents per class that were used, for Centroid, C-EM, C-Inc, SVM, Naive Bayes, k-NN, LSI and Vector.

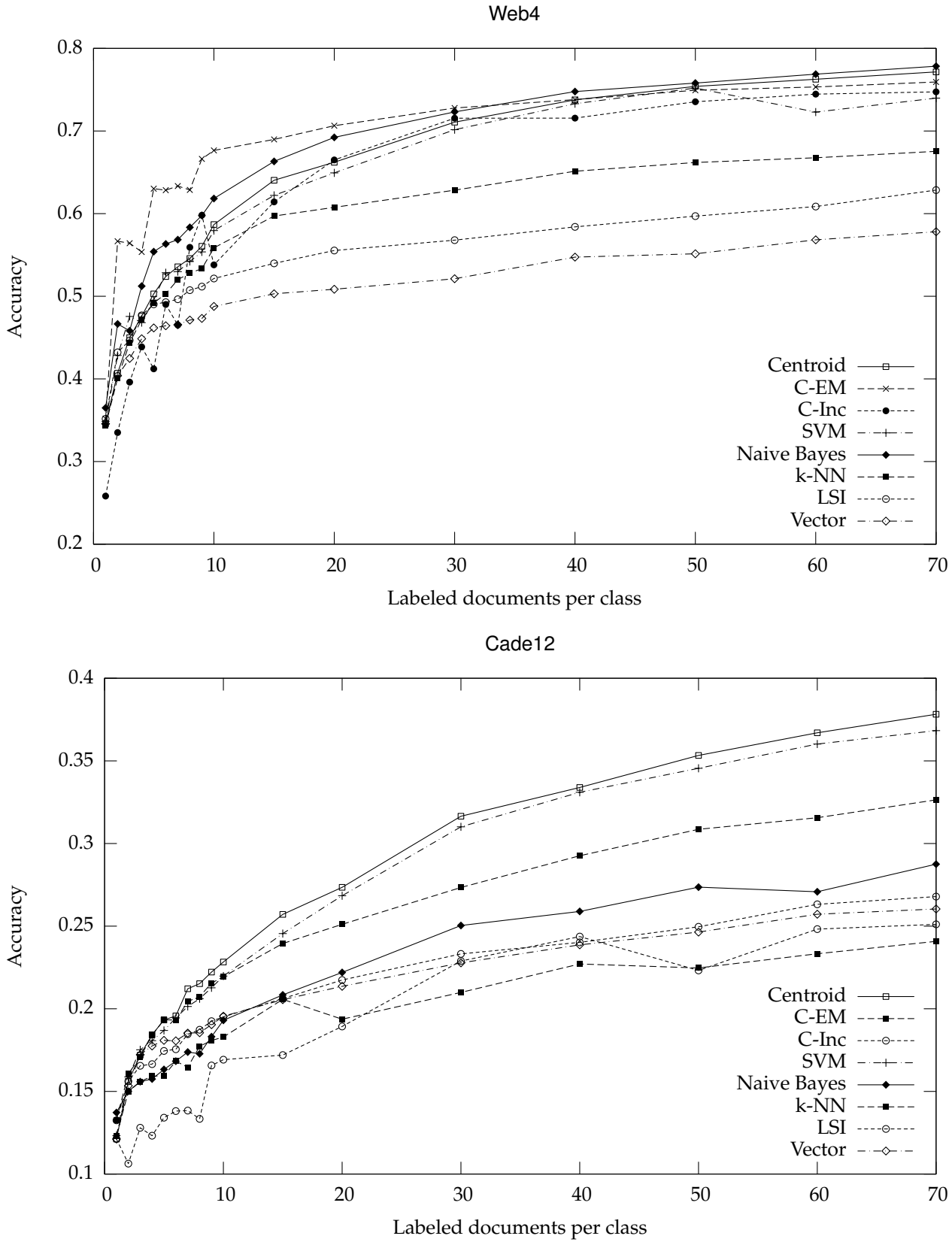

Figure 6.5: Accuracy for two of the real world datasets, Web4 and Cade12, as a function of the number of labeled documents per class that were used, for Centroid, C-EM, C-Inc, SVM, Naive Bayes, k-NN, LSI and Vector.

6.5 Summary and Conclusions

This chapter described how it is possible incorporate information about unlabeled documents during the training phase to complement the information conveyed by labeled documents, and hence improve classification Accuracy. It used a centroid-based method and explained how to incorporate information about the unlabeled documents in batch mode using EM, and incrementally, by updating the centroids every time a new unlabeled document arrives.

Using one synthetic dataset and four real-world datasets, it provided empirical evidence that, if the initial model of the data is reasonable, using unlabeled documents improves performance. On the other hand, using unlabeled data can actually hurt performance if the initial model of the data is not good enough.

The results presented in this Chapter lead to the following conclusions:

- For the three datasets for which it is possible to find good classifiers using all the training documents (R8, 20Ng, and Web4), it is worth to incorporate information about the unlabeled documents when there are only a few labeled documents available. And in this case it is better to incorporate that information using C-EM.
- For the dataset for which it was not possible to find a good classifier (Cade12), it is better to ignore the information about the unlabeled documents, because it will worsen the results. For this dataset, when there are only a few labeled documents available, it is better to use the simple centroid-based method, even when considering all the classification methods compared in this work.

Chapter 7

Conclusions and Future Work

This chapter presents the main contributions of this work, as well as a description of possible developments that can be pursued as future work.

7.1 Main Contributions

The work described in this dissertation explored several state-of-the-art methods for Text Categorization (TC), and how some of them can be combined in order to improve their performance.

To be able to perform a "fair" comparison between the different methods, a computational framework named IREP—*Information Retrieval Experimentation Package*— was developed. IREP allows testing different classification methods while using the same datasets. IREP was built with the goals of being very configurable, of allowing combinations of different classification methods, and of allowing combinations of different term weighting schemes for the several datasets that were used. With IREP, it was possible to perform the tests of the several methods using the same datasets, the same terms and the same machine.

It was also necessary to create a new data collection, and to retrieve several other standard collections, so that the results obtained with this work could be compared to the results obtained by other researchers. The datasets that resulted from pre-processing the freely available collections were made publicly available on a website. Part of the pre-processing that was applied to the original collections was applied to ensure that each document belongs to a single class, that is, that the datasets are single-label.

Because this work was concerned with single-label TC, it was also necessary to choose an evaluation measure more adequate to the single-label setting. The measure that was used was Accuracy, and it was complemented with the Mean Reciprocal Rank.

One of the first uses for the IREP framework was to perform a comprehensive comparison of the several classification methods explored in this work, including the ones that are most frequently used in the TC area. For this comparison were used not only the standard collections that are used in this area, but also two other less known collections in a different language, to confirm the results obtained. This comparison led to several conclusions: of the several centroid-based methods that are used in the literature, C-NormSum is the one that provides the best results; C-NormSum is competitive with other more computational demanding methods, like SVM; the traditional *tfidf* term weighting approach is very effective, even when compared to a more recent approach.

This work presented the combination of different classification methods with LSI, namely k-NN and SVM. Its results suggested that k-NN-LSI, the combination of k-NN with LSI, usually shows a performance that is intermediate between the ones of these two methods. Overall, k-NN-LSI presents an average Accuracy over the five datasets that is higher than the average Accuracy of each original method. It also showed that SVM-LSI, the combination of SVM with LSI, out-

performs both original methods in some datasets. Because the SVM method has been considered the top-performing method in several method comparisons in the literature, the fact that SVM-LSI performs even better in some situations is even more important.

This work also presented the combination of EM with a centroid-based method that uses information from small volumes of labeled documents together with information from larger volumes of unlabeled documents for text categorization, and showed how a centroid-based method can be used to update the model of the data incrementally, based on new evidence from the unlabeled documents. Using one synthetic dataset and three real-world datasets, the results obtained with this work provided empirical evidence that, if the initial model of the data is reasonable, using unlabeled documents improves performance. On the other hand, using unlabeled data can actually hurt performance if the initial model of the data is not good enough.

Finally, this work also provided a comparison between the most used classification methods and the combinations of methods proposed in this work. This comparison led to the conclusion that the combinations of methods can improve the results obtained by the individual methods, even when they initially showed a good performance, and that the improvement also depends on the difficulty level of the dataset that was used.

7.2 Future Work

The work described in this dissertation opens up a series of interesting lines of research for the future. Some of the most interesting possible developments for this work include, but are not limited to, the following:

- Given the present results regarding the combinations of methods, it will be interesting to explore them even further, namely regarding the combinations between the number of dimensions that is considered in the LSI method and the kernel function that is used by the SVM method.

- It will be important to extend the centroid-based approach to multi-label datasets. This will involve finding a threshold from which a document belongs to a class, and extending the semi-supervised approach for incorporating unlabeled data using multi-labeled datasets.

- The knowledge that was acquired in the development of the present IREP framework is currently being applied in the specification and development of IR-BASE [Calado et al., 2007], an object oriented framework that allows the integration of several components, documentation and services, and whose focus is on the rapid development of prototypes of Information Retrieval tools.

- A very active research area concerns Web 2.0, which refers to a second generation of web-based services which, among other things, allows internet users to publish documents that they find interesting along with classification keywords. The goal of these keywords is to facilitate searches on related topics. The classification methods studied in this work may be successfully applied to improve these searches.

Appendix A

Datasets

This appendix contains tables with the number of training documents, number of test documents and total number of documents per class for each dataset, for the train/test split used in my experiments. It also has a section detailing the pre-processing applied to the documents in each collection.

A.1 Bank37 Dataset

Class	Training	Test	Total
bnffaps	51	21	72
bnffcf	33	19	52
bnfftib	15	3	18
bnfftin	14	4	18
bnqucum	21	10	31
bnqumt	22	9	31
bnqupco	31	15	46
bnqupe	37	25	62
bnqutib	14	10	24
bnqutl	11	10	21
boccaob	18	10	28
bocdob	35	21	56
ccarp	21	4	25
ccarpdi	27	7	34
cccan	4	3	7
ccccs	27	9	36
cccp	16	6	22
ccreq	7	6	13
ccv	9	5	14
cdratm	10	5	15
cdsub	10	4	14
cdua	11	4	15
chribn	29	17	46
co100	4	1	5
cocd	7	4	11
coepbv	26	19	45
coj	5	4	9
copee	23	6	29
cos	13	4	17
crhprc	6	7	13
nhi	230	116	346
oac	15	9	24
ocam	12	6	18
ocpbdp	30	16	46
ocpbig	31	11	42
op	37	21	58
ottib	16	12	28
Total	928	463	1391

Table A.1: Training, Test and Total number of documents for each of the 37 classes of the Bank collection, considering my random split — Bank37 dataset.

A.2 Web4 Dataset

Class	Training	Test	Total
project	336	168	504
course	620	310	930
faculty	750	374	1124
student	1097	544	1641
Total	2803	1396	4199

Table A.2: Training, Test and Total number of documents for each of the 4 classes of the Webkb collection, considering my random split — Web4 dataset.

A.3 R8 and R52 Datasets

# Topics	Training	Test	Other	Total
0	1828	280	8103	10211
1	6552	2581	361	9494
2	890	309	135	1334
3	191	64	55	310
4	62	32	10	104
5	39	14	8	61
6	21	6	3	30
7	7	4	0	11
8	4	2	0	6
9	4	2	0	6
10	3	1	0	4
11	0	1	1	2
12	1	1	0	2
13	0	0	0	0
14	0	2	0	2
15	0	0	0	0
16	1	0	0	1

Table A.3: Training, Test, Other and Total number of documents having a certain number of topics for the Reuters-21578 collection, considering the standard Mod Apté split. Here, "# Topics" stands for the number of topics of the documents, "Train" for the number of training documents with a certain number of topics, "Test" for the number of test documents with a certain number of topics, "Other" for the number of documents that are not considered as training nor test with a certain number of topics, and "Total" for the total number of documents with a certain number of topics.

Class	Training	Test	Total
acq	1596	696	2292
crude	253	121	374
earn	2840	1083	3923
grain	41	10	51
interest	190	81	271
money-fx	206	87	293
ship	108	36	144
trade	251	75	326
Total	5485	2189	7674

Table A.4: Training, Test and Total number of documents per class for the 8 most frequent classes of the Reuters-21578 collection, considering documents with a single topic and the standard Mod Apté split — R8 dataset.

Class	Training	Test	Total
acq	1596	696	2292
alum	31	19	50
bop	22	9	31
carcass	6	5	11
cocoa	46	15	61
coffee	90	22	112
copper	31	13	44
cotton	15	9	24
cpi	54	17	71
cpu	3	1	4
crude	253	121	374
dlr	3	3	6
earn	2840	1083	3923
fuel	4	7	11
gas	10	8	18
gnp	58	15	73
gold	70	20	90
grain	41	10	51
heat	6	4	10
housing	15	2	17
income	7	4	11
instal-debt	5	1	6
interest	190	81	271
ipi	33	11	44
iron-steel	26	12	38
jet	2	1	3
jobs	37	12	49
lead	4	4	8
lei	11	3	14
livestock	13	5	18
lumber	7	4	11
meal-feed	6	1	7
money-fx	206	87	293
money-supply	123	28	151
nat-gas	24	12	36
nickel	3	1	4
orange	13	9	22
pet-chem	13	6	19
platinum	1	2	3
potato	2	3	5
reserves	37	12	49
retail	19	1	20
rubber	31	9	40
ship	108	36	144
strategic-metal	9	6	15
sugar	97	25	122
tea	2	3	5
tin	17	10	27
trade	251	75	326
veg-oil	19	11	30
wpi	14	9	23
zinc	8	5	13
Total	6532	2568	9100

Table A.5: Training, Test and Total number of documents per class for the 52 classes with at least one training and one test document of the Reuters-21578 collection, considering documents with a single topic and the standard Mod Apté split — R52 dataset.

A.4 20Ng Dataset

Class	Training	Test	Total
talk.religion.misc	377	251	628
talk.politics.misc	465	310	775
alt.atheism	480	319	799
talk.politics.guns	545	364	909
talk.politics.mideast	564	376	940
comp.os.ms-windows.misc	572	394	966
comp.sys.mac.hardware	578	385	963
comp.graphics	584	389	973
misc.forsale	585	390	975
comp.sys.ibm.pc.hardware	590	392	982
sci.electronics	591	393	984
sci.space	593	394	987
comp.windows.x	593	392	985
rec.autos	594	395	989
sci.med	594	396	990
sci.crypt	595	396	991
rec.sport.baseball	597	397	994
rec.motorcycles	598	398	996
soc.religion.christian	598	398	996
rec.sport.hockey	600	399	999
Total	11293	7528	18821

Table A.6: Training, Test and Total number of documents for each of the 20 classes of the 20-Newsgroups collection, considering the standard Bydate split — 20Ng dataset.

A.5 Cade12 Dataset

Class	Training	Test	Total
01–servicos	5627	2846	8473
02–sociedade	4935	2428	7363
03–lazer	3698	1892	5590
04–informatica	2983	1536	4519
05–saude	2118	1053	3171
06–educacao	1912	944	2856
07–internet	1585	796	2381
08–cultura	1494	643	2137
09–esportes	1277	630	1907
10–noticias	701	381	1082
11–ciencias	569	310	879
12–compras-online	423	202	625
Total	27322	13661	40983

Table A.7: Training, Test and Total number of documents for each of the 12 classes of the Cade collection, considering my random split — Cade12 dataset.

A.6 Pre-Processing the Data

For the Bank collection, I applied the following pre-processing:

1. Substitute TAB, NEWLINE, RETURN and punctuation characters by SPACE.
2. Substitute multiple SPACES by a single SPACE.
3. Turn all letters to lowercase.
4. Substitute accented characters by their unaccented counterparts.
5. Add the subject of each message in the beginning of the message's text.
6. Remove words that are less than 3 characters long.
7. Truncate words that are more than 20 characters long.

For the Webkb collection, I applied the following pre-processing:

1. Remove mime headers using `StripMimeHeaders.java`.
2. Delete the documents corresponding to classes "department", "staff" and "other".
3. Convert from `html` to text using `lynx`.
4. Randomize files so that documents in each class are not in a predefined order.
5. Separate two thirds for training and one third for testing.
6. Substitute TAB, NEWLINE, RETURN and punctuation characters by SPACE.
7. Substitute multiple SPACES by a single SPACE.
8. Turn all letters to lowercase.
9. Remove words that are less than 3 characters long. For example, remove "he" but keep "him".
10. Truncate words that are more than 20 characters long.

For the Reuters-21578 and 20-Newsgroups collections, from the original documents, I applied the following pre-processing:

1. Substitute TAB, NEWLINE, RETURN and punctuation characters by SPACE.
2. Substitute multiple SPACES by a single SPACE.
3. Turn all letters to lowercase.
4. Add the title/subject of each document in the beginning of the document's text.
5. Remove words that are less than 3 characters long.
6. Remove the 524 SMART stopwords. Some of them had already been removed, because they were shorter than 3 characters.
7. Apply Porter's Stemmer to the remaining words.

For the Cade collection, which was already preprocessed when I obtained it, I simply transformed the file containing the terms of the documents to the same format as the other files containing the processed datasets.

All the files for the processed datasets are text files containing one document per line. Each document is composed by its class and its terms. Each document is represented by a "word" representing the document's class, a TAB character and then a sequence of "words" delimited by spaces, representing the terms contained in the document.

Bibliography

Al-Kofahi, K., Tyrrell, A., Vachher, A., Travers, T., and Jackson, P. Combining multiple classifiers for text categorization. In *Proceedings of CIKM-01, 10th ACM International Conference on Information and Knowledge Management*, pages 97–104. ACM Press, New York, US, Atlanta, US, 2001.

B. Schölkopf, A. S., C. Burges, (Editor) *Advances in Kernel Methods: Support Vector Machines*. MIT Press, Cambridge, MA, USA, 1999.

Baeza-Yates, R. and Ribeiro-Neto, B. *Modern Information Retrieval*. Addison-Wesley, Reading, Massachusetts, USA, 1999.

Banerjee, A., Krumpelman, C., Ghosh, J., Basu, S., and Mooney, R. J. Model-based overlapping clustering. In *KDD '05: Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*, pages 532–537. ACM Press, New York, NY, USA, 2005.

Bell, D. A., Guan, J., and Bi, Y. On combining classifier mass functions for text categorization. *IEEE Transactions on Knowledge and Data Engineering*, volume 17(10):pages 1307–1319, 2005.

Bennett, P. N., Dumais, S. T., and Horvitz, E. Probabilistic combination of text classifiers using reliability indicators: models and results. In *Proceedings of SIGIR-02, 25th ACM International Conference on Research and Development in Information Retrieval*, pages 207–214. ACM Press, New York, US, Tampere, FI, 2002.

Bennett, P. N., Dumais, S. T., and Horvitz, E. The combination of text classifiers using reliability indicators. *Information Retrieval*, volume 8(1):pages 67–100, 2005.

Berger, A., Caruana, R., Cohn, D., Freitag, D., and Mittal, V. O. Bridging the lexical chasm: statistical approaches to answer-finding. In *Proceedings of the 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 192–199. Athens, Greece, 2000.

Bi, Y., Bell, D., Wang, H., Guo, G., and Guan, J. Combining multiple classifiers using dempster's rule for text categorization. *Journal of Applied Artificial Intelligence*, volume 21(3):pages 211–239, 2007.

Bilenko, M., Basu, S., and Mooney, R. J. Integrating constraints and metric learning in semi-supervised clustering. In *ICML '04: Proceedings of the twenty-first international conference on Machine learning*, pages 81–88. ACM Press, New York, NY, USA, 2004.

Burges, C. J. C. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, volume 2(2):pages 121–167, 1998.

Calado, P. *Utilização da Estrutura de Ligações para Recuperação de Informação na World Wide Web*. Ph.D. thesis, Universidade Federal de Minas Gerais, Departamento de Ciência da computação, 2004.

Calado, P., Cardoso-Cachopo, A., and Oliveira, A. IR-BASE: An integrated framework for the research and teaching of information retrieval technologies. In *TLIR'07 — First International Workshop on Teaching and Learning of Information Retrieval*. London, UK, 2007.

Cardoso-Cachopo, A. and Oliveira, A. Combining LSI with other classifiers to improve accuracy of single-label text categorization. In *EWLSATEL 2007 — First*

European Workshop on Latent Semantic Analysis in Technology Enhanced Learning. Heerlen, The Netherlands, 2007a.

Cardoso-Cachopo, A. and Oliveira, A. L. An empirical comparison of text categorization methods. In *Proceedings of SPIRE-03, 10th International Symposium on String Processing and Information Retrieval*, pages 183–196. Springer Verlag, Heidelberg, DE, Manaus, BR, 2003. Published in the "Lecture Notes in Computer Science" series, number 2857.

Cardoso-Cachopo, A. and Oliveira, A. L. Empirical evaluation of centroid-based models for single-label text categorization. Technical Report 7/2006, INESC-ID, 2006.

Cardoso-Cachopo, A. and Oliveira, A. L. Semi-supervised single-label text categorization using centroid-based classifiers. In *ACM SAC 2007 — The 22nd Annual ACM Symposium on Applied Computing, Special Track on Information Access and Retrieval (IAR)*, pages 844–851. Seoul, South Korea, 2007b.

Caron, J. Experiments with LSA scoring: Optimal rank and basis. Presented at SIAM Computational Information Retrieval Workshop, 2000.

Caropreso, M. F., Matwin, S., and Sebastiani, F. A learner-independent evaluation of the usefulness of statistical phrases for automated text categorization. In A. G. Chin, (Editor) *Text Databases and Document Management: Theory and Practice*, pages 78–102. Idea Group Publishing, Hershey, US, 2001.

Chang, C.-C. and Lin, C.-J. *LIBSVM: a library for support vector machines*, 2001.

Chuang, W. T., Tiyyagura, A., Yang, J., and Giuffrida, G. A fast algorithm for hierarchical text classification. In *Proceedings of DaWaK-00, 2nd International Conference on Data Warehousing and Knowledge Discovery*, pages 409–418. Springer Verlag, Heidelberg, DE, London, UK, 2000. Published in the "Lecture Notes in Computer Science" series, number 1874.

Cohen, W. W. and Singer, Y. Context-sensitive learning methods for text categorization. *ACM Transactions on Information Systems*, volume 17(2):pages 141–173, 1999.

Cormack, G. V. and Lynam, T. R. Validity and power of t-test for comparing map and gmap. In *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. Amsterdam, The Netherlands, 2007.

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

Creecy, R. M., Masand, B. M., Smith, S. J., and Waltz, D. L. Trading MIPS and memory for knowledge engineering: classifying census returns on the Connection Machine. *Communications of the ACM*, volume 39(1):pages 48–63, 1996.

Cristianini, N. and Shawe-Taylor, J. *An Introduction to Support Vector Machines*. Cambridge University Press, Cambridge, MA, USA, 2000.

Debole, F. and Sebastiani, F. An analysis of the relative hardness of reuters-21578 subsets. *Journal of the American Society for Information Science and Technology*, volume 56(6):pages 584–596, 2004a.

Debole, F. and Sebastiani, F. Supervised term weighting for automated text categorization. In S. Sirmakessis, (Editor) *Text Mining and its Applications*, Number 138 in the "Studies in Fuzziness and Soft Computing" series, pages 81–98. Physica-Verlag, Heidelberg, DE, 2004b.

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

Dempster, A., Laird, N., and Rubin, D. Maximum likelihood from incomplete

data via the em algorithm (with discussion). *Journal of the Royal Statistical Society, Series B*, volume 39:pages 1–38, 1977.

Fung, G. and Mangasarian, O. Semi-supervised support vector machines for unlabeled data classification. In *Optimization Methods and Software, 15*, pages 29–44. 2001.

Furnas, G. W., Deerwester, S. C., Dumais, S. T., Landauer, T. K., Harshman, R. A., Streeter, L., and Lochbaum, K. Information retrieval using a singular value decomposition model of latent semantic structure. In *Proceedings of the 11th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 465–480. Grassau, France, 1988.

Han, E.-H. and Karypis, G. Centroid-based document classification: Analysis and experimental results. In *Principles of Data Mining and Knowledge Discovery*, pages 424–431. 2000.

Han, E.-H., Karypis, G., and Kumar, V. Text categorization using weight-adjusted *k*-nearest neighbor classification. In *Proceedings of PAKDD-01, 5th Pacific-Asia Conferenece on Knowledge Discovery and Data Mining*, pages 53–65. Springer Verlag, Heidelberg, DE, Hong Kong, CN, 2001. Published in the "Lecture Notes in Computer Science" series, number 2035.

Hsu, C. and Lin, C. A comparison of methods for multi-class support vector machines. Technical Report 19, Department of Computer Science and Information Engineering, National Taiwan University, Taipei, Taiwan, 2001.

Hull, D. Using statistical testing in the evaluation of retrieval experiments. In *Proceedings of the 16th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 329–338. Pittsburgh, PA, USA, 1993.

Hull, D. A. Improving text retrieval for the routing problem using latent semantic indexing. In *Proceedings of SIGIR-94, 17th ACM International Conference on Research and Development in Information Retrieval*, pages 282–289. Springer Verlag, Heidelberg, DE, Dublin, IE, 1994.

Hull, D. A., Pedersen, J. O., and Schütze, H. Method combination for document filtering. In *Proceedings of SIGIR-96, 19th ACM International Conference on Research and Development in Information Retrieval*, pages 279–288. ACM Press, New York, US, Zürich, CH, 1996.

Ittner, D. J., Lewis, D. D., and Ahn, D. D. Text categorization of low quality images. In *Proceedings of SDAIR-95, 4th Annual Symposium on Document Analysis and Information Retrieval*, pages 301–315. Las Vegas, US, 1995.

Joachims, T. A probabilistic analysis of the Rocchio algorithm with TFIDF for text categorization. In *Proceedings of ICML-97, 14th International Conference on Machine Learning*, pages 143–151. Morgan Kaufmann Publishers, San Francisco, US, Nashville, US, 1997.

Joachims, T. Text categorization with support vector machines: learning with many relevant features. In *Proceedings of the 10th European Conference on Machine Learning*, pages 137–142. Springer-Verlag, Chemnitz, Germany, 1998a. Published in the "Lecture Notes in Computer Science" series, number 1398.

Joachims, T. Text categorization with support vector machines: learning with many relevant features. In *Proceedings of ECML-98, 10th European Conference on Machine Learning*, pages 137–142. Springer Verlag, Heidelberg, DE, Chemnitz, DE, 1998b. Published in the "Lecture Notes in Computer Science" series, number 1398.

Joachims, T. Transductive inference for text classification using support vector

machines. In *Proceedings of the 16th International Conference on Machine Learning*, pages 200–209. Morgan Kaufmann Publishers, Inc., Bled, Slovenia, 1999a.

Joachims, T. Transductive inference for text classification using support vector machines. In *Proceedings of ICML-99, 16th International Conference on Machine Learning*, pages 200–209. Morgan Kaufmann Publishers, San Francisco, US, Bled, SL, 1999b.

Jones, K. S. and Willett, P., (Editors) *Readings in Information Retrieval*. Morgan Kaufmann Publishers, Inc., Los Altos, USA, 1997.

Koller, D. and Sahami, M. Hierarchically classifying documents using very few words. In *Proceedings of ICML-97, 14th International Conference on Machine Learning*, pages 170–178. Morgan Kaufmann Publishers, San Francisco, US, Nashville, US, 1997.

Koster, C. H. and Seutter, M. Taming wild phrases. In *Proceedings of ECIR-03, 25th European Conference on Information Retrieval*, pages 161–176. Springer Verlag, Heidelberg, DE, Pisa, IT, 2003.

Lam, W. and Lai, K.-Y. A meta-learning approach for text categorization. In *Proceedings of SIGIR-01, 24th ACM International Conference on Research and Development in Information Retrieval*, pages 303–309. ACM Press, New York, US, New Orleans, US, 2001.

Larkey, L. S. and Croft, W. B. Combining classifiers in text categorization. In *Proceedings of SIGIR-96, 19th ACM International Conference on Research and Development in Information Retrieval*, pages 289–297. ACM Press, New York, US, Zürich, CH, 1996.

Lertnattee, V. and Theeramunkong, T. Effect of term distributions on centroid-based text categorization. *Information Sciences*, volume 158(1):pages 89–115, 2004.

Lewis, D. D. An evaluation of phrasal and clustered representations on a text categorization task. In *Proceedings of SIGIR-92, 15th ACM International Conference on Research and Development in Information Retrieval*, pages 37–50. ACM Press, New York, US, Kobenhavn, DK, 1992a.

Lewis, D. D. Text representation for intelligent text retrieval: a classification-oriented view. *Text-based intelligent systems: current research and practice in information extraction and retrieval*, pages 179–197, 1992b.

Lewis, D. D. Naive (Bayes) at forty: The independence assumption in information retrieval. In *Proceedings of ECML-98, 10th European Conference on Machine Learning*, pages 4–15. Springer Verlag, Heidelberg, DE, Chemnitz, DE, 1998. Published in the "Lecture Notes in Computer Science" series, number 1398.

Lewis, D. D. and Jones, K. S. Natural language processing for information retrieval. *Communications of the ACM*, volume 39(1):pages 92–101, 1996.

Lewis, D. D. and Ringuette, M. A comparison of two learning algorithms for text categorization. In *Proceedings of SDAIR-94, 3rd Annual Symposium on Document Analysis and Information Retrieval*, pages 81–93. Las Vegas, US, 1994.

Ling, C., Huang, J., and Zhang, H. Auc: a statistically consistent and more discriminating measure than accuracy. In *Proceedings of IJCAI-2003*, pages 519–526. 2003.

Liu, T.-Y., Yang, Y., Wan, H., Zhou, Q., Gao, B., Zeng, H.-J., Chen, Z., and Ma, W.-Y. An experimental study on large-scale web categorization. In *WWW '05: Special interest tracks and posters of the 14th international conference on World Wide Web*, pages 1106–1107. ACM Press, New York, NY, USA, 2005.

MacQueen, J. Some methods for classifcation and analysis of multivariate observations. In *5th Berkley Symposium on Mathematical Statistics and Probability*, pages 281–297. 1967.

Masand, B., Linoff, G., and Waltz, D. Classifying news stories using memory-based reasoning. In *Proceedings of the 15th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 59–65. ACM Press, Copenhagen, Denmark, 1992.

McCallum, A. K. and Nigam, K. Employing EM in pool-based active learning for text classification. In *Proceedings of ICML-98, 15th International Conference on Machine Learning*, pages 350–358. Morgan Kaufmann Publishers, San Francisco, US, Madison, US, 1998.

Miller, D. J. and Uyar, H. S. A mixture of experts classifier with learning based on both labelled and unlabelled data. In *Advances in Neural Information Processing Systems*, volume 9, pages 571–577. MIT Press, 1997.

Moschitti, A. and Basili, R. Complex linguistic features for text classification: A comprehensive study. In *Proceedings of ECIR-04, 26th European Conference on Information Retrieval Research*, pages 181–196. Springer Verlag, Heidelberg, DE, Sunderland, UK, 2004. Published in the "Lecture Notes in Computer Science" series, number 2997.

Nigam, K., Lafferty, J., and McCallum, A. Using maximum entropy for text classification. In IJCAI-99 Workshop on Machine Learning for Information Filtering, 1999.

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

Ramakrishnan, G., Chitrapura, K. P., Krishnapuram, R., and Bhattacharyya, P. A model for handling approximate, noisy or incomplete labeling in text classification. In *ICML '05: Proceedings of the 22nd international conference on Machine learning*, pages 681–688. ACM Press, New York, NY, USA, 2005.

Robertson, S. E. and Harding, P. Probabilistic automatic indexing by learning from human indexers. *Journal of Documentation*, volume 40(4):pages 264–270, 1984.

Robertson, S. E. and Jones, K. S. Relevance weighting of search terms. *Journal of the American Society for Information Science*, volume 27(3):pages 129–146, 1976. Also reprinted in [Willett, 1988, pages 143–160].

Sable, C. and Church, K. Using bins to empirically estimate term weights for text categorization. In *Proceedings of EMNLP-01, 6th Conference on Empirical Methods in Natural Language Processing*, pages 58–66. Association for Computational Linguistics, Morristown, US, Pittsburgh, US, 2001.

Salton, G. *The SMART Retrieval System*. Prentice-Hall, Inc., New Jersey, USA, 1971.

Salton, G. *Automatic Text Processing: The Transformation Analysis and Retrieval of Information by Computer*. Addison-Wesley, 1989.

Salton, G. and Buckley, C. Term-weighting approaches in automatic text retrieval. *Information Processing and Management*, volume 24(5):pages 513–523, 1988. Also reprinted in [Jones and Willett, 1997, pages 323–328].

Salton, G. and Lesk, M. Computer evaluation of indexing and text processing. *Journal of the ACM*, volume 15(1):pages 8–36, 1968. Also reprinted in [Jones and Willett, 1997, pages 60–84].

Sanderson, M. and Zobel, J. Information retrieval system evaluation: effort, sensitivity, and reliability. In *Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 162–169. Salvador, Brasil, 2005.

Schapire, R. E. and Singer, Y. BOOSTEXTER: a boosting-based system for text categorization. *Machine Learning*, volume 39(2/3):pages 135–168, 2000.

Schütze, H., Hull, D. A., and Pedersen, J. O. A comparison of classifiers and document representations for the routing problem. In *Proceedings of SIGIR-95, 18th ACM International Conference on Research and Development in Information Retrieval*, pages 229–237. ACM Press, New York, US, Seattle, US, 1995.

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

Sebastiani, F. Text categorization. In L. C. Rivero, J. H. Doorn, and V. E. Ferraggine, (Editors) *The Encyclopedia of Database Technologies and Applications*, pages 683–687. Idea Group Publishing, Hershey, US, 2005.

Shah, C. and Croft, W. B. Evaluating high accuracy retrieval techniques. In *Proceedings of the 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 2–9. Sheffield, UK, 2004.

Shahshahani, B. M. and Landgrebe, D. A. The effect of unlabeled samples in reducing the small sample size problem and mitigating the Hughes Phenomenon. *IEEE Transactions on Geoscience and Remote Sensing*, volume 32(5):pages 1087–1095, 1994.

Shankar, S. and Karypis, G. Weight adjustment schemes for a centroid based classifier. 2000.

Sindhwani, V. and Keerthi, S. S. Large scale semi-supervised linear svms. In *Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 477–484. Seattle, Washington, USA, 2006.

Tan, S., Cheng, X., Ghanem, M. M., Wang, B., and Xu, H. A novel refinement approach for text categorization. In *CIKM '05: Proceedings of the 14th ACM international conference on Information and knowledge management*, pages 469–476. ACM Press, New York, NY, USA, 2005a.

Tan, S., Cheng, X., Wang, B., Xu, H., Ghanem, M. M., and Guo, Y. Using drag-pushing to refine centroid text classifiers. In *Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 653–654. Salvador, Brasil, 2005b.

Tsay, J.-J., Wei, Y.-G., and Wang, J.-D. Combining multiple classifiers for automatic text categorization. In *Proceeding of the Taiwan National Computer Symposium*. 2003.

Vapnik, V. *The Nature of Statistical Learning Theory*. Springer-Verlag, Heidelberg, Germany, 1995.

Voorhees, E. M. The TREC-8 Question Answering Track Report. In *Proceedings of the 8th Text REtrieval Conference*, pages 77–82. Gaithersburg, Maryland, USA, 1999.

Weiss, S. M., Apté, C., Damerau, F. J., Johnson, D. E., Oles, F. J., Goetz, T., and Hampp, T. Maximizing text-mining performance. *IEEE Intelligent Systems*, volume 14(4):pages 63–69, 1999.

Wiener, E. D., Pedersen, J. O., and Weigend, A. S. A neural network approach to topic spotting. In *Proceedings of SDAIR-95, 4th Annual Symposium on Document Analysis and Information Retrieval*, pages 317–332. Las Vegas, US, 1995.

Willett, P., (Editor) *Document Retrieval Systems*. Taylor Graham, Cambridge, MA, USA, 1988.

Yang, Y. Expert network: effective and efficient learning from human decisions in text categorisation and retrieval. In *Proceedings of the 17th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 13–22. Springer-Verlag, Dublin, Ireland, 1994.

Yang, Y. and Liu, X. A re-examination of text categorization methods. In *Proceedings of SIGIR-99, 22nd ACM International Conference on Research and Development*

in Information Retrieval, pages 42–49. ACM Press, New York, US, Berkeley, US, 1999.

Yang, Y., Zhang, J., and Kisiel, B. A scalability analysis of classifiers in text categorization. In *Proceedings of SIGIR-03, 26th ACM International Conference on Research and Development in Information Retrieval*, pages 96–103. ACM Press, New York, US, Toronto, CA, 2003.