Studies in Computational Intelligence, Volume 255

Editor-in-Chief
Prof. Janusz Kacprzyk
Systems Research Institute
Polish Academy of Sciences
ul. Newelska 6
01-447 Warsaw
Poland
E-mail: kacprzyk@ibspan.waw.pl

Further volumes of this series can be found on our homepage: springer.com

Vol. 234. Aruna Chakraborty and Amit Konar
Emotional Intelligence, 2009
ISBN 978-3-540-68606-4

Vol. 235. Reiner Onken and Axel Schulte
System-Ergonomic Design of Cognitive Automation, 2009
ISBN 978-3-642-03134-2

Vol. 236. Natalio Krasnogor, Belén Melián-Batista, José A. Moreno-Pérez, J. Marcos Moreno-Vega, and David Pelta (Eds.)
Nature Inspired Cooperative Strategies for Optimization (NICSO 2008), 2009
ISBN 978-3-642-03210-3

Vol. 237. George A. Papadopoulos and Costin Badica (Eds.)
Intelligent Distributed Computing III, 2009
ISBN 978-3-642-03213-4

Vol. 238. Li Niu, Jie Lu, and Guangquan Zhang
Cognition-Driven Decision Support for Business Intelligence, 2009
ISBN 978-3-642-03207-3

Vol. 239. Zong Woo Geem (Ed.)
Harmony Search Algorithms for Structural Design Optimization, 2009
ISBN 978-3-642-03449-7

Vol. 240. Dimitri Plemenos and Georgios Miaoulis (Eds.)
Intelligent Computer Graphics 2009, 2009
ISBN 978-3-642-03451-0

Vol. 241. János Fodor and Janusz Kacprzyk (Eds.)
Aspects of Soft Computing, Intelligent Robotics and Control, 2009
ISBN 978-3-642-03632-3

Vol. 242. Carlos Artemio Coello Coello, Satchidananda Dehuri, and Susmita Ghosh (Eds.)
Swarm Intelligence for Multi-objective Problems in Data Mining, 2009
ISBN 978-3-642-03624-8

Vol. 243. Imre J. Rudas, János Fodor, and Janusz Kacprzyk (Eds.)
Towards Intelligent Engineering and Information Technology, 2009
ISBN 978-3-642-03736-8

Vol. 244. Ngoc Thanh Nguyen, Rados law Piotr Katarzyniak, and Adam Janiak (Eds.)
New Challenges in Computational Collective Intelligence, 2009
ISBN 978-3-642-03957-7

Vol. 245. Oleg Okun and Giorgio Valentini (Eds.)
Applications of Supervised and Unsupervised Ensemble Methods, 2009
ISBN 978-3-642-03998-0

Vol. 246. Thanasis Daradoumis, Santi Caballé, Joan Manuel Marquès, and Fatos Xhafa (Eds.)
Intelligent Collaborative e-Learning Systems and Applications, 2009
ISBN 978-3-642-04000-9

Vol. 247. Monica Bianchini, Marco Maggini, Franco Scarselli, and Lakhmi C. Jain (Eds.)
Innovations in Neural Information Paradigms and Applications, 2009
ISBN 978-3-642-04002-3

Vol. 248. Chee Peng Lim, Lakhmi C. Jain, and Satchidananda Dehuri (Eds.)
Innovations in Swarm Intelligence, 2009
ISBN 978-3-642-04224-9

Vol. 249. Wesam Ashour Barbakh, Ying Wu, and Colin Fyfe
Non-Standard Parameter Adaptation for Exploratory Data Analysis, 2009
ISBN 978-3-642-04004-7

Vol. 250. Raymond Chiong and Sandeep Dhakal (Eds.)
Natural Intelligence for Scheduling, Planning and Packing Problems, 2009
ISBN 978-3-642-04038-2

Vol. 251. Zbigniew W. Ras and William Ribarsky (Eds.)
Advances in Information and Intelligent Systems, 2009
ISBN 978-3-642-04140-2

Vol. 252. Ngoc Thanh Nguyen and Edward Szczerbicki (Eds.)
Intelligent Systems for Knowledge Management, 2009
ISBN 978-3-642-04169-3

Vol. 253. Akitoshi Hanazawa, Tsutom Miki, and Keiichi Horio (Eds.)
Brain-Inspired Information Technology, 2009
ISBN 978-3-642-04024-5

Vol. 254. Kyandoghere Kyamakya, Wolfgang A. Halang, Herwig Unger, Jean Chamberlain Chedjou, Nikolai F. Rulkov, and Zhong Li (Eds.)
Recent Advances in Nonlinear Dynamics and Synchronization, 2009
ISBN 978-3-642-04226-3

Vol. 255. Catarina Silva and Bernardete Ribeiro
Inductive Inference for Large Scale Text Classification, 2009
ISBN 978-3-642-04532-5

Catarina Silva and Bernardete Ribeiro

Inductive Inference for Large Scale Text Classification

Kernel Approaches and Techniques

Springer

Catarina Silva
School of Technology and Management
Polytechnic Institute of Leiria
Alto do Vieiro, 2401-951 Leiria
Portugal
E-mail: catarina@estg.ipleiria.pt

Centre for Informatics and Systems
University of Coimbra
Polo II, 3030-290 Coimbra
Portugal
E-mail: catarina@dei.uc.pt

Bernardete Ribeiro
Centre for Informatics and Systems
Department of Informatics Engineering
University of Coimbra
Polo II, 3030-290 Coimbra
Portugal
E-mail: bribeiro@dei.uc.pt

ISBN 978-3-642-04532-5 e-ISBN 978-3-642-04533-2

DOI 10.1007/978-3-642-04533-2

Studies in Computational Intelligence ISSN 1860-949X

Library of Congress Control Number: Applied for

Typeset & *Cover Design:* Scientific Publishing Services Pvt. Ltd., Chennai, India.

Printed in acid-free paper

9 8 7 6 5 4 3 2 1

springer.com

To Nuno and Miguel
To my family
Catarina Silva

To Miguel and Alexander
To my family
Bernardete Ribeiro

Preface

Motivation and Scope

Text classification is becoming a crucial task to analysts in different areas. In the last few decades the production of textual documents in digital form has increased exponentially. Their applications range from web pages to scientific documents, including emails, news and books. Searching for a digital text in Google is now more than a reality, it is a commonplace. In the near future, with the advent of intelligent text classification methods, people will have even more access to a large variety of enhanced digital text services, viz. filtering, searching and filing.

Despite the widespread use of digital texts, handling them is inherently difficult - the large amount of data necessary to represent them and the subjectivity of classification complicate matters. Earlier research has addressed the extraction of information from relatively small collections of well-structured documents such as news wires and scientific publications.

Intelligent text classification methods, which rely heavily on machine learning algorithms, have the potential to supersede existing information retrieval techniques and provide superior facilities that will save time and money for users and companies, while providing a vital tool for dealing with the proliferation of digital texts they are faced with.

The book is based on the PhD thesis of the first author and gives a concise view on how to use kernel approaches for inductive inference in large scale text classification; it presents a series of new techniques to enhance, scale and distribute text classification tasks. The approaches combine high performance and robustness with an extensive assessment of the suggested techniques.

The book is not intended to be a comprehensive survey of the state-of-the-art of the whole field of text classification. Its purpose is less ambitious and more practical: to explain and illustrate some of the important methods used in this field, in particular kernel approaches and techniques.

Challenges and Contributions

The relevant kernel approaches and techniques used to respond to some of the most prominent challenges are unfolded in this book, covering several facets of the whole problem.

Automatic text classification is an effervescent field of research. Many scientific and industrial fields generate enormous amounts of text data, such as news wires, microarray gene data and web pages. This trend seems to be spreading and there is no end in sight. Users are overwhelmed with the amount of information and thus need efficient, reliable and mostly intelligible text classification methods that they can relate to and understand. Another great challenge is the available knowledge that can be integrated by users and engineers in processing, learning and evaluation procedures.

From an academic point of view, putting together a probabilistic formulation which integrates the underlying knowledge of the problem at hand can also present an immense challenge. Acknowledging there is no free lunch i.e. that any two algorithms are equivalent when their performance is averaged across all possible problems, research can instead be focused on methods to tackle their inherent high dimensionality.

From an engineering perspective, dealing with the curse of dimensionality of text representations and learning is an interesting problem.

The content of this book sets out the importance of the challenges mentioned above and explains and illustrates the main approaches and techniques for dealing with them. The most relevant original contributions of this book are related to the challenges described above:

- **Empirical Evaluation of Text Pre-processing Methods.** We have undertaken an empirical study to compare the influence and relative importance of standard pre-processing dimensionality reduction methods in text classification performance. Low frequency word removal, stopword removal and stemming were tested and stopword removal was found to be the most useful technique to apply since it yields the best performing classifiers in all tested conditions. Stemming also plays an important role, especially in the precision of the classifiers. While stopword removal significantly alters the content of input data, stemming only alters its shape, i.e. the reduction of information is not significant. We can therefore say that stemming is more relevant in terms of efficiency of the learning machine (there is less redundancy in data). Low frequency word removal has little influence on classification performance, but it can decrease training complexity by reducing the number of features. A general conclusion is that the evaluated dimensionality reduction techniques can strengthen classifier performance.

- **Knowledge Integration in SVMs.** We have investigated the introduction of unlabeled data in the support vector machine (SVM) learning stage and the potential of using several learning machines organized in a

committee. We have presented two margin-based approaches to introduce unlabeled document information into the learning stage: background knowledge and active learning. We have also proposed an SVM ensemble with a two-step learning strategy using the separating margin as a differentiating factor in positive classifications.

Both the proposed enhancements to SVMs in text classification integrate new knowledge in the learning procedures and show improvements over the baseline SVMs. The separating margin plays a crucial role in both techniques and can be used in further enhancements.

- **Reducing the Dimensionality of RVMs.** Due to the poor scaling capabilities of the relevance vector machine (RVM) algorithm when faced with high-dimensional data sets, like text classification training sets, it is crucial to limit the training set dimension to a minimum. We have examined ways to reduce the dimensionality, viz. active learning and similitude measure between terms. We introduced an active learning RVM method based on the kernel trick. Using a kernel distance metric we have defined a higher-dimensional space where active examples are selected. Complexity escalation was controlled since the number of added documents was fixed and the kernel trick provides a simple strategy to determine those active documents. To reduce the number documents in the training set, we presented a two-step RVM : the first stage selects which training documents go to the next level, using a similitude measure between documents, based on the co-occurrence of words; the second step gathers all remaining documents and infers an RVM classifier. While maintaining the RVMs' sparseness, we still show competitive accuracy, as long as training examples are carefully established.

- **Divide-and-Conquer RVM Approaches.** To keep pursuing the scalability of RVMs, we have focused on three divide-and-conquer RVM methods: incremental, boosting and ensemble strategies. These methods rely on a selection of small working chunks from the training set and then explore different combining strategies that permit the use of all training examples in RVM expansion to large datasets. We demonstrated that it is possible to make use of an RVM's advantages, such as predictive distributions for testing instances and sparse solutions, while maintaining and even improving the classification performance. The proposed methods adapt RVMs to large scale text sets, maintaining their probabilistic Bayesian nature and providing sparse solutions.

- **Hybrid SVM-RVM.** SVMs and RVMs constitute two state-of-the-art learning machines that are currently the focus of cutting-edge research. SVMs present accuracy and complexity preponderance, but are surpassed by RVMs when probabilistic outputs or kernel selection come into the discussion. We have proposed a two-level hierarchical hybrid SVM-RVM

model to combine the best of both learning machines. The first level of the proposed model uses an RVM to determine the less confident classified examples and the second level uses an SVM to learn and classify the tougher examples. The hierarchical approach outperforms both baseline learning machines.

- **Deployment in Distributed Environments.** In cooperation with researchers from the Laboratory of Adaptive Systems and Parallel Processing of the University of Ljubljana, Slovenia, we have deployed text classification in distributed environments. This work was carried out both in a cluster in the University of Ljubljana, Slovenia, and in the Center for Informatics and Systems of the University of Coimbra, Portugal. The proposed deployment employs a combination of the text classification task (and data) decomposition, configuration evaluation through the modeling of the design phases, and a high performance distributed computing model. We have shown that it is not only possible, but also advantageous to deploy text classification in a cluster environment, while using available middleware distributed platforms and existing sequential code.

- **Text Classification Framework.** In the last chapter of the book, we propose and discuss a framework for inductive inference-based text classification using kernels methods. The main components of this development, beyond the generally good performances of kernel methods on real-world problems and ease of use provided by current implementations, involve active learning, ensembles, incremental learning, boosting and knowledge integration. Finally, some especially promising lines of work will open windows to further research in the field.

Plan and Organization

This book has six chapters and two appendices. The chapters are organized into two parts: the first part relating to fundamental topics encloses the first two chapters and the second part concerning approaches and techniques includes the other four chapters.

Chapter 1 contains background material on text classification. In particular we review document corpora, representations, reduction methods, classifiers, and evaluation techniques.

Chapter 2 introduces the concept of kernel methods, and summarizes into a single framework the foundations of two paradigmatic techniques: support vector machines and relevance vector machines. Both approaches are introduced in a text classification perspective, along with results and comparisons of their application to benchmark corpora.

Chapter 3 discusses the learning techniques developed to integrate knowledge in the classification task in order to improve the performance of support vector machines (SVMs) in text classification applications.

Chapter 4 explores relevance vector machines (RVMs) and their application to text classification. We propose new approaches to tackle RVMs' scaling problems. In particular we examine techniques that reduce the dimensionality of the problem and we introduce incremental, boosting and ensemble divide-and-conquer strategies. Finally, a hybrid RVM-SVM combination is presented that substantially improves baseline results.

Chapter 5 describes the deployment of text classification in cluster environments, using a distributed system to optimize the procedures involved. In this Chapter we look at various ways to deploy complete text classification systems in distributed environments, employing a combination of the text classification task (and data) decomposition, configuration evaluation through the modeling and the design phases, and a high performance distributed computing model.

In the final Chapter (6) we propose a framework for inductive inference text classification and present a unified view of the field across four stages of design: sources, preprocessing, learning and evaluation. We outline each of the framework phases in a coherent result which can be a guide to real-world applications. In addition we present research trends for future work with special focus in the area of web applications and information technology.

Audience

The book is designed for practitioners and researchers and is suitable for postgraduate students in computer science, engineering, information technology and other related disciplines. Some knowledge in the area of machine learning and computational intelligence will be beneficial.

Acknowledgments

We would like to acknowledge and thank all those who have contributed to bringing this book to publication for their help, support, and input.

We would also like to acknowledge and thank Professor Andrej Dobnikar and Dr. Uroš Lotrič and everyone else in the Laboratory for Adaptive Systems and Parallel Processing of the Faculty of Computer and Information Science, University of Ljubljana, Slovenia, especially Dr. Branko Šter, for the fruitful cooperation and stimulating discussions on distributed text classification systems. They have made an invaluable contribution to the book.

We also wish to thank the support of the School of Technology and Management of the Polytechnic Institute of Leiria and of the Centre of Informatics and Systems of the Informatics Engineering Department, Faculty of Science and Technologies, University of Coimbra, for the means provided during the research.

Our thanks also to Jean Burrows who reviewed the syntactic aspects of the book.

Our special thanks and appreciation to our editor, Professor Janusz Kacprzyk, of Studies in Computational Intelligence, Springer, for his essential encouragement.

Lastly, to our families and friends for all their love and support.

July 2009 Catarina Silva
Coimbra, Portugal Bernardete Ribeiro

Contents

Acronyms

ARD	Automatic Relevance Determination
AUC	Area Under the Curve
BEP	Break-Even Point
BOW	Bag-Of-Words
DAG	Direct Acyclic Graph
DF	Document Frequency
DNF	Disjunctive Normal Form
EM	Expectation-Maximization
ERM	Empirical Risk Minimization
FN, c	False Negative
FNR	False Negative Rate
FP, b	False Positive
FPR	False Positive Rate
FTP	File Transfer Protocol
HTC	High Throughput Computing
HTTP	Hypertext Transfer Protocol
IDF	Inverse Document Frequency
K-NN	K-Nearest Neighbour
KDA	Kernel Discriminant Analysis
KPCA	Kernel - PCA
LDA	Linear Discriminant Analysis
LR	Logistic Regression
LSI	Latent Semantic Indexing
MeSH	Medical Subject Headings
NASA	National Aeronautics and Space Administration
NB	Naïve Bayes
NE	Named-entities
NLP	Natural Language Processing
NN	Neural Network
OSH	Optimal Separating Hyperplane
P	Precision

PC	Personal Computer
PCA	Principal Components Analysis
POS	Part-of-speech
QBC	Query-By-Committee
R	Recall
RBF	Radial Basis Function
RCV1	Reuters Corpus Volume 1
ROC	Receiver Operating Characteristic
RV	Relevance Vector
RVM	Relevance Vector Machine
SETI	Search for ExtraTerrestrial Intelligence
SGML	Standard Generalized Markup Language
SRM	Structural Risk Minimization
SV	Support Vector
SVM	Support Vector Machine
TF	Term Frequency
TFIDF	Term Frequency - Inverse Document Frequency
TN, d	True Negative
TP, a	True Positive
TSVM	Transductive SVM
VC-dimension	Vapnik-Chervonenkis dimension

Notation

$\|.\|$	Number of elements in a set
$\|\|.\|\|$	$L_2 - norm$
$[.]$	Integer part of .
$.^T$	Transpose
α	Lagrange multipler in SVMs optimization
α	Precision hyperparameter in RVMs optimization
$\boldsymbol{\alpha}$	Set (vector) of α
c_j	Category
$\mathscr{C}$	Set of categories
C	Regularization parameter in SVMs
b	bias
$\mathbf{d}, \mathbf{d}_i$	Document
$\mathscr{D}$	Set of documents
$F()$	Distribution function
$F\beta, F1$	van Rijsbergen's measure
h	Hypothesis
H	Hypothesis space
$i.i.d.$	Independent and identically distributed
$k(.), \phi(.), \Phi(.)$	Kernel functions
λ, β, γ	Parameters for Rocchio's method
Neg	Negative examples, not belonging to a category
$O(.)$	Order of
p	Number of computing nodes
$P(.)$	Probability
$p(.)$	Probability function
$p(.\|.)$	Conditional probability
$\boldsymbol{\Psi}$	Design matrix
Pos	Positive examples, belonging to a category
$\boldsymbol{\varphi}$	Set of RVs
$\boldsymbol{\varphi}_i$	RV
q	Query

$\mathbf{Q}$	Hessian matrix
ρ	Separating margin of SVMs
s_i	Complexity parameters for phases in distributed environment
S_{ij}	Similitude measure between documents $\mathbf{d}_i$ and $\mathbf{d}_j$
$\sigma(.)$	Sigmoid function
t	Target
τ_{task}	Time to complete a task
$\mathbf{u}_j$	Unlabeled document
$\mathbf{U}$	Set of unlabeled documents
$\mathscr{W}$	Set of features (words or terms), dictionary
w_k	Word or term
w_{ik}	Value representing word w_k in a document $\mathbf{d}_i$
ω_k	Weight of term w_k for a given model
ω_{ik}	Weight of term w_k in document $\mathbf{d}_i$ for a given model
ω_{qk}	Weight of term w_k in query q
$\boldsymbol{\omega}$	Set of weights that define a model
y	Output of a model

Chapter 1
Background on Text Classification

Abstract. In this chapter background material for studying text classification problems is presented along with the notation used throughout the book. After describing the problem, a summary of typical applications is given and document representation issues are introduced followed by commonly used pre-processing steps, including dimensionality reduction. Next, state-of-the-art classifiers for text classification are briefly reviewed with current achievements, followed by some widely accepted performance evaluation metrics and benchmarks.

To determine the influence and relative importance of pre-processing methods in text classification performance an empirical study was carried out to compare dimensionality reduction techniques, using standard learning machines and benchmarks. Results and analysis of this study are reported and finally the conclusions on the relative success of the several pre-processing, learning and evaluation approaches are presented.

1.1 Problem Setting

The text classification task consists of learning models for a given set of classes and applying these models to new unseen documents for class assignment. It is mainly a supervised classification task, where a training set consisting of documents with previously assigned classes is provided, and a testing set is used to evaluate the models. Text classification is illustrated in Figure 1.1, including the pre-processing steps (document representation and space reduction/feature extraction) and the learning/evaluation procedure (support vector machines, neural networks, genetic algorithms, etc.). Great relevance has been rightfully given to learning procedures in text classification. However, there must be a pre-processing stage before the learning process, and this usually represents up to 80% of both time and computational efforts. Pre-processing alters the input space used to represent the documents that are ultimately included in the training and

C. Silva and B. Ribeiro: Inductive Inference for Large Scale Text Classification, SCI 255, pp. 3–29.
springerlink.com

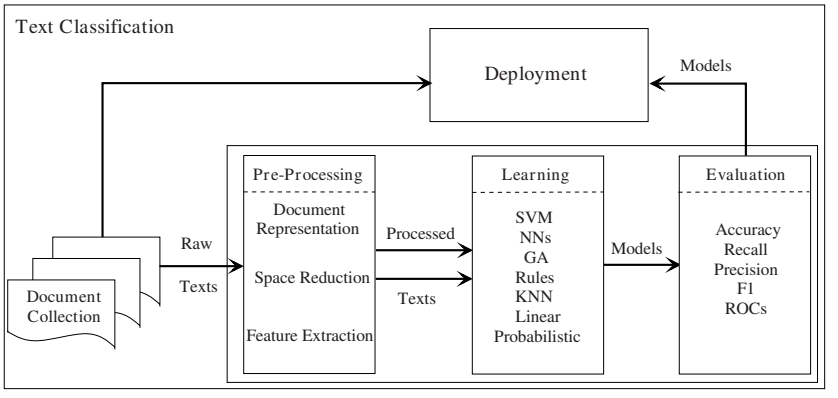

Fig. 1.1 Text classification overview.

testing sets, used by machine learning algorithms to learn classifiers, which then are evaluated.

Text classification may be formalized as the task of approximating the unknown target function $f : \mathscr{D} \times \mathscr{C} \to \{-1,1\}$ that corresponds to how documents would be classified by an authoritative expert. The function f is the text classifier, $\mathscr{C} = \{c_1, c_2, \ldots, c_j, \ldots, c_{|\mathscr{C}|}\}$ is a predefined set of categories and $\mathscr{D}$ is a set of documents. Each document is represented using the set of features, usually words, $\mathscr{W} = \{w_1, w_2, \ldots, w_k, \ldots, w_{|\mathscr{W}|}\}$, with each one as a vector $\mathbf{d}_i = (w_{i1}, w_{i2}, \ldots, w_{ik}, \ldots, w_{i|\mathscr{W}|})$, where w_{ik} describes each feature's representation for that specific document. When $f(\mathbf{d}_i, c_j) = 1$, $\mathbf{d}_i$ is a positive example or member of category c_j, whilst when $f(\mathbf{d}_i, c_j) = -1$ it is a negative example of c_j.

Text classification can be a subjective problem, since the label(s) or category(ies) that an expert can attribute to a document may vary with the purpose of the classification and personal experience. For instance, a news article on the crash of a sports team's stocks due to unfair government policy can be classified under *Sports*, or under *Politics*, or under *Markets*, or under any combination of the three. The purpose of a machine learning approach is to capture this subjectivity by examining the documents classified by the expert under $\mathscr{C}$.

Text classification is usually a binary problem, as mentioned earlier, but there are also multiclass approaches. Depending on the application, text classification may be either a single-label task (i.e. exactly one $c_j \in \mathscr{C}$ must be assigned to each $d_i \in \mathscr{D}$), or a multi-label task (i.e. any number between 0 and $|\mathscr{C}|$ of categories may be assigned to each document $d_i \in \mathscr{D}$). However, most multi-label, multiclass tasks are usually tackled as $|\mathscr{C}|$ independent binary classification problems under $\{c_j, \overline{c}_j\}$, for $j = 1, \ldots, |\mathscr{C}|$. In this case, a classifier for $\mathscr{C}$ is thus actually composed of $|\mathscr{C}|$ one-against-all binary classifiers.

In the rest of this book we use classification and categorization synonymously. Similarly, the terms class, label, or category are used interchangeably, as well as word, term, example or feature, and dataset, corpus or collection.

1.2 Applications of Text Classification

Text classification applications are spreading rapidly due to an ever-increasing amount of digital textual data. Algorithms are faced with large amounts of text data to organize in a given set of categories. Documents can take several forms, e.g. web pages, emails, newswire or scientific articles. Here a sample of the most significant applications is introduced.

1.2.1 Document Organization

The general problem of document organization is the one of classifying or filing a set of text documents into a set of categories.

A typical example is the classification of news stories in a news agency [45]. In publishing houses, like the international Reuters or Portuguese Lusa, a large number of news stories arrive each day. It is of utmost importance that there is at least a semi-automated system that tags these documents within a set of predefined categories. The challenges include the online nature of the system and the variation in the set of categories.

Another example is patent analysis, where submissions of new patent applications are increasing. The task is to determine if a given patent is unique or if it has already been stated. The challenges are the large number of patents that already exist, the longer than usual size of documents and the, sometimes deliberately, technical, non-standard, and abstruse vocabulary used by applicants [63].

1.2.2 Text Filtering

Text filtering includes applications in which a set of texts should be filtered out before reaching a user or application. Both the structures of such texts and the motives for filtering them vary widely.

The best known example is anti-spam filtering for e-mails [5]. At a user's personal computer or at a centralized server, a spam filter should identify and filter out (or at least mark) the unsolicited e-mail. Usually, machine learning techniques to analyze e-mail elements, i.e. subject, sender and message, cooperate with hand-crafted rules and black lists[1], for each e-mail element.

Another growing application is filters of unsuitable content. Mostly these filters are employed for children's safe use of the internet, but more can be found each day on Internet service providers to filter illegal sites.

1.2.3 Word Sense Disambiguation

Word sense disambiguation is to determine the sense of a particular ambiguous word occurrence, viz. to disambiguate both polysemous and homonymous words. For

[1] A black list is a list of items of e-mail elements that is checked for every e-mail. Any e-mail containing one of those items is classified as spam.

instance, the word chip may have (at least) two different senses in English, as in the processor chip (an electronic device in computers) or potato chip (a very popular snack). It may be seen as a text classification task, where the contexts of occurrence are the documents and the senses of interpretation are the categories [40].

1.2.4 Other Applications

Some important applications of text classification were briefly reviewed. The boundaries between the different classes of applications listed here are blurred, and some may be considered special cases of others.

Other noteworthy applications of text classification, but less relevant to the subject of this book, include authorship attribution, hierarchical categorization of web pages, speech categorization, image categorization using captions and automated essay grading.

1.3 Document Representation

Typically, text documents are unstructured data. Before learning, one must transform them into a representation that is suitable for computing.

Once the features in the documents, usually words or terms, are extracted, each document is represented in a vector space, also known as the bag-of-words (BOW), widely used in information retrieval [9]. This representation method is equivalent to an attribute value representation used in machine learning [90]. Each dimension of this space represents a single feature, whose importance in the document corresponds to the exact distance from the origin. Documents are thus points (vectors) in a $|\mathscr{W}|$-dimensional vector space, where $|\mathscr{W}|$ denotes the dimension of the vocabulary or dictionary, $\mathscr{W} = \{w_1, w_2, \ldots, w_k, \ldots, w_{|\mathscr{W}|}\}$, with unique terms or features, w_k, as components. Representation of features is very important as it constitutes the basis for most classification algorithms. Every document is represented as a vector $\mathbf{d}_i = (w_{i1}, w_{i2}, \ldots, w_{ik}, \ldots, w_{i|\mathscr{W}|})$, where w_{ik} describes each feature, word in the dictionary, for the document. Algorithms work on these document vectors and the classification task aims at distinguishing documents of one class from the others, so there must be significant differences in the vectors of different documents [41].

The simplest representation uses the binary event model, where if a feature $w_k \in \mathscr{W}$ appears in document $\mathbf{d}_i$, then the k^{th} component w_{ik} is 1, otherwise it is 0. This representation can be replaced by the number of times the word occurs in the document (TF - term frequency), $TF(w_k)$, illustrated in Figure 1.2, or some variation of its value, thereby ignoring the sequence in which words occur.

Since the TF representation can bias the document vector towards features with more occurrences, the TF representation can be modified by the significance or rarity with which the feature w_k occurs in all documents of the corpus. The document frequency (DF) of a word, $DF(w_k)$, is the number of documents in the collection in which the word w_k occurs. The inverse document frequency or $IDF(w_k)$ is

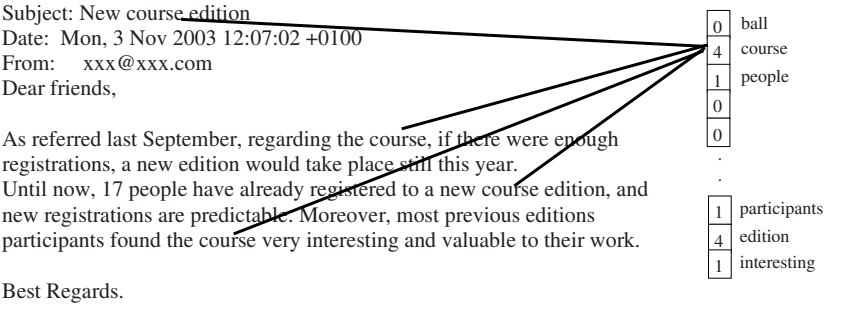

Fig. 1.2 Term frequency representation of a text document.

$$IDF(w_k) = \frac{|\mathscr{D}|}{DF(w_k)}, \tag{1.1}$$

where $|\mathscr{D}|$ is the number of documents in the collection. The $IDF(w_k)$ of a term w_k decreases with the number of documents it appears in [26]. Vector components are weighted according to the $IDF(w_k)$ of the corresponding term. Usually some monotonous function of the $IDF(w_k)$, such as the logarithm or the square root, is used instead of the $IDF(w_k)$ itself, to avoid amplifying the importance of multiple occurrence of terms [121]. This is known as the TFIDF representation system defined by (1.2) and it will be used throughout this book, unless otherwise stated.

$$TFIDF(w_k) = TF(w_k) \times \log(IDF(w_k)). \tag{1.2}$$

Finally, all documents in the collection are mapped to a matrix called the term by document matrix representing the feature space (see Figure 1.3).

	w_1	w_2	$\cdots$	w_k	$\cdots$	$w_{|\mathscr{W}|}$
$\mathbf{d}_1$	w_{11}	w_{12}	$\cdots$	w_{1k}	$\cdots$	$w_{1|\mathscr{W}|}$
$\mathbf{d}_2$	w_{21}	w_{22}	$\cdots$	w_{2k}	$\cdots$	$w_{2|\mathscr{W}|}$
$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$
$\mathbf{d}_i$	w_{i1}	w_{i2}	$\cdots$	w_{ik}	$\cdots$	$w_{i|\mathscr{W}|}$
$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$
$\mathbf{d}_{|\mathscr{D}|}$	$w_{|\mathscr{D}|1}$	$w_{|\mathscr{D}|2}$	$\cdots$	$w_{|\mathscr{D}|k}$	$\cdots$	$w_{|\mathscr{D}||\mathscr{W}|}$

Fig. 1.3 Term by document matrix representing a document collection.

Each row of the matrix corresponds to a document. The columns of the matrix correspond to the unique terms in the document collection. Each intersection (w_{ik}) represents the TFIDF weight of term w_k in document $\mathbf{d}_i$.

1.4 Pre-processing Text

Once the representation of texts has been defined, the high dimension of the feature or term space may pose some problems for most learning machines used in text classification. These problems occur not only for computational reasons, but also due to overfitting, i.e. tuning the classifier to contingent characteristics of the data set. When a classifier is overfitting, it is able to classify the data on which it has been trained, but is unable to generalize to previously unseen data. Therefore, a number of pre-processing steps are required before a learning machine can infer a classifier. They are usually referred to as dimensionality reduction methods, since their goal is to reduce the size of the vector space, controlling the computational time involved, whilst maintaining or improving performance. However, removing terms can be harmful and care must be taken, since potentially useful information may be removed.

These techniques can be divided into two types: feature selection and feature extraction. When using selection methods, the idea is to determine a reduced set of the available terms (or their representation) in a document. On the other hand, extraction methods aim at generating a new set of terms from the original terms that constitute the document. Regardless of the approach used, there are a number of issues to be considered, in particular, the time spent in reduction, the information that may be lost, the learning time reduction and the classification improvement. A reduction technique should only be applied if the combined evaluation of these issues is positive.

This section will explore dimensionality reduction techniques used in text classification algorithms. Later in this chapter an empirical comparison of the most representative methods will be presented.

1.4.1 Feature Selection

Feature selection methods aim at choosing from the available set of terms a smaller set that more efficiently represents the documents. Feature selection is not needed for all classification algorithms as some classifiers are capable of feature selection themselves. However for other classifiers feature selection is mandatory, since a large number of irrelevant features can significantly impair classifier accuracy.

Selection methods can be divided into filter methods and wrapper methods, and they will be explored in the rest of this section.

1.4.1.1 Filter Methods

The filtering approach is the simpler alternative, and it is based on keeping the terms that receive a higher score according to a function that measures the relative importance of a term [122].

Every document $\mathbf{d}_i$ contains all the words, spaces, markups, and tags which occur in it. In a straightforward approach, a term is any space-separated word in a document.[2] Taking any news article or random page from the web, not all the words in the document are useful for text classification. Some applications, including web search, rely heavily on the markup or link structure of documents, but text classification tasks usually consider only the text portions of the page.

Stopword removal can be considered a basic filter method. Stopwords are non-informative words, such as articles, prepositions and conjunctions that appear in the documents. A stopword list is usually built with words that should be filtered in the document representation process. In this case, the filter function will return a positive value (for instance) only if a word does not belong to the stopword list. Words that are to be included in the stopword list are language and task dependent. Besides the obvious reduction in the number of features, there may be another potential advantage associated with stopword removal, since some stopwords can mislead the learning machine in defining non-existent correlations between documents. The standard stopword list used for English language is the SMART list[3].

Another simple but efficient technique is to consider the document frequency of a term. Only terms that occur in the highest number of documents are retained, with terms that appear in very few documents being filtered out, and terms that occur only a small number of times being removed [6, 164, 33, 75].

Other more sophisticated filter feature selection typically employs some statistical measures over the training corpus and ranks features in order of their amount of information (correlation) with respect to the class labels of the classification task at hand. The typical measures used to rank feature lists are information gain [71, 93, 164, 91], mutual information with the class label [164, 33, 108], χ^2 [120, 163], and other such measures [4].

After the feature set has been ranked, only the top few features are retained (typically in the order of hundreds or a few thousand). Typical large text corpora contain tens to hundreds of thousands of unique features.

1.4.1.2 Wrapper Methods

The wrapper approach is more demanding computationally. The feature subset is selected using the same learning algorithm that will be used for learning the classifier [91]. Using this technique the feature subset selection algorithm exists as a wrapper around the induction algorithm [56]. The feature selection algorithm

[2] This is a valid definition for Portuguese or English texts, but not for languages using composite nouns, like German or Finnish.

[3] ftp://ftp.cs.cornell.edu/pub/smart/english.stop

conducts a search for a good subset using the induction algorithm itself as part of the evaluation function.

In the wrapper approach the reduced term set is generated by either adding or removing a term. Each new set is tested on a validation set using the chosen learning algorithm. The best performing set is elected, resulting in a training set tuned with the learning machine.

The major drawback of wrapper methods, when applied to high-dimensional feature spaces like text classification, is its the computational complexity that makes them cost-prohibitive for standard applications.

1.4.1.3 Natural Language Processing

In the approaches described so far, the bag-of-words (BOW) model was always considered as starting point. However, there have been some attempts to move forward from the BOW model. A fruitful field of research is natural language processing (NLP).

Using the BOW model, words like 'New' and 'York' will occur as separate features instead of the intuitive single feature 'New York'. A lot of natural language processing (NLP) research is devoted to detecting such phrases or named-entities (NE) in text documents [162, 170]. This is a first step in moving from the BOW model to more intelligent models for feature selection in documents. Part-of-speech (POS) tagging is also employed to generate POS n-grams as other useful features [155]. All these NLP derived feature sets have been used in text classification but surprisingly none have been found to significantly improve the simple BOW model [41].

1.4.2 Feature Extraction

Feature extraction methods aim at generating a new set of (synthetic) terms from the original terms that constitute the document. The rationale for using synthetic (rather than naturally occurring) terms is that, due to the pervasive problems of polysemy, homonymy, and synonymy, the original terms may not be optimal dimensions for document content representation [122].

To extract terms, one first has to devise a method to extract the new terms from the old ones, and then convert the initial document representation using the new synthetic terms. Two types of feature extraction can be used in text classification: term clustering, and latent semantic indexing.

1.4.2.1 Term Clustering

Term clustering aims at grouping (or clustering) terms with high similitude, usually according to some semantic point of view, and then replacing the terms in the cluster by a unified representative term, e.g. the cluster centroid.

The simplest technique that can be cast into this category is stemming, where a word stem is derived from the occurrence of a word by removing case and inflection

information. For example “viewer”,“view”, and “preview” are all contracted to the same stem “view”. Stemming does not significantly alter the information included in document representation, but it does avoid feature expansion. The Porter stemmer [101] for English is freely available in many forms and widely used in text classification.

Initial attempts at term clustering include the reciprocal nearest neighbor clustering [67], consisting of creating clusters of two terms that are the most alike according to some measure of similarity. More recently, supervised term clustering has been pursued, by [10] for example, with their distributional clustering method, where clusters are used together with those terms that tend to indicate the presence of the same category, or group of categories. In this case, the elected classifier was the naïve Bayes classifier (see Section 1.5.3).

1.4.2.2 Latent Semantic Indexing

Latent semantic indexing (LSI) [29] comes from information retrieval research to address problems derived from the use of synonymous, and polysemous words in documents.

LSI takes advantage of implicit higher-order structure in the association of terms with documents, i.e. their semantic structure, in order to improve the detection of relevant documents. The technique used is singular-value decomposition, in which a large term by document matrix is decomposed into a set of orthogonal factors from which the original matrix can be approximated by linear combination. In other words, LSI compresses document vectors into vectors of a lower-dimensional space whose dimensions are obtained as combinations of the original dimensions by looking at their patterns of co-occurrence.

In text classification this technique is applied by deriving the mapping function from the training set and then applying it to training and testing documents alike.

A drawback of LSI, though, is that if an original term is particularly good in itself at discriminating a category, that discrimination power may be lost in the new vector space [122]. Another issue is that the new dimensions representing the documents, unlike other dimensionality reduction techniques described so far, are not intuitively interpretable.

However, LSI works well in bringing out the *latent* semantic structure of the vocabulary used in the corpus. In fact, if there is a large number of terms which all contribute a small amount of critical information, then the combination of evidence is a major problem for a term-based classifier [120]. LSI is able to correctly classify a document even if the exact expected words are not present, but have a pattern of co-occurence with words that do in fact appear in the document.

1.5 Classifiers

Initial classifiers in the 1980s consisted of using knowledge engineering techniques to manually build an expert system capable of making text classification decisions.

Such an expert system would typically consist of a set of manually defined logical rules.

In the 1990s, successful machine learning approaches were introduced, using a general inductive procedure to automatically build a classifier for a category by observing the characteristics of a set of documents that had previously been classified manually by a human expert. This is supervised learning, where the engineering effort goes towards the construction not of a specific classifier, but of a way to automatically build a classifier from labeled examples [122]. Many different types of supervised learners have been used in text classification, including probabilistic naïve Bayesian methods [33], Bayesian networks [149], decision trees [69], decision rules [21], neural networks, incremental or batch methods for learning linear classifiers, example-based methods [90], classifier ensembles (including boosting methods [38]), and support vector machines [151].

Training methods for machine learning classifiers are often characterized as being generative or discriminative. Generative classifiers learn a model of the joint probability, $P(\mathbf{d}_i, c_j)$, of the input document $\mathbf{d}_i$ and the class label c_j, make their predictions by using Bayes' rule to calculate the conditional probability $P(c_j|\mathbf{d}_i)$, and then pick the most likely label c_j. In contrast, discriminative classifiers do not have a probability framework and can thus be interpreted as modeling the posterior $P(c_j|\mathbf{d}_i)$ directly, thus providing an approximation. It has often been argued that for many application domains, discriminative classifiers achieve higher testing set accuracy than generative classifiers [152]. Nonetheless, generative classifiers also have several advantages, like methods for handling missing data, and they often perform better for small training set sizes. Specifically, it has been shown that a simple generative classifier (naïve Bayes) outperforms its conditionally-trained, discriminative counterpart (logistic regression) when the amount of available labeled training data is small [95].

Generative methods include naïve Bayes, Latent Dirichlet Allocation, the Aspect model, and BayesANIL. Discriminative methods are typified by support vector machines (SVM), logistic regression (LR), decision trees and ensembles (like AdaBoost).

In the following a short overview of learning methods for text classification is presented, namely Rocchio's method, decision trees and rules, naïve Bayes, k-nearest neighbor, neural networks , kernel-based machines and also committee based techniques and active learning methods.

The foundations of kernel-based machines, i.e. support vector machines (SVMs) and relevance vector machines (RVMs), are left for the next chapter since they constitute the starting point of the techniques proposed.

1.5.1 Rocchio's Method

Rocchio's method [106] is a classic method for document routing or filtering that originated from information retrieval tradition, initially used for expanding user queries on the basis of relevance judgments. It is used for inducing linear, profile-style classifiers, i.e. linear classifiers that consist of an explicit profile

(or prototypical document) of the category. This has obvious advantages in terms of interpretability, because a profile of this type can be readily understood by a human.

This method relies on an adaptation to text classification of the well-known Rocchio formula for relevance feedback in the vector-space model. This adaptation was first proposed in [48], and has been used by many authors since then [122] either as the main subject of research [51, 114], baseline classifier [23, 52, 72, 113] or member of a classifier committee [64].

Using this method, each term w_k is assigned a weight ω_k, that combines the term weight in the original query q with the weights of both judged relevant (*Pos*) and irrelevant (*Neg*) documents for the category:

$$\omega_k = \lambda\ \omega_{qk} + \beta \sum_{i \in Pos} \frac{\omega_{ik}}{|Pos|} - \gamma \sum_{i \in Neg} \frac{\omega_{ik}}{|Neg|}, \tag{1.3}$$

where ω_{ik} is the weight or representation of term w_k in document $\mathbf{d}_i$, *Pos* are the training documents belonging to the category, whereas *Neg* are the training documents not belonging to the category and $|.|$ represents the number of elements of the set. In the context of text classification there is no initial query, so the term weight in the original query, ω_{qk}, is not considered, setting $\lambda = 0$ [33]. β and γ are control parameters that allow setting the relative importance of positive and negative examples.

A classifier built using Rocchio's method rewards the closeness of a testing document to the centroid of the positive training examples, and its distance from the centroid of the negative training examples. The role of negative examples is usually deemphasized, by setting β to a high value and γ to a low one [122]. This method is quite easy to implement, and is also quite efficient, since learning a classifier basically comes down to averaging weights. In terms of effectiveness, however, a drawback is that if the documents in the category tend to occur in disjoint clusters (e.g., a set of documents labeled with *Biology* category and dealing with either *Animals* or *Vegetables*), such a classifier may miss most of them, as the centroid of these documents may fall completely outside these two clusters. More generally, Rocchio's classifiers, like all linear classifiers, have the disadvantage of dividing the space of documents linearly.

1.5.2 Decision Trees and Rules

Inductive rule learners and decision tree learners are the most important examples of symbolic (nonnumeric) algorithms, which are usually praised for their ease of interpretability by humans.

The internal nodes of a decision tree for text classification are labeled by words or terms. The branches departing from each node are labeled by tests on the weight that the term has in the testing document. The leaves represent the categories according to the branches followed. To classify a testing document $\mathbf{d}_i$, starting from the root

of the decision tree, each term weight in the document is tested to determine which branch to follow, until a leaf node is reached. The category label of this leaf node is then assigned to the document $\mathbf{d}_i$.

To construct a decision tree, a *divide and conquer* strategy is usually followed by selecting a term w_k, partitioning the training set into classes of documents that have the same value (or range of values) for w_k, and placing each such class in a separate subtree. The process is recursively repeated on the subtrees until each leaf of the tree so generated contains the training examples assigned to the same category, which is then chosen as the label for the leaf [122]. To determine which term w_k is used to make the partition an information gain or entropy criterion is generally applied. However, a complete tree built by this method may be prone to overfitting, as some branches may be too specific to the training data. Most decision tree learning methods thus include a method for growing the tree and one for pruning it, that is, for removing the overly specific branches. Decision tree text classifiers have been used either as the main classification tool [69, 71], or as baseline classifiers [23, 52], or as members of classifier committees [75, 113, 158].

Decision trees and decision rules are similar in that both can encode any Boolean function. A decision rule classifier built by an inductive rule learning method consists of a DNF (Disjunctive Normal Form) rule, that is, of a conditional rule with a premise in DNF. The literals in the premise denote the presence (non-negated keyword) or absence (negated keyword) of the keyword in the testing document $\mathbf{d}_i$, while the clause head denotes the decision to classify $\mathbf{d}_i$ under the category.

An advantage of DNF rule learners is that they tend to generate more compact classifiers than decision trees learners. Rule learning methods usually attempt to select from all the possible covering rules (i.e. rules that correctly classify all the training examples) the one that minimizes some criterion. While decision trees are typically built by a top-down strategy, DNF rules are often built in a bottom-up fashion.

To create a set of decision rules, each training document $\mathbf{d}_i$ is considered as a complex clause with reference to the presence or absence of every word or term, leading to the conclusion of a category or its negation, according to whether $\mathbf{d}_i$ is a positive or negative example of the category. This complex representation is highly prone to overfitting, which is generally mitigated with simplifications (e.g., removing premises from clauses, or merging clauses) that maximize its compactness, while at the same time not affecting the covering property of the classifier. Furthermore, at the end of this process, a pruning phase similar in spirit to that employed in decision trees is applied, where the ability to correctly classify all the training examples is traded for more generality.

DNF rule learners vary widely in terms of the methods, heuristics and criteria employed for generalization and pruning. Among the DNF rule learners that have been applied to text classification are CHARADE [92], DL-ESC [74], RIPPER [21, 22, 23], SCAR [93], and SWAP-1 [6].

1.5.3 Naïve Bayes

Naïve Bayes is a probabilistic classifier that determines the category of a document using the probability of a document $\mathbf{d}_i = (w_{i1}, w_{i2}, \ldots, w_{ik}, \ldots, w_{iV})$ belonging to category c_j, $P(c_j|\mathbf{d}_i)$. To determine this probability, Bayes' theorem is applied

$$P(c_j|\mathbf{d}_i) = \frac{P(c_j)P(\mathbf{d}_i|c_j)}{P(\mathbf{d}_i)}, \tag{1.4}$$

where $P(\mathbf{d}_i)$ represents the probability that a randomly picked document has vector $\mathbf{d}_i$ as its representation and $P(c_j)$ is the probability that a randomly picked document belongs to category c_j. The estimation of $P(\mathbf{d}_i|c_j)$ and $P(\mathbf{d}_i)$ is troublesome, since the number of possible document vectors $\mathbf{d}_i$ is very large. This latter however does not differ between categories and can be neglected, since it can be interpreted only as a scale factor. To compute $P(\mathbf{d}_i|c_j)$, this classifier makes the naïve assumption behind its denomination, i.e. that any two features of the document vector are, when viewed as random variables, statistically independent of each other [122], or, mathematically:

$$P(\mathbf{d}_i|c_j) = \prod_{k=1}^{|\mathscr{W}|} P(w_{ik}|c_j). \tag{1.5}$$

Applying these two simplifications we get

$$P(c_j|\mathbf{d}_i) = P(c_j) \prod_{k=1}^{|\mathscr{W}|} P(w_{ik}|c_j). \tag{1.6}$$

An estimate of $P(c_j)$ can be calculated from the fraction of training documents that is assigned to class c_j.

Despite the fact that the naïve assumption of independence is generally not true for word or features appearing in documents, the naïve Bayes classifier is surprisingly effective [1].

1.5.4 K-Nearest Neighbor

K-nearest neighbor (k-NN) is an example-based classifier, i.e. it does not build an explicit, declarative representation of the category c_j, but relies on the category labels attached to the training documents similar to the testing document. Methods of this type have thus been called lazy learners, since they defer the decision on how to generalize beyond the training data until each new query instance is encountered [90]. Consequently k-NN does not have an offline training phase. The main computation is the online scoring of training documents, given a testing document, in order to find the k nearest neighbors.

To classify an unknown document $\mathbf{d}_i$, the k-NN algorithm [31] ranks the document's neighbors among the training document vectors, and uses the class label of the k most similar neighbors to predict the class of the testing document. The classes

of these neighbors are weighted using the similarity of each neighbor to $\mathbf{d}_i$, where the similarity may be measured by, for example, the Euclidean distance or the cosine between the two document vectors. Note that k-NN, unlike linear classifiers, does not divide the document space linearly, and hence does not suffer from the problems discussed in Section 1.5.1. However, k-NN's most important drawback is its inefficiency at classification time: while, for example, with a linear classifier only a dot product needs to be computed to classify a testing document, k-NN requires the entire training set to be ranked for similarity with the testing document, which is much more expensive. This is a drawback of lazy learning methods, since they do not have a true training phase and thus defer all the computation to classification time.

1.5.5 Neural Networks

Neural networks (NNs) have wide appeal for many researchers due to their closeness to the structure of the brain, a characteristic not shared by other classifiers. In an analogy to the brain, an entity made up of interconnected neurons, NNs are made up of interconnected processing elements called units, which respond in parallel to a set of input signals given to each. The unit is the equivalent of its brain counterpart, the neuron, and the connections resemble the synapses that occur between biological neurons.

An NN text classifier is a network of units, where the input units represent words or terms, the output unit(s) represent the category or categories of interest, and the weights on the edges connecting units represent dependence relations [122].

Figure 1.4 depicts the most usual setting: a feedforward neural network with one hidden layer. When a new testing document $\mathbf{d}_i$ is to be classified, its terms, or their representations, are loaded into the NN through the input units, then they are fed forward through the different layers, where they are weighted, summed and, before being fed to the next layer, they go through a normally non-linear activation function f, for instance a sigmoid function.

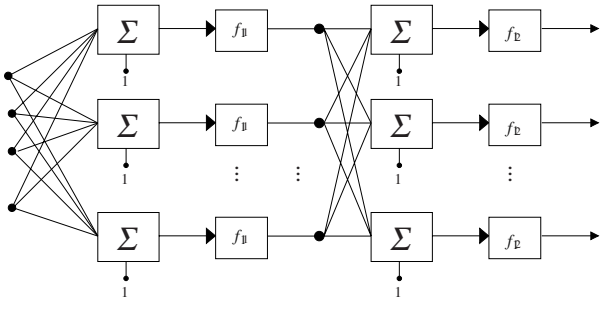

Fig. 1.4 Feedforward neural network with one hidden layer.

The value of the output unit(s) determines the categorization decision(s). A typical way of training NNs is backpropagation, whereby the term weights of a training document are loaded into the input units, as just described, and if a misclassification occurs the error is *backpropagated* so as to change the parameters of the network and eliminate or reduce the error. More details on NNs and backpropagation can be found in [44].

Several authors have studied the application of both linear and non-linear NNs in text classification. A recurrent problem is the huge number of inputs, which is usually overcome with elaborated feature selection methods as in [96], and sometimes using *thesauri* as in [28].

Other types of linear NN classifiers implementing a form of logistic regression have also been proposed and tested in [120, 159], showing interesting alternatives. Hierarchical approaches like [109] also show promising results.

Nonlinear NNs supposedly have the capacity to learn higher-order interactions between terms. However, comparative experiments relating nonlinear NNs to their linear counterparts have been performed, and the former have yielded either no improvement [120] or very small improvements [159] over the latter [122]. A feasible rationale for this result is the fact that text classification is most often a linearly separable problem, despite its high dimensionality.

1.5.6 Kernel-Based Learning Machines

Kernel-based learning machines or kernel methods are state-of-the-art learning machines for text classification and constitute the main focus of this book. Chapter 2 will therefore describe the kernel machines used, viz. support vector machines (SVMs) [151] and relevance vector machines (RVMs) [147]. Nevertheless, a short introduction is given here, to make this brief review of classifiers more complete.

Kernel-based learning machines approach the classification problem by mapping the data into a high-dimensional feature space, where each coordinate corresponds to one feature of the data items, transforming the data into a set of points in a Euclidean space. In that space, a variety of methods can be used to find relations in the data. Since the mapping can be quite general (not necessarily linear, for example), the relations found this way are accordingly very general. This approach is called the kernel trick.

Kernel-based learning machines owe their name to the use of kernel functions, that enable them to operate in the feature space without ever computing the coordinates of the data in that space, but instead by simply computing the inner products between the images of all pairs of data in the feature space. This operation is often computationally cheaper than the explicit computation of the coordinates.

Algorithms capable of operating with kernels include SVMs, RVMs, kernel discriminant analysis (KDA), kernel canonical correlation analysis, kernel principal components analysis (KPCA), kernel ridge regression, spectral clustering, along with many others.

SVMs were introduced in text classification by [52] and have subsequently been used by several others [33, 62, 30, 163, 32].

1.5.7 Committees

Ensemble based systems (also known under various other names, such as multiple classifier systems, committee of classifiers, or mixture of experts) have shown favorable results compared to those of single-expert systems for a broad range of applications requiring automated decision making under a variety of scenarios.

In matters of great importance that have financial, medical, or other implications, people often seek a second opinion before making a decision, sometimes more. In doing so, they analyze each one, and combine them using some implicit process to reach a final decision that is apparently the best informed one. This process of consulting several experts before making a final decision is perhaps second nature to us; yet, the extensive benefits of such a process in classification systems is still being discovered by the computational intelligence community [100]. In [60], a sample of the vast literature on classifier combination can be found, on both the theory and implementation of ensemble based classifiers.

There are several theoretical and practical reasons why one may prefer an ensemble system [100]:

- Statistical reasons: Good performance in training data does not guarantee good generalization performance. Classifiers with similar training performances may have different generalization capabilities. In such situations, combining the outputs of several classifiers may reduce the risk of selecting a poorly performing classifier. The ensemble may not surpass the best individual classifier, but will reduce the overall risk of making a particularly poor selection;
- Large volumes of data: In some applications the amount of data to be analyzed can be too large to be handled effectively by a single classifier; partitioning the data into smaller subsets, training different classifiers with different partitions of data, and combining their outputs can often be a more efficient approach;
- Too little data: Ensemble systems can also be used to address exactly the opposite problem of having too little data. Data are the support of a good classifier. When there is no adequate training data, resampling techniques can be used to define random subsets of the available data, each of which can be used to train a different classifier, creating the ensemble. These approaches have also proven to be very effective;
- Divide and conquer: When the problem at hand is too complex for a single classifier to solve, i.e. when the decision boundary that separates data from different classes lies outside the space of functions that can be implemented by the chosen classifier model, using several classifiers can be helpful. A decision based on the majority voting of a sufficient number of such classifiers can easily learn a more complex boundary and this approach can be cast as a divide-and-conquer technique;
- Data fusion: When data is obtained from different sources, where the nature of features are different (heterogeneous features), a single classifier cannot be used to learn the information contained in all of the data. In this case an ensemble based approach can be used to merge the data.

Classifier committees or ensembles are based on the idea that, given a task that requires expert knowledge, k experts may perform better than one, if their individual judgments are appropriately combined. A classifier committee is then characterized by (i) a choice of k classifiers, and (ii) a choice of a combination function [122], usually denominated voting algorithm. The classifiers should be as independent as possible to guarantee a large number of inductions on the data. To this end, the representation of texts can be different, but it is quite usual for the classifiers or at least their parameters to be diverse.

The simplest voting algorithm is the majority voting one, where each base classifier (expert) votes on the class the document should belong to and the majority wins (in two-class problems an odd number of classifiers should be used). A straightforward evolution is the weighted linear combination, where each classifier has different weight in the final decision. In this case the weights can be fixed or adaptive.

In fact, voting algorithms can be divided into two types: those that adaptively change the distribution of the training set based on the performance of previous classifiers and those that do not. Boosting is the standard example of the first type and bagging of the second type, hence a brief overview of boosting and bagging comes next.

1.5.7.1 Boosting

Boosting is a machine learning meta-algorithm for performing supervised learning. Boosting is based on the question: can a set of weak learners create a single strong learner? A weak learner is defined as a classifier which is only slightly correlated with the true classification. In contrast, a strong learner is a classifier that is arbitrarily well-correlated with the true classification. The main idea of boosting is thus to generate many relatively weak classification rules and to combine them into a single highly accurate classification rule [113]. The boosting algorithm assigns different importance weights to different training examples. The algorithm proceeds by incrementally assigning increasing significance to examples which are hard to classify, while easier training examples get lower weights. This weight strategy is the basis of the weak learners evaluation. The final combined hypothesis classifies a new testing example by computing the prediction of each of the weak hypotheses and taking a vote on these predictions.

For text classification, AdaBoost [38] is the standard boosting algorithm. It starts with a set of input-target pairs, i.e. training documents with associated labels usually defined by a human expert. Initially the importance weights of the examples are uniformly distributed. Then the AdaBoost algorithm repeatedly retrieves *weak hypotheses* and these are evaluated and used to determine the final hypothesis. On each iteration, using the set of importance weights determined in the previous iteration, the hypothesis error is computed and the weight or importance of that *weak classifier*, is determined, assigning larger weights to *good classifiers*, i.e. classifiers with low error, whereas lower weights (even with negative values) are assigned for *bad classifiers*.

1.5.7.2 Bagging

Bootstrap aggregating (bagging) is a meta-algorithm to improve classification and regression models in terms of stability and classification accuracy. Bagging also reduces variance and helps to avoid overfitting. Although this method is usually applied to decision tree models, it can be used with any type of model. The bagging algorithm [17] votes on classifiers generated by different bootstrap samples (replicates). A bootstrap sample is generated by uniformly sampling m instances from the training set with replacement [34]. Several bootstrap samples are generated and a classifier is built from each bootstrap sample. A final classifier is built from the individual bootstrap classifiers, defining its output as the class predicted most often by its sub-classifiers.

For a given bootstrap sample, an instance in the training set has probability $1-(1-\frac{1}{m})^m$ of being selected at least once in the m times instances randomly selected from the training set. For large m, this is about $1-\frac{1}{e}=63.2\%$, which means that each bootstrap sample contains only about 63.2% unique instances from the training set [122]. This perturbation causes different classifiers to be built if the inducer is unstable (e.g., neural networks, decision trees) and the performance can improve if the induced classifiers are good and not correlated; however, bagging may slightly degrade the performance of stable algorithms (e.g., k-NN) because effectively smaller training sets are used for training each classifier [11].

1.5.8 Active Learning

Active learning in its most general sense refers to any form of learning wherein the learning algorithm has some degree of control over the examples on which it is trained [76]. There are situations in which unlabeled data is abundant but labeling data is expensive or the learning machine is not able to deal with the deluge of training examples, for algorithmic or computational reasons. In such scenarios, the learning algorithm can actively ask the supervisor for labels. As the learner chooses the examples, the number of examples to learn a concept can often be much lower than the number required in normal supervised learning. However, with this approach there is a risk that the algorithm might focus on unimportant or even invalid examples.

There have been some promising results in the active learning area. In [24], the theory for an active learning method called selective sampling is presented and then applied to some small to moderate sized problems as a demonstration of its the viability.

Lewis and Gale developed a method called uncertainty sampling, which is conceptually similar to selective sampling, but which is specifically meant for use in text classification. Their method selects for labeling those documents whose membership is most unclear by using an approximation based on Bayes' rule, certain independence assumptions, and logistic regression [70].

While approaches and results vary, these and other studies have concluded that active learning greatly improves learning efficiency by reducing the number of labeled examples used [27, 39].

1.5.9 Other Methods

Throughout this section an effort was made to present an overview of the learning approaches proposed in the text classification literature. Nevertheless, we believe it is not possible to give a complete survey, despite the helpful references, not only because the spatial context does not allow it, but also because some approaches are hybrid systems that are not covered by any specific umbrella. Among these, the ones most worth mentioning are those based on Bayesian inference networks [33, 149], genetic algorithms [20, 85], maximum entropy modeling [84], and fuzzy rules [43].

1.6 Evaluation

The common metric of interest in evaluating classifiers is the accuracy, i.e. what fraction of documents belonging to known classes are correctly assigned to those classes. The standard way of doing this is to take a labeled portion of the corpus as the training data, and use the remaining fraction of labeled data as testing data. The procedure is to use training data for model learning and evaluating the performance in the testing data - the known labels of testing data are hidden and compared against label predictions from the system.

Standard benchmark corpora have predefined training/testing splits to allow for comparison of approaches. Otherwise, a typical training/testing split of a labeled corpus is the ratio 70 : 30. Sometimes a part of labeled data (called validation data) is also held aside for tuning purposes. The system is first trained on training data, and then tuned on the validation data, before being tested against the testing data. A typical ratio for training, validation, and testing data is 60 : 20 : 20.

Experiments are often repeated with different random splits into training, validation, and testing datasets. Usually the corpus is randomly split and evaluated from 5 to 30 times and the mean and variance of the criteria are reported. Techniques like cross-validation and k-fold validation help guard against randomness in particular data splits and allow sounder results. K-fold validation involves splitting the data in k parts, and using $(k-1)$ parts for training and the remaining part for testing. This is repeated k times, considering all possible testing sets one at a time. Average results of k runs are reported. Refer to Haykin [44] for details on cross-validation.

When comparing two algorithms, experiments are performed multiple times and statistical tests like the paired t-test can be used to qualitatively judge whether one algorithm is better than the other.

In the rest of this section, performance criteria for evaluating text classification performance are introduced along with some commonly used benchmark corpora.

1.6.1 Performance Criteria

The text classification problem is usually broken into several one-against-all binary classification problems, as mentioned earlier. Multi-class approaches are also subject of research, but the results achieved are yet not comparable to binary approaches.

In order to evaluate a text classification binary decision task, a contingency matrix representing the possible outcomes of the classification should be first defined as shown in Table 1.1. When an example is positive (Positive Class) it can be assigned a positive or a negative label by the classifier, resulting in a True Positive (a) or a False Negative (c) respectively. When an example is negative (Negative Class) it can also be assigned a positive or a negative label by the classifier, resulting this time in a False Positive (b) or a True Negative (d) respectively. Several measures have been defined based on this contingency table. Table 1.2 shows the most relevant with their formulas.

Table 1.1 Contingency table for binary classification.

	Positive Class	Negative Class
Assigned Positive	a	b
	(TP - True Positives)	(FP - False Positives)
Assigned Negative	c	d
	(FN - False Negatives)	(TN - True Negatives)

Table 1.2 Performance measures for binary classification.

Measure	Formula
Accuracy	$\frac{a+d}{a+b+c+d}$
Error Rate	$\frac{b+c}{a+b+c+d}$
Precision (P)	$\frac{a}{a+b}$
Recall (R) / Sensitivity/ TP rate	$\frac{a}{a+c}$
Specificity / TN rate	$\frac{d}{b+d}$
FP rate	$\frac{b}{b+d}$
FN rate	$\frac{c}{a+c}$

None of the measures is perfect or even appropriate for every problem. For example, recall, if used alone might show deceptive results, e.g., a system that classifies all testing examples as belonging to a given category will show perfect recall, since measure c (False Negatives) in Table 1.1 will be zero, making recall ($\frac{a}{a+c}$) reach its maximum. Accuracy (1-Error Rate), on the other hand works well if the number of positive and negative examples is balanced, but in extreme conditions it might be deceptive too. If the number of negative examples is overwhelming compared with the positive examples, as in text classification, then a system that assigns no documents to a category, i.e. classifies all as negative examples of that category, will obtain an accuracy value close to 1.

For text classification, precision and recall are usually preferred. However, they provide partial views on errors and are to some extent complementary, since precision puts emphasis on false positives and recall draws attention to false negatives. As a result, measures that combine precision and recall are often employed, such as the break-even point (BEP) and the $F\beta$ measure. BEP was proposed by Lewis [72] and is defined as the point at which recall equals precision. van Rijsbergen's $F\beta$ measure [150] combines recall (R) and precision (P) into a single score:

$$F\beta = \frac{(\beta^2+1)P \times R}{\beta^2 P + R} = \frac{(\beta^2+1)a}{(\beta^2+1)a + b + \beta^2 c}. \tag{1.7}$$

$F0$ is the same as Precision, $F\infty$ is the same as Recall. Intermediate values between 0 and ∞ correspond to different weights assigned to recall and precision. The most common values assigned to β are 0.5 (recall is half as important as precision), 1.0 (recall and precision are equally important) and 2.0 (recall is twice as important as precision). Drawing our attention to Table 1.1 and Equation (1.7), if TP, FP and FN are all 0, $F\beta$ is defined as 1 (this occurs when a classifier assigns no documents to the category and there are no related documents in the collection).

As pointed out by [113], the BEP has some drawbacks. Usually the value of the BEP has to be interpolated. If the values of recall and precision are too far apart then the BEP will show values that are not achievable by the system. Also the point where recall equals precision is not informative and not necessarily desirable from the user's perspective [108]. van Rijsbergen's $F\beta$ measure is the best suited metric, but still has the drawback that it might be difficult for the user to define the relative importance of recall and precision. Usually F1 is used, representing a harmonic mean of precision and recall, (1.8).

$$F1 = \frac{2 \times P \times R}{P + R}. \tag{1.8}$$

In general, the F1 performance (or the performance of any other measure) is reported as an average value for all the categories of a given corpus. There are two ways of computing this average: macro-average, and micro-average. With macro-average, the F1 value is computed for each category and these are averaged to get the final macro-averaged F1. With micro-average, we first obtain the global values for every class for the true positive, true negative, false positive, and false negative measures and then compute the micro-averaged F1 value using micro-recall and micro-precision (computed with these global values). In this book the results are determined using macro-averaging, unless otherwise stated.

ROC (receiver operating characteristic) curves are two-dimensional graphs that offer a visual comparison of binary classifiers. ROC curves plot the TP rate on the y axis vs. FP rate on the x axis (see Table 1.2), by varying the threshold of decision between positive and negative classifications. Taking as example an output decision range in $[-1, 1]$, the threshold is usually set to zero, i.e. testing examples with outputs less than zero are classified as negative, and the rest are classified as positive. However, to build a ROC curve this threshold is varied step-wise from -1 to 1 and,

for each cut-off point, the TP rate and FP rate are calculated and plotted, defining the curve. A ROC curve depicts relative trade-offs between benefits (true positives) and costs (false positives) [37]. The purpose is to develop classifiers with high a TP rate and low FP rate, i.e. on the northwest corner of a ROC curve. Additionally, the AUC (area under the curve) of a ROC can also be used to quantify and compare the performance of classifiers. If instead of varying the threshold (or cut-off point), we use a single value, the curve gives way to a point, constituting an alternative to one-to-one classifier comparisons.

1.6.2 Document Corpora

Standard text benchmark datasets, also referred to as corpora or collections, are used in text classification research for evaluation. These corpora are high-dimensional sets of documents, manually classified for business or research purposes, constituting multi-class, multi-labeled problems, i.e. there are more than two classes and each document can be classified in several classes. The text classification problem is usually divided into several one-against-all binary problems, one for each class, and some form of averaging is used to achieve a final performance (see Section 1.6.1). In this section some of the common benchmark datasets are reviewed. Their complete characteristics are left for the appendixes.

1.6.2.1 Reuters-21578

The Reuters-21578 Text Categorization Test collection[4] is the most popularly used benchmark text dataset. It contains over 100 classes distributed in a set of SGML[5] files. It is a financial corpus of news articles averaging 200 words each. The dataset is a multi-class multi-labeled dataset and it is a very heterogeneous corpus, since the number of documents assigned to each category is very variable. There are documents not assigned to any of the categories and others assigned to more than 10 categories. On the other hand the number of documents assigned to each category is not homogeneous either. There are categories with only one assigned document and others with thousands of assigned documents.

Various researchers have used many standard subsets of this dataset. The standard ModApte split, described in Appendix A, was adopted in this book, using 75% of the articles (9603 items) for training and 25% (3,299 items) for testing. In particular people have used only the most populous ten classes with at least a minimum specified number of training documents, i.e. the classes earn, acquisitions, money-fx, grain, crude, trade, interest, ship, wheat and corn, presented in Table 1.3 with the corresponding number of positive training and testing examples. These 10 categories are widely accepted as a benchmark, since 75% of the documents belong to at least one of them. Figure 1.5 presents an example of a document in the collection. Detailed information about Reuters-21578 corpus can be found in Appendix A.

[4] http://www.daviddlewis.com/resources/testcollections/reuters21578/

[5] Standard Generalized Markup Language.

Table 1.3 Positive documents for Reuters-21578 categories.

Category	Training	Testing
Earn	2,715	1,044
Acquisitions	1,547	680
Money-fx	496	161
Grain	395	138
Crude	358	176
Trade	346	113
Interest	313	121
Ship	186	89
Wheat	194	66
Corn	164	52

```
<REUTERS TOPICS="YES" LEWISSPLIT="TRAIN" CGISPLIT="TRAINING-SET"
OLDID="5552" NEWID="9">
<DATE>26-FEB-1987 15:17:11.20</DATE>
<TOPICS><D>earn</D></TOPICS>
<PLACES><D>usa</D></PLACES>
<PEOPLE></PEOPLE>
<ORGS></ORGS>
<EXCHANGES></EXCHANGES>
<COMPANIES></COMPANIES>
<UNKNOWN>
&#5;&#5;&#5;F
&#22;&#22;&#1;f0762&#31;reute
r f BC-CHAMPION-PRODUCTS-&lt;CH   02-26 0067</UNKNOWN>
<TEXT>&#2;
<TITLE>CHAMPION PRODUCTS &lt;CH> APPROVES STOCK SPLIT</TITLE>
<DATELINE>    ROCHESTER, N.Y., Feb 26 - </DATELINE><BODY>Champion
Products Inc said its
board of directors approved a two-for-one stock split of its
common shares for shareholders of record as of April 1, 1987.
    The company also said its board voted to recommend to
shareholders at the annual meeting April 23 an increase in the
authorized capital stock from five mln to 25 mln shares.
 Reuter
&#3;</BODY></TEXT>
</REUTERS>
```

Fig. 1.5 Reuters-21578 document.

1.6.2.2 Reuters Corpus Volume I

The Reuters corpus volume 1[6] (RCV1) is an archive of 806,791 English language news documents. The documents are similar to the Reuters-21578 document presented in Figure 1.5. Despite its popularity, Reuters-21578 presents a number of disadvantages, particularly that of overall size (only 21,578 documents). By contrast, RCV1 is 35 times larger. Reuters-21578 covers only a fraction of a year, with somewhat inconsistent coverage of that time period, whereas RCV1 covers a complete year of editorial output, allowing the investigation of temporal issues. In addition, RCV1 was created from a news archive product (i.e. a database) rather than

[6] http://trec.nist.gov/data/reuters/reuters.html

a raw newswire feed, which helps to ensure consistency (fewer duplicates, corrections, brief alerts, etc.). However, one advantage of the older collection is that a great deal of effort has been applied to identify suitable training/testing splits for various applications, particularly those of text categorization and machine learning. In fact, Reuters-21578 constitutes a benchmark for algorithm comparison within the research community, while work on the new corpus is still developing.

Categories for RCV1 are hierarchically organized into four top-level nodes: Corporate/Industrial (CCAT), Economics (ECAT), Government/ Social (GCAT) and Markets (MCAT). There are 103 categories actually assigned to the data. In the second column of Table 1.4 (Documents) the total number of positive documents for the ten most populated classes of RCV1 is given. For training, RCV1 defines 22,370 documents that should be used. For testing we selected the first 50,000 documents not used for training. Table 1.4 also presents the effective number of positive documents used for training and testing. Detailed information about RCV1 corpus can be found in Lewis' paramount article [73] and in Appendix B.

Table 1.4 Positive documents for RCV1 categories.

Category	Documents	Training	Testing
CCAT	381,327	10,416	23,077
GCAT	239,267	2,050	5,180
MCAT	204,820	5,154	11,110
C15	151,785	3,122	7,454
ECAT	119,920	3,162	7,539
M14	85,440	1,799	4,887
C151	81,890	515	698
C152	73,092	1,088	435
GPOL	56,878	1,627	3,756
M13	53,634	1,095	2,613

1.7 Empirical Evaluation of Pre-processing Methods

To further assess the influence and relative importance of pre-processing methods on text classification performance, an empirical study to promote their comparison was carried out. This study used the Reuters-21578 collection with the ModApte split. The experiments were carried out using the SVM classifier. These choices of benchmark corpus and classifier correspond to the most widely used benchmark and the state-of-the art classifier in text classification, thus permitting the deduction of significant results. In order to fulfill the objectives described, three pre-processing methods for dimensionality reduction were considered, namely:

- Low frequency word removal: whether or not words that appeared in fewer than three documents are removed;
- Stopword removal: the removal or not of words in the SMART stopword list;
- Stemming: whether Porter stemming was applied or not.

Table 1.5 presents the possible combinations of the tested dimensionality reduction methods. Table 1.6 presents macro-averaged results for accuracy, precision, recall and F1, achieved for the defined combinations of pre-processing methods [126, 127].

Table 1.5 Testing conditions of pre-processing methods.

Test	Low freq.words	Stopwords	Stemming
A	No	No	No
B	No	No	Yes
C	No	Yes	No
D	No	Yes	Yes
E	Yes	No	No
F	Yes	No	Yes
G	Yes	Yes	No
H	Yes	Yes	Yes

Table 1.6 Performances for different pre-processing methods.

Test	Accuracy	Precision	Recall	F1
A	96.98%	83.96%	55.66%	65.59%
B	97.01%	84.23%	55.92%	66.14%
C	96.54%	84.06%	62.81%	71.26%
D	97.30%	83.95%	62.54%	71.09%
E	97.00%	84.93%	58.05%	67.74%
F	97.05%	85.05%	56.05%	66.27%
G	97.32%	85.05%	63.06%	71.75%
H	97.31%	85.91%	62.54%	71.77%

Comparing the results it is clear that, there are only slight differences where accuracy and precision are concerned. This leads to the assertion that the pre-processing methods tested do not greatly influence these values. However, in text classification tasks, recall values are usually more sensitive due to the distribution of positive and negative examples. Usually the number of examples is large, but the number of positive examples is small (less than 5% in Reuters-21578 case). Hence, if a learning machine classifies all documents as not belonging to a category, it will still have a large accuracy, making the false negatives an issue to be considered. Table 1.7 shows the average number of false negatives and false positives for each combination of pre-processing methods. As expected, the false positive values do not show great divergence, while false negative exhibit substantial differences. The worst (highest) false negative values correspond to situations where stopword removal was not carried out, especially tests A, B, but also tests E and F, suggesting that preserving those words can be harmful to recall values in text classification.

Table 1.7 False Positives and False Negatives for different pre-processing methods.

Test	False Positives	False Negatives
A	22.9	62.9
B	22.4	62.5
C	23.3	55.0
D	22.2	54.6
E	22.6	62.6
F	21.8	62.2
G	23.0	53.3
H	22.0	54.5

Comparing G and H, which respectively correspond to tests without and with stemming, one can also conclude that stemming is not of major importance for recall values, but it can play an important role in precision matters. While stopword removal significantly alters the content of input data, stemming only alters its shape, i.e. the reduction of information is not significant. We can therefore say that stemming relate to the efficiency of the learning machine (there is less redundancy in the data).

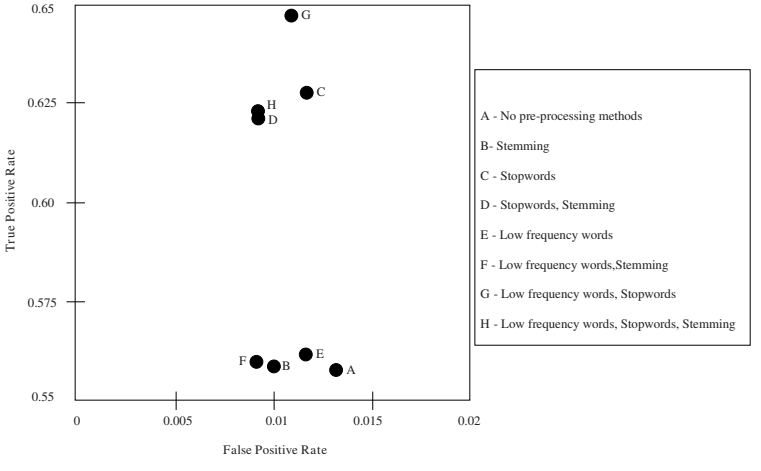

Fig. 1.6 Evaluation of pre-processing methods with ROC graph.

Figure 1.6 presents the ROC graph with one point for each of the combinations of pre-processing methods, maintaining the decision threshold or cut-off point halfway between positive and negative examples. For an output decision range in $[-1,1]$ the threshold was set to zero. Notice that the axes are not equally ranged, permitting an easy comparison of approaches.

Analyzing this graph, it can be verified that testing conditions C,D,G, and H, where stopword removal was performed, are definitely better than the rest. These results confirmed that stopword removal removes information that could mislead the learning machine, compromising the performance.

1.8 Conclusion

This chapter has introduced the background to text classification tasks, viz. representation, pre-processing, learning and evaluation issues.

Text classification is a high-dimensional learning problem that can potentially enhance performance if the correct dimensionality reduction techniques are applied. Even though these pre-processing steps are not the core of this book, they constitute a vital component of a text classification system. As such, we have undertaken an empirical study to promote the comparison, determine the influence, and relative importance of dimensionality reduction methods on text classification performance.

In the study, three standard dimensionality reduction techniques were tested: low frequency word removal, stopword removal and stemming. It was possible to draw several conclusions. Stopword removal was found to be the most significant technique applied, yielding the best performing classifiers in all tested conditions. Stemming also plays an important role, especially for precision of the classifiers. While stopword removal alters significantly the contents of input data, stemming only alters its shape, i.e. the reduction of information is not significant. We can therefore say that stemming is more relevant in terms of efficiency of the learning machine (there is less redundancy in data). Low frequency word removal has little influence in classification performance matters, but it can reduce training complexity by decreasing the number of features. A general conclusion is that the evaluated dimensionality reduction techniques can strengthen the classifier performance.

Support vector machines (SVMs) were used in the presented evaluation. Other learning techniques were mentioned and, while most of them still retain their popularity, it is fair to say that in recent years SVM [54] and boosting [113] have been the two dominant learning methods in text classification [123]. The next chapter refers to some of these issues and presents a more detailed discussion on the subject of kernel-based learning machines for text classification.

Chapter 2
Kernel Machines for Text Classification

Abstract. This chapter details the concept of kernel methods and presents the foundations of two paradigmatic techniques: support vector machines and relevance vector machines. Both are introduced in a text classification perspective and then results and comparisons of their application to benchmark corpora are presented.

2.1 Kernel Methods

The concept of kernels was brought to the machine learning scenario by Boser et al. [16], introducing the technique of support vector machines (SVMs). Since then there has been considerable interest in this topic [115, 118, 124]. Kernel methods are properly motivated theoretically and exhibit good generalization performance on many real-life data sets, including in their application to text classification [54, 148]. The most popular kernel methods are support vector machines (SVMs) [151, 117, 152] and the recently introduced relevance vector machines (RVMs) [147, 14]. Kernel approaches are mostly binary while, as noted in Chapter 1, text classification applications are usually multi-label, multiclass problems. However, a text classification problem with a set of $|\mathscr{C}|$ categories, $\mathscr{C} = \{c_1, c_2, \ldots, c_j, \ldots, c_{|\mathscr{C}|}\}$, like the benchmarks in Section 1.6.2, is usually tackled as $|\mathscr{C}|$ independent binary classification problems under $\{c_j, \overline{c}_j\}$, for $j = 1, \ldots, |\mathscr{C}|$. In this case, a classifier for $\mathscr{C}$ is thus actually composed of $|\mathscr{C}|$ one-against-all binary classifiers.

In the following sections two of the most accepted kernel methods will be detailed, namely SVMs and RVMs, treating both as solutions to a binary problem of classifying a set of documents $\mathscr{D} = \{\mathbf{d}_1, \mathbf{d}_2, \ldots, \mathbf{d}_i, \ldots, \mathbf{d}_{|\mathscr{D}|}\}$ into two possible target classes: $\{c_j, \overline{c}_j\}$.

Kernel methods can be cast into a class of pattern recognition techniques in which the training data points, or at least a subset of them, are kept and used also during the prediction phase. Moreover, many linear parametric models can be transformed into an equivalent dual representation in which the predictions are based on a linear combination of a kernel function evaluated at the training data points [13]. For models which are based on fixed nonlinear feature space mapping, $\phi(\mathbf{d})$, the kernel function is given by the relation

C. Silva and B. Ribeiro: Inductive Inference for Large Scale Text Classification, SCI 255, pp. 31–48.
springerlink.com

$$k(\mathbf{d}_1,\mathbf{d}_2) = \phi(\mathbf{d}_1)^T \phi(\mathbf{d}_2), \tag{2.1}$$

making the kernel a symmetric function of its arguments so that $k(\mathbf{d}_1,\mathbf{d}_2) = k(\mathbf{d}_2,\mathbf{d}_1)$ [13]. The simplest example of a kernel function is the linear kernel, obtained by considering the identity mapping for the feature space in (2.1), so that $\phi(\mathbf{d}) = \mathbf{d}$, in which case $k(\mathbf{d}_1,\mathbf{d}_2) = \mathbf{d}_1^T \mathbf{d}_2$.

The concept of a kernel formulated as an inner product in a feature space allows to build interesting extensions of many well known algorithms by making use of the kernel trick, also known as kernel substitution. The general idea is that, if we have an algorithm formulated in such a way that the input document $\mathbf{d}$ enters only in the form of scalar products, then we can replace the scalar product with some other choice of kernel. For instance, the technique of kernel substitution can be applied to principal component analysis (PCA) in order to develop a nonlinear version of PCA, KPCA (kernel-PCA) [119].

2.2 Support Vector Machines

Support vector machines (SVMs) were introduced by Vapnik [151] based on the structural risk minimization (SRM) principle, as an alternative to the traditional empirical risk minimization (ERM) principle. The main idea is to find a hypothesis h from a hypothesis space H for which one can guarantee the lowest probability of error $Err(h)$. Given a training set of n examples:

$$(\mathbf{d}_1,t_1),\ldots,(\mathbf{d}_n,t_n), \mathbf{d}_i \in \mathbb{R}^N, t_i \in \{-1,+1\}, \tag{2.2}$$

the SRM connects the true error of a hypothesis h with the error on the training set, Err_{train}, and with the complexity of the hypothesis (2.3):

$$Err(h) \leq Err_{train}(h) + O\left(\frac{v\ln(\frac{n}{v}) - \ln(\eta)}{n}\right). \tag{2.3}$$

This upper bound holds with a probability of at least $1-\eta$. v denotes the VC-dimension[1] [152], which is a property of the hypothesis space H and indicates its expressiveness. This bound reflects the tradeoff between the complexity of the hypothesis space and the training error [54], i.e a simple hypothesis space will not approximate the desired functions, resulting in high training and true errors. On the other hand, a too-rich hypothesis space, i.e. with large VC-dimension, will result in overfitting. This problem arises because, although a small training error will occur, the second term in the right-hand side of (2.3) will be large. Hence, it is crucial to determine the hypothesis space with the sufficient complexity. In the SRM this is achieved by defining a nested structure of hypothesis spaces H_i, so that their respective VC-dimension v_i increases:

$$H_1 \subset H_2 \subset H_3 \subset \cdots \subset H_i \subset \ldots \quad \text{and} \quad \forall i : v_i \leq v_{i+1} \tag{2.4}$$

[1] Vapnik-Chervonenkis dimension.

This structure is *apriori* defined to find the index for which (2.3) is minimum. To build this structure the number of features is restricted. SVMs learn linear threshold functions of the type:

$$h(\mathbf{d}) = sign(\boldsymbol{\omega}.\mathbf{d} + b) = \begin{cases} +1 & \text{if } \boldsymbol{\omega}.\mathbf{d} + b > 0 \\ -1 & \text{otherwise,} \end{cases} \tag{2.5}$$

where $\boldsymbol{\omega} = (\omega_1, \omega_2, \ldots, \omega_N)^T$ are linear parameters (weights) of the model and b is the bias. Linear threshold functions with N features have a VC-dimension of N+1 [152].

2.2.1 Linear Hard-Margin SVMs

In the simplest SVM formulation, a training set can be separated by at least one hyperplane, i.e. data are linearly separable and a linear model can be used:

$$y(\mathbf{d}) = \boldsymbol{\omega}^T \mathbf{d} + b. \tag{2.6}$$

In this case, SVMs are called linear hard-margin and there is a weight vector $\boldsymbol{\omega}'$ and a threshold b' that correctly define the model. Basically there is a function of the form (2.6) that satisfies $y(\mathbf{d}_i) > 0$ for documents with $t_i = 1$ and $y(\mathbf{d}_i) < 0$ for points having $t_i = -1$, so that for each training document $(\mathbf{d}_i, t_i)$

$$t_i y(\mathbf{d}_i) = t_i(\boldsymbol{\omega}'.\mathbf{d}_i + b') > 0. \tag{2.7}$$

As a rule there can be multiple hyperplanes that allow such separation without error (see Figure 2.1), and the SVM determines the one with largest margin ρ, i.e. furthest from the hyperplane to the closest training examples. The examples closer to the hyperplane are called support vectors (SVs) and have an exact distance of ρ to the hyperplane. Figure 2.2 depicts an optimal separating hyperplane and four SVs.

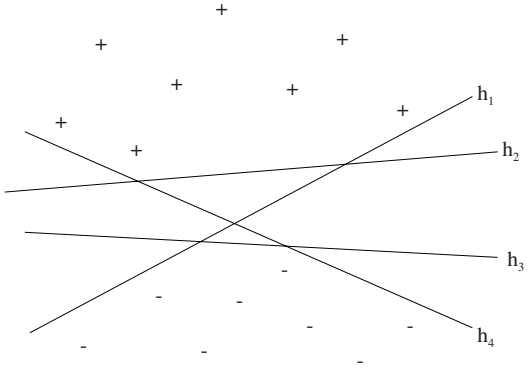

Fig. 2.1 Possible hyperplanes separating positive and negative training examples.

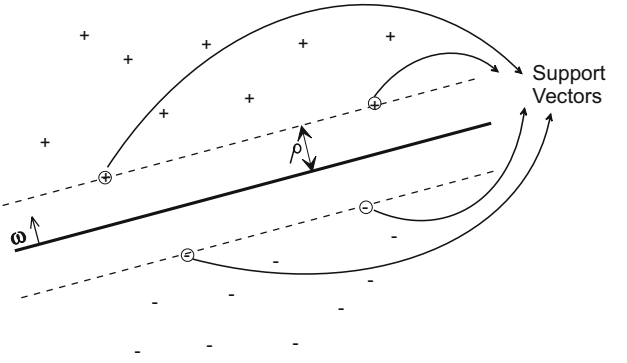

Fig. 2.2 SVM optimal separating hyperplane and separating margin, ρ.

The perpendicular distance of a point from a hyperplane defined by $y(\mathbf{d}) = 0$, where $y(\mathbf{d})$ takes the form of (2.6), is given by $\frac{|y(\mathbf{d})|}{||\boldsymbol{\omega}||}$ [13]. As for now the interest is only in solutions for which all documents are correctly classified, so that $t_i y(\mathbf{d}_i) > 0$, the distance of a document $\mathbf{d}_i$ to the decision surface is given by

$$\frac{t_i y(\mathbf{d}_i)}{||\boldsymbol{\omega}||} = \frac{t_i(\boldsymbol{\omega}^T.\mathbf{d}_i + b)}{||\boldsymbol{\omega}||}. \tag{2.8}$$

The margin is then given by the perpendicular distance to the closest point $\mathbf{d}_i$ of the data set, and we wish to optimize the parameters $\boldsymbol{\omega}$ and b in order to maximize this distance. Thus, the maximum margin solution is found by solving

$$\arg\max_{\boldsymbol{\omega},b}\left\{\frac{1}{||\boldsymbol{\omega}||}\min_i[t_i(\boldsymbol{\omega}^T\mathbf{d}_i + b)]\right\}. \tag{2.9}$$

Using a canonical representation of the decision hyperplane, all data points satisfy

$$t_i(\boldsymbol{\omega}^T\mathbf{d}_i + b) \geq 1. \qquad i = 1,\ldots,n \tag{2.10}$$

When the above equality holds, the constraints are said to be active for that data point, while for the rest the constraints are inactive. There will be always at least one active constraint, since there will always be a closest point, and once the margin has been maximized, there will be at least two active constraints. Then, the optimization problem consists of maximizing $||\boldsymbol{\omega}||^{-1}$, which is equivalent to minimizing $||\boldsymbol{\omega}||^2$, and so we have to solve the (primal) optimization problem:

$$\begin{aligned} &\text{minimize: } \frac{1}{2}\boldsymbol{\omega}.\boldsymbol{\omega}, \\ &\text{subject to: } \forall_{i=1}^n : t_i\,[\boldsymbol{\omega}.\mathbf{d}_i + b] \geq 1 \end{aligned} \tag{2.11}$$

These constraints establish that all training examples should lie on the correct side of the hyperplane. Using the value of 1 on the right-hand side of the inequalities enforces a certain distance from the hyperplane:

$$\rho = \frac{1}{||\boldsymbol{\omega}||}, \tag{2.12}$$

where $||\boldsymbol{\omega}||$ denotes the L_2-norm of $\boldsymbol{\omega}$. Therefore minimizing $\boldsymbol{\omega}.\boldsymbol{\omega}$ is equivalent to maximizing the margin. The weight vector $\boldsymbol{\omega}$ and the threshold b describe the optimal (maximum margin) hyperplane.

Vapnik [151] showed there is a relation between the margin and the VC-dimension. Considering the hyperplanes $h(\mathbf{d}) = sign(\boldsymbol{\omega}.\mathbf{d}+b)$ in an N-dimensional space as a hypothesis, if all examples $\mathbf{d}_i$ are contained in a hyper-circle of diameter R, it is required that, for examples $\mathbf{d}_i$:

$$abs(\boldsymbol{\omega}.\mathbf{d}_i + b) \geq 1 \tag{2.13}$$

and then this set of hyperplanes has a VC-dimension, v, bounded by:

$$v \leq min\left(\left[\frac{R^2}{\rho^2}\right], N\right) + 1, \tag{2.14}$$

where $[c]$ is the integer part of c. Thus, the VC-dimension is lower for larger margins. Moreover, the VC-dimension of the maximum margin hyperplane does not necessarily depend on the number of features, but on the Euclidean length $||\boldsymbol{\omega}||$ of the weight vector optimized by the SVM. Intuitively this means that the true error of a separating maximum margin hyperplane is close to the training error even in high-dimensional spaces, as long as it has a small weight vector.

The primal constrained optimization problem (2.11) is numerically difficult to handle. Therefore, one introduces Lagrange multipliers, $\alpha_i \geq 0$, with one multiplier for each of the constraints in (2.11), obtaining the Lagrangian function

$$L(\boldsymbol{\omega}, b, \boldsymbol{\alpha}) = \frac{1}{2}||\boldsymbol{\omega}||^2 - \sum_{i=1}^{n} \alpha_i[t_i(\boldsymbol{\omega}^T \mathbf{d}_i + b) - 1], \tag{2.15}$$

where $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_n)^T$. Note the minus sign in front of the Lagrangian multiplier term, because we are minimizing with respect to $\boldsymbol{\omega}$ and b, and maximizing with respect to $\boldsymbol{\alpha}$. Setting the derivatives of $L(\boldsymbol{\omega}, b, \boldsymbol{\alpha})$ with respect to $\boldsymbol{\omega}$ and b to zero, we obtain the following two conditions:

$$\boldsymbol{\omega} = \sum_{i=1}^{n} \alpha_i t_i \mathbf{d}_i, \tag{2.16}$$

$$0 = \sum_{i=1}^{n} \alpha_i t_i. \tag{2.17}$$

Eliminating $\boldsymbol{\omega}$ and b from $L(\boldsymbol{\omega}, b, \boldsymbol{\alpha})$ gives the dual representation of the maximum margin problem

$$\begin{aligned} \text{maximize: } & \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} t_i t_j \alpha_i \alpha_j (\mathbf{d}_i . \mathbf{d}_j) \\ \text{subject to: } & \sum_{i=1}^{n} t_i \alpha_i = 0 \\ & \forall i \in [1..n] : 0 \leq \alpha_i \end{aligned} \tag{2.18}$$

The matrix $\mathbf{Q}$ with $Q_{ij} = t_i t_j (\mathbf{d}_i . \mathbf{d}_j)$ is commonly referred to as the Hessian matrix. The result of the optimization process is a vector of Lagrangian coefficients $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_n)^T$ for which the dual problem (2.18) is optimized. These coefficients can then be used to construct the hyperplane, solving the primal optimization problem (2.11) just by linearly combining training examples (2.19).

$$\boldsymbol{\omega} . \mathbf{d} = \left(\sum_{i=1}^{n} \alpha_i t_i \mathbf{d}_i \right) . \mathbf{d} = \sum_{i=1}^{n} \alpha_i t_i (\mathbf{d}_i \mathbf{d}) \qquad \text{and} \qquad b = t_{SV} - \boldsymbol{\omega} . \mathbf{d}_{SV} \tag{2.19}$$

Only support vectors (SVs) have a non-zero α_i coefficient. To determine b from (2.18) and (2.19), an arbitrary support $\mathbf{d}_{SV}$ vector with its class label t_{SV} can be used.

2.2.2 Soft-Margin SVMs

Hard-margin SVMs fail when the training documents are not linearly separable, since there is no solution to the optimization problems (2.11) and (2.18). Even though most text-classification problems are linearly separable, it might still be preferable to allow some errors in the training data, as indicated by the structural risk minimization [152, 54]. This is the rationale of the soft-margin SVMs. They minimize the weight vector, like the hard-margin SVMs, but they simultaneously minimize the number of training errors by the introduction of slack variables, ξ_i. The primal optimization problem of (2.11) is now reformulated as

$$\begin{aligned} \text{minimize: } & \frac{1}{2} \boldsymbol{\omega} . \boldsymbol{\omega} + C \sum_{i=1}^{n} \xi_i, \\ \text{subject to: } & \forall_{i=1}^{n} : t_i [\boldsymbol{\omega} . \mathbf{d}_i + b] \geq 1 - \xi_i \\ & \forall_{i=1}^{n} : \xi_i > 0. \end{aligned} \tag{2.20}$$

If a training example is wrongly classified by the SVM model, the corresponding ξ_i will be greater than 1. Therefore $\sum_{i=1}^{n} \xi_i$ constitutes an upper bound on the number of training errors. The factor C in (2.20) is a parameter that allows the tradeoff between training error and model complexity. A small value of C will increase the training errors, while a large C will lead to behavior similar to that of a hard-margin SVM.

Again, this primal problem is transformed in its dual counterpart, for computational reasons. The dual problem is similar to the hard-limit SVM (2.18), except for the C upper bound on the Lagrange multipliers α_i:

$$\begin{aligned} \text{maximize: } & \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} t_i t_j \alpha_i \alpha_j (\mathbf{d}_i . \mathbf{d}_j) \\ \text{subject to: } & \sum_{i=1}^{n} t_i \alpha_i = 0 \\ & \forall i \in [1..n] : 0 \leq \alpha_i \leq C \end{aligned} \tag{2.21}$$

As before, all training examples with $\alpha_i > 0$ are called support vectors. To differentiate between those with $0 \leq \alpha_i < C$ and those with $\alpha_i = C$, the former are called unbounded SVs and the latter are called bounded SVs.

From the solution of (2.21) the classification rule can be computed exactly, as in the hard-margin SVM:

$$\boldsymbol{\omega}.\mathbf{d} = \left(\sum_{i=1}^{n} \alpha_i t_i \mathbf{d}_i \right) .\mathbf{d} = \sum_{i=1}^{n} \alpha_i t_i (\mathbf{d}_i \mathbf{d}) \qquad \text{and} \qquad b = t_{SV} - \boldsymbol{\omega}.\mathbf{d}_{SV} \tag{2.22}$$

The only additional restriction is that the SV $(\mathbf{d}_{SV}, t_{SV})$ for calculating b has to be an unbounded SV.

2.2.3 Nonlinear SVMs

So far in this section, we have only discussed SVMs for linear classification rules. Linear classifiers are inappropriate for many real-world problems, since the problems may have an inherently nonlinear structure. A remarkable property of SVMs is that they can easily be transformed into nonlinear learners [16]. The attribute document vectors $\mathbf{d}$ are mapped into a high-dimensional feature space X'. Despite the fact that the classification rule is linear in X', it is nonlinear when projected into the original input space. The following example with a document $\mathbf{d} = (w_1, w_2)$, with only two input attributes w_1 and w_2 (words or terms) illustrates this. Let us choose

$$\Phi(\mathbf{d}) = \Phi((w_1, w_2)^T) = (w_1^2, w_2^2, \sqrt{2} w_1 w_2, \sqrt{2} w_1, \sqrt{2} w_2, 1)^T \tag{2.23}$$

as the nonlinear mapping. Although it is not possible to linearly separate the examples in 2.3 a), they become linearly separable in 2.3 b), after the mapping with $\Phi(\mathbf{d})$ into the higher dimensional feature space [54].

In general, such a mapping $\Phi(\mathbf{d})$ is inefficient to compute. However, there is a special property of SVMs that handles this problem. During both training and testing, it is sufficient to be able to compute dot-products in feature space, i.e. $\Phi(\mathbf{d}_i).\Phi(\mathbf{d}_j)$. For special mappings $\Phi(\mathbf{d})$ such dot-products can be computed very efficiently using kernel functions $K(\mathbf{d}_1, \mathbf{d}_2)$. If a function $K(\mathbf{d}_1, \mathbf{d}_2)$ satisfies Mercer's Theorem, i.e. it is a continuous symmetric kernel of a positive integer operator,

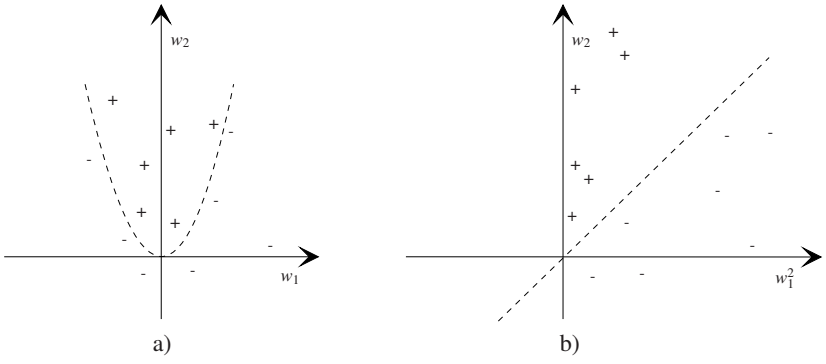

Fig. 2.3 a) Nonlinearly separable training set in (w_1, w_2); b) Projection of the same set onto (w_1^2, w_2), where the set becomes linearly separable.

it is guaranteed to compute the inner product of the vectors $\mathbf{d}_1$ and $\mathbf{d}_2$ after they have been mapped into a new space by some nonlinear mapping Φ [151]:

$$\Phi(\mathbf{d}_1).\Phi(\mathbf{d}_2) = K(\mathbf{d}_1, \mathbf{d}_2) \tag{2.24}$$

Depending on the choice of kernel function, SVMs can learn polynomial classifiers (2.25), radial basis function (RBF) classifiers (2.26) or two-layer sigmoid neural networks (2.27).

$$K_{poly}(\mathbf{d}_1, \mathbf{d}_2) = (\mathbf{d}_1.\mathbf{d}_2 + 1)^d \tag{2.25}$$

$$K_{RBF}(\mathbf{d}_1, \mathbf{d}_2) = \exp\left(-\gamma(\mathbf{d}_1 - \mathbf{d}_2)^2\right) \tag{2.26}$$

$$K_{sigmoid}(\mathbf{d}_1, \mathbf{d}_2) = \tanh\left(s(\mathbf{d}_1.\mathbf{d}_2) + c\right) \tag{2.27}$$

The kernel $K_{poly}(\mathbf{d}_1, \mathbf{d}_2) = (\mathbf{d}_1.\mathbf{d}_2 + 1)^2$ corresponds to the mapping in Equation (2.23). To use a kernel function, one simply substitutes every occurrence of the inner product in equations (2.21) and (2.22) with the desired kernel function.

2.3 Relevance Vector Machines

Relevance vector machines (RVMs) are a Bayesian treatment of the SVM prediction model. However, RVMs are not a Bayesian treatment of the SVMs methodology, but use kernels as simply defining a set of basis functions, rather than as a definition of a dot-product in some space [13]. SVMs are state-of-the-art learning algorithms for classification, and have several desirable properties, such as fitting functions in high-dimensional feature spaces, sparsity and good generalization performance by margin maximization [115]. RVMs seem to have overcome some of the SVMs' disadvantages, notably [147]:

- Although relatively sparse, SVM make unnecessary use of basis functions since the number of SVs required typically grows linearly with the size of the training set. Some form of post-processing is often required to reduce computational complexity;
- Predictions are not probabilistic. In regression the SVM outputs a point estimate and in classification a 'hard' binary decision. Ideally we want to estimate the conditional distribution in order to capture the uncertainty in the prediction.
- It is necessary to estimate the error/margin tradeoff parameter C (and in regression the insensitivity parameter ε too). This generally entails a cross-validation procedure, which is wasteful of both data and computation.
- The kernel function must satisfy the Mercer condition.

Relevance vector machines (RVMs) constitute a viable alternative to SVMs. Their advantages rise due to the ability to yield a decision function that is much sparser than SVM, slightly penalizing its classification performance. This can lead to a significant reduction in the computational complexity of the decision function, thereby making it more suitable for real-time applications [157].

RVMs were proposed by Tipping [147] as a Bayesian probabilistic framework for learning models of the type

$$y(\mathbf{d}) = \sum_{i=1}^{n} \omega_i \phi_i(\mathbf{d}) = \boldsymbol{\omega}^T \boldsymbol{\Phi}(\mathbf{d}), \tag{2.28}$$

where the output is a linearly-weighted sum of n, generally nonlinear and fixed, basis functions $\boldsymbol{\Phi}(\mathbf{d}) = (\phi_1(\mathbf{d}), \phi_2(\mathbf{d}), \ldots, \phi_n(\mathbf{d}))^T$. Functions of the type (2.28) are analyzed by adjusting the weights $\boldsymbol{\omega} = (\omega_1, \omega_2, \ldots, \omega_n)^T$ that are linear parameters of the model, for which the learning process estimates adequate values.

We shall now introduce the Bayesian approaches, and then the RVM approach.

2.3.1 Bayesian Approaches

The basis of all Bayesian approaches is Bayes' theorem:

$$p(\mathbf{t}|\mathbf{d}) = \frac{p(\mathbf{d}|\mathbf{t})p(\mathbf{t})}{p(\mathbf{d})}, \tag{2.29}$$

where $\mathbf{t}$ is the target and $\mathbf{d}$ is the input document. This theorem states that some prior probability $p(\mathbf{t})$ can be transformed into a posterior probability $p(\mathbf{t}|\mathbf{d})$, by incorporating evidence provided by the observed data $p(\mathbf{d}|\mathbf{t})$.

When adapted to learning algorithms, Bayes' theorem can be used to determine model parameters:

$$p(\boldsymbol{\omega}|\mathbf{t}) = \frac{p(\mathbf{t}|\boldsymbol{\omega})p(\boldsymbol{\omega})}{p(\mathbf{t})}, \tag{2.30}$$

where $\boldsymbol{\omega} = (\omega_1, \omega_2, \ldots, \omega_n)^T$ are the model parameters and $\mathbf{t}$ is again the target. In this form, it allows us to evaluate the uncertainty in $\boldsymbol{\omega}$ after we have observed $\mathbf{t}$,

in the form of the posterior probability $p(\boldsymbol{\omega}|\mathbf{t})$. The quantity $p(\mathbf{t}|\boldsymbol{\omega})p(\mathbf{t}|\boldsymbol{\omega})$ on the right hand side of (2.30) is evaluated for the observed dataset (with targets $\mathbf{t}$) and can be viewed as a function of the parameter vector $\boldsymbol{\omega}$, in which case it is called the likelihood function. It expresses how probable the observed data is for different settings of the parameter vector $\boldsymbol{\omega}$. Given this definition of likelihood, we can state Bayes' theorem as

$$\text{posterior} \propto \text{likelihood} \times \text{prior}, \tag{2.31}$$

where all of these quantities are viewed as functions of $\boldsymbol{\omega}$ [13]. The denominator of (2.30) is the normalization constant, which ensures that the posterior distribution on the left-hand side is a valid probability density and integrates to one.

2.3.2 RVM Approach

Relevance vector machines' (RVMs') key feature is that as well as offering good generalization performance, the inferred predictors are remarkably sparse in that they contain few non-zero ω_i parameters. Most parameters are automatically set to zero during the learning process, giving a procedure that is extremely effective at discerning those basis functions which are *relevant* for making good predictions [147].

Specifically, a fully probabilistic framework is adopted by introducing a prior over the model weights. Each weight ω_i is determined by a hyperparameter α_i, whose most probable values are iteratively estimated from the data. Sparsity is achieved because, in practice, the posterior distributions of many of the weights are sharply (in fact infinitely) peaked around zero. Those training vectors associated with the remaining non-zero weights are the relevance vectors (RVs), in deference to the principle of automatic relevance determination (ARD) which motivates the approach [82].

RVMs were initially introduced for regression problems, but the formulation for classification follows an essentially identical framework. Consider a training set with n documents $\mathbf{d}_i$ and a binary target variable t_i

$$(\mathbf{d}_1, t_1), \ldots, (\mathbf{d}_n, t_n), \mathbf{d}_i \in \mathbf{R}^N, t_i \in \{0, 1\}. \tag{2.32}$$

For two-class classification, it is desired to predict the posterior probability of membership of one of the classes given the input document $\mathbf{d}$. Following the statistical convention and generalizing the linear model (2.6) by applying the logistic sigmoid link function $\sigma(y) = 1/(1+e_{-y})$ to $y(\mathbf{d})$:

$$y(\mathbf{d}, \boldsymbol{\omega}) = \sigma\left(\sum_{i=1}^{n} \omega_i \phi_i(\mathbf{d}) + \omega_0\right) \quad , \tag{2.33}$$

where $\boldsymbol{\omega} = (\omega_0, \ldots, \omega_n)^T$ are the weights (the bias was integrated into the weight vector as ω_0), $\phi_i(\mathbf{d}) = k(\mathbf{d}, \mathbf{d}_i)$ with $k(\cdot, \cdot)$ representing the kernel function and $\mathbf{d}_i$ an example (or a *relevance vector*) from the training set.

Furthermore, adopting the Bernoulli distribution for $P(t|\mathbf{d})$, the likelihood can be written as:

$$p(t|\mathbf{d}) = \prod_{i=1}^{n} \sigma\{(y(\mathbf{d}_i;\boldsymbol{\omega})\}^{t_i}[1-\sigma\{y(\mathbf{d}_i;\boldsymbol{\omega})\}]^{1-t_i}, \tag{2.34}$$

where, following from the probabilistic specification, the targets $t_i \in \{0,1\}$[2].

Under the RVM framework, priors are introduced over the weight vector $\boldsymbol{\omega}$, obtaining a model by applying an ARD (Automatic Relevance Determination) prior $p(\boldsymbol{\omega}|\boldsymbol{\alpha}) = \prod_{i=1}^{M} \mathcal{N}(\omega_i|0,\alpha_i^{-1})$, where α_i is the precision hyperparameter of the i^{th} parameter ω_i. Note that now we have $M = n+1$ hyperparameters, one for each input and another for the bias. It should also be emphasized that this analysis is valid for arbitrary choices of basis functions, i.e. there is no restriction (as there was for SVMs) of using positive definite kernels or to use the same kernel for all instances.

In the regression case, the weights are analytically integrated out, and both the weight posterior $p(\boldsymbol{\omega}|\mathbf{t},\boldsymbol{\alpha})$ and the marginal likelihood $p(\mathbf{t}|\boldsymbol{\alpha})$ have closed-form expressions. However, for classification this is not possible, and so the following approximation procedure is adopted, as used by [81], based on Laplace's method:

1. For the current, fixed, values of $\boldsymbol{\alpha}$, the most probable weights $\boldsymbol{\omega}_{MP}$ are found, giving the location of the mode of the posterior distribution. Since $p(\boldsymbol{\omega}|\mathbf{t},\boldsymbol{\alpha}) \propto p(t|\boldsymbol{\omega})p(\boldsymbol{\omega}|\boldsymbol{\alpha})$, this is equivalent to finding the maximum, over $\boldsymbol{\omega}$, of

$$\ln\{p(\mathbf{t}|\boldsymbol{\omega})p(\boldsymbol{\omega}|\boldsymbol{\alpha})\} = \sum_{i=1}^{n}[t_i \ln y_i + (1-t_i)\ln(1-y_n)] - \frac{1}{2}\boldsymbol{\omega}^T\mathbf{A}\boldsymbol{\omega}, \tag{2.35}$$

   with $y_i = \sigma\{y(\mathbf{d}_i;\boldsymbol{\omega})\}$ and $\mathbf{A} = \mathrm{diag}(\alpha_i)$. This can be done using iterative-reweighted least squares (IRLS) [13].

2. For IRLS, one needs the gradient vector and Hessian matrix of the log posterior distribution [14], obtained by deriving (2.35),

$$\nabla \ln p(\boldsymbol{\omega}|\mathbf{t},\boldsymbol{\alpha}) = \boldsymbol{\Phi}^T(\mathbf{t}-\mathbf{y}) - \mathbf{A}\boldsymbol{\omega}, \tag{2.36}$$

$$\nabla^2 \ln p(\boldsymbol{\omega}|\mathbf{t},\boldsymbol{\alpha}) = -(\boldsymbol{\Phi}^T\mathbf{B}\boldsymbol{\Phi} + \mathbf{A}), \tag{2.37}$$

   where $\mathbf{B}$ is an $n \times n$ diagonal matrix with elements $b_i = y_i(1-y_i)$, $\mathbf{y} = (y_1,\ldots,y_n)^T$ and $\boldsymbol{\Phi}$ is the design matrix with elements $\Phi_{ij} = \phi_j(\mathbf{d}_i)$.
   This is then negated and inverted to give the covariance $\boldsymbol{\Sigma}$ for a Gaussian approximation to the posterior over weights centered at $\boldsymbol{\omega}_{MP}$.
3. The mode of the resulting approximation to the posterior distribution corresponding to the mean of the Gaussian approximation, is obtained setting (2.36) to zero, giving the mean and covariance of the Laplace approximation in the form

$$\boldsymbol{\omega}_{MP} = \mathbf{A}^{-1}\boldsymbol{\Phi}^T(\mathbf{t}-\mathbf{y}) \tag{2.38}$$

[2] Note that this is the reason for the difference between the values of the targets in SVMs (2.2) and RVMs (2.32).

$$\boldsymbol{\Sigma} = (\boldsymbol{\Phi}^T \mathbf{B} \boldsymbol{\Phi} + \mathbf{A})^{-1}. \tag{2.39}$$

We now can use this Laplace approximation to evaluate the marginal likelihood. Evaluating an integral using the Laplace approximation, we have [13]

$$\begin{aligned} p(\mathbf{t}|\boldsymbol{\alpha}) &= \int p(\mathbf{t}|\boldsymbol{\omega}) p(\boldsymbol{\omega}|\boldsymbol{\alpha}) d\boldsymbol{\omega} \\ &\simeq p(\mathbf{t}|\boldsymbol{\omega}_{MP}) p(\boldsymbol{\omega}_{MP}|\boldsymbol{\alpha}) (2\pi)^{M/2} |\boldsymbol{\Sigma}|^{1/2}. \end{aligned} \tag{2.40}$$

If we substitute $p(\mathbf{t}|\boldsymbol{\omega}_{MP})$ and $p(\boldsymbol{\omega}_{MP}|\boldsymbol{\alpha})$ and then set the derivative of the marginal likelihood with respect to α_i equal to zero, we obtain [13]

$$-\frac{1}{2}(w_{iMP}) + \frac{1}{2\alpha_i} - \frac{1}{2}\Sigma_{ii} = 0. \tag{2.41}$$

Defining $\gamma_i = 1 - \alpha_i \Sigma_{ii}$ and rearranging then gives

$$\alpha_i^{new} = \frac{\gamma_i}{(w_{iMP})^2} \tag{2.42}$$

which is identical to the regression updating formula.

In a batch-learning approach, all training examples will be considered as relevance vectors at the initial stage and the remaining vectors will be *pruned* after re-evaluation of $\boldsymbol{\alpha}$ in each iteration. In other words, every α_i has a finite value at the beginning of the iteration, and the Hessian matrix to be computed in each estimation loop has a size of $(n+1) \times (n+1)$. Since the inversion of Hessian matrices is involved in the learning algorithm, the overall training complexity is $O(n^3)$. This implies that

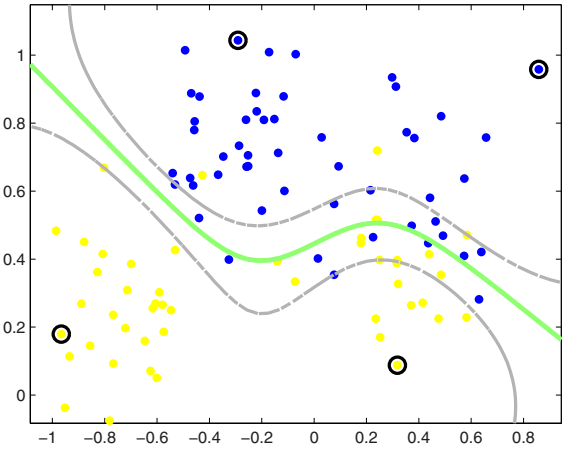

Fig. 2.4 RVM two-class classification example with four relevance vectors.

if the initial sample size is huge, the learning algorithm may take a long time to converge.

The idea of what a relevance vector (RV) is can be gleaned from Fig. 2.4, which shows a two-dimensional RVM classification example with four RVs. Informally, RVMs try to describe the decision surface *as simply as possible* by selecting the RVs as *typical* instances. On the other hand, Support Vector Machines describe the decision surface by selecting as *support vectors* (SVs) the borderline and miss-classified instances [55], usually resulting in a larger set of vectors to define a model.

2.4 Baseline Kernel Machines Performances with Benchmark Corpora

In this section the baseline classification results for support vector machines and relevance vector machines are presented. Performances are evaluated in both the

Table 2.1 Performances for SVM with Reuters-21578.

Default parameters

Category	SVs	Accuracy	Precision	Recall	F1
Earn	1,632	95.92%	93.53%	95.50%	94.50%
Acquisitions	1,751	94.93%	93.09%	85.15%	88.94%
Money-fx	908	96.13%	71.43%	52.80%	60.72%
Grain	771	97.96%	92.55%	63.04%	75.00%
Crude	693	97.04%	84.85%	63.64%	72.73%
Trade	647	97.64%	79.49%	54.87%	64.92%
Interest	742	97.15%	77.03%	47.11%	58.46%
Ship	500	98.45%	89.36%	51.85%	65.62%
Wheat	487	98.77%	84.44%	57.58%	68.47%
Corn	484	99.08%	93.33%	53.85%	68.29%
Average	861.50	97.31%	85.91%	62.24%	71.77%

Doubled importance of errors on the positive class

Category	SVs	Accuracy	Precision	Recall	F1
Earn	1,000	98.58%	97.81%	98.47%	98.14%
Acquisitions	1,249	97.96%	96.81%	94.44%	95.59%
Money-fx	645	97.45%	76.30%	73.05%	74.64%
Grain	645	98.54%	97.96%	71.64%	82.76%
Crude	517	98.32%	90.20%	80.12%	84.87%
Trade	501	97.74%	76.60%	64.29%	69.90%
Interest	545	98.32%	82.92%	68.00%	74.73%
Ship	500	98.54%	94.12%	56.47%	70.59 %
Wheat	420	98.94%	84.91%	68.18%	75.63%
Corn	394	99.12%	87.50%	58.33%	70.00%
Average	641.60	98.35%	88.51%	73.29%	79.68%

Table 2.2 Performances for RVM with Reuters-21578.

1,000 training documents

Category	RVs	Accuracy	Precision	Recall	F1
Earn	24	96.64%	93.43%	98.08%	95.70%
Acquisitions	26	94.02%	90.66%	83.05%	86.69%
Money-fx	19	95.95%	74.19%	32.62%	45.32 %
Grain	22	97.23%	69.86%	76.12%	72.86%
Crude	16	96.24%	73.02%	57.14%	64.11%
Trade	19	96.24%	56.72%	33.93%	42.46%
Interest	16	96.75%	58.73%	37.00%	45.40 %
Ship	15	97.48%	80.77%	24.71%	37.84%
Wheat	19	98.54%	68.06 %	74.24%	71.01%
Corn	15	98.65 %	60.78%	64.58%	62.63%
Average	19.10	96.78%	72.62%	58.15%	62.40%

2,000 training documents

Category	RVs	Accuracy	Precision	Recall	F1
Earn	35	98.10%	97.24%	97.80%	97.52%
Acquisitions	49	96.13%	93.66%	89.58%	91.57%
Money-fx	35	96.28%	70.10%	48.23%	57.14 %
Grain	28	97.74%	77.69%	75.37%	76.52%
Crude	28	96.61%	73.61%	65.84%	69.51%
Trade	39	96.32%	56.04%	45.54%	50.25%
Interest	35	97.34%	68.00%	51.00%	58.29 %
Ship	26	98.21%	80.00%	56.47%	66.21%
Wheat	19	98.72%	70.67 %	80.30%	75.18%
Corn	27	99.87 %	68.09%	66.67%	67.37%
Average	32.10	97.43%	75.51%	67.68%	70.95%

Reuters datasets presented in Section 1.6.2, Reuters-21578 and RCV1 corpora. A discussion is put forward on the comparison of the behavior of each kernel-based machine.

2.4.1 SVM Performance

SVMs were tested using the SVMlight software[3], first with the default parameters and then doubling the importance of the least represented class, i.e. the positive class. Tables 2.1 and 2.3 display the performance results for both datasets (see Section 1.6.1 for insight on the performance measures). These results were obtained using the ten most populated categories for both datasets (see Appendixes A and B). To provide a complexity measure of the resulting models, the number of support vectors (SVs) is also presented.

[3] http://svmlight.joachims.org/

Analyzing the performance measures, it is easy to conclude that accuracy values offer a deceptive notion of correctness of the resulting models, when compared with other measures. For instance, for the corn category in Reuters-21578, even though an accuracy of 99.12% is reached, only 58.33% of the 52 positive testing documents are detected (see Table 1.3 for information on the number positive examples in Reuters-21578 categories). Another effect that can be observed is that more positive documents in a category result in more complex models, i.e. more SVs.

Comparing the performances in Reuters-21578 and RCV1, it can be argued that the RCV1 corpus is more difficult to classify, since the SVM models have many more SVs for similar training set sizes. While the average number of SVs for Reuters-21578 is between 600 and 900, for RCV1 the average number of SVs is between 3,000 and 5,000. Moreover, the F1 averaged performance is around 10% lower for RCV1 (71.77% to 61.91% and 79.68% to 69.43%).

Table 2.3 Performances for SVM with RCV1.

Default parameters

Category	SVs	Accuracy	Precision	Recall	F1
CCAT	6,919	93.02%	93.33%	92.24%	92.78%
GCAT	3,893	92.96%	78.26%	44.37 %	56.63%
MCAT	4,466	95.28%	93.67 %	87.94%	90.72%
C15	3,974	91.81%	92.46 %	58.85%	71.92%
ECAT	4,723	93.31%	88.00 %	65.30%	74.97%
M14	3,060	95.57%	90.49%	60.33 %	72.39%
C151	1,148	96.90%	66.74%	9.10%	16.01%
C152	2,422	97.23%	20.67%	2.96%	5.18%
GPOL	2,252	95.94%	78.28%	61.34 %	68.78 %
M13	2,126	96.69%	87.24%	58.01%	69.69 %
Average	3,498.3	94.87%	78.91%	54.04%	61.91%

Doubled importance of errors on the positive class

Category	SVs	Accuracy	Precision	Recall	F1
CCAT	6,791	92.52%	90.41%	94.17%	92.25%
GCAT	4,942	92.72%	68.39%	57.32 %	62.37%
MCAT	4,878	95.59%	89.70 %	93.08%	91.36%
C15	5,144	94.43%	93.53 %	67.79%	78.61%
ECAT	5,657	93.76%	82.85 %	77.55%	80.11%
M14	3,141	96.11%	90.53%	69.04 %	78.34%
C151	1,855	98.78%	59.44%	39.57%	47.51%
C152	3,961	95.59%	3.78%	16.22%	6.13%
GPOL	2,959	96.31%	76.87%	73.42 %	75.11 %
M13	2,840	97.89%	87.49%	78.03%	82.49 %
Average	4,216.8	95.37%	74.30%	66.62%	69.43%

2.4.2 RVM Performance

RVMs were tested using Tipping's software[4]. Given the RVMs' Bayesian roots, all training set documents must be evaluated at once, and this results in scalability issues. Making the learning faster and computationally less demanding is therefore an ongoing research area.

We carried out a set of experiments to determine how many documents to use in RVM training. Figure 2.5 is a cubic model for the computing time fitted to the experimental data, obtained using the text classification benchmark Reuters-21578, for different training set sizes.

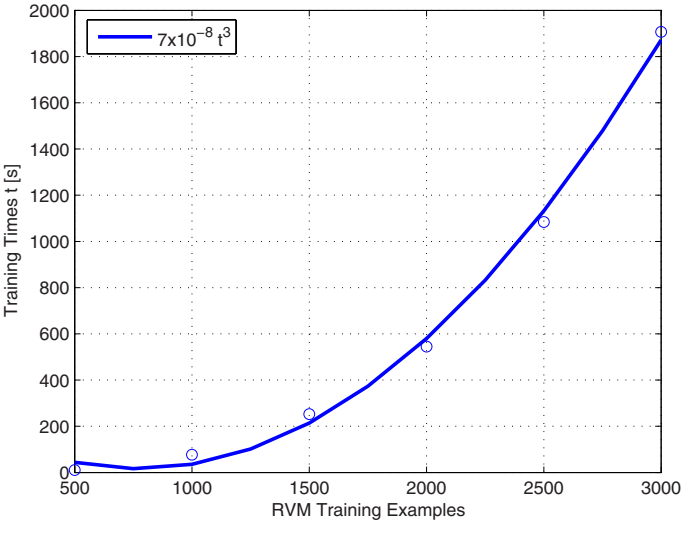

Fig. 2.5 Model of RVM training times on Reuters-21578.

Observing this model conclusions can be drawn on the importance of limiting the training set size to indispensable items. Thus, in the RVMs baseline results we have used just 1,000 or 2,000 training documents, which already takes 600 seconds, instead of the full ModApte split training set used by SVMs.

Tables 2.2 and 2.4 show performance measures (see Section 1.6.1) for RVMs on Reuters-21578 and RCV1 corpora respectively.

These results were obtained using the ten most populated categories for both datasets (see Appendixes A and B). To provide a complexity measure of the resulting models relevance vectors (RVs) are also presented.

Analyzing the performance measures, there are some common points with SVMs, in particular that accuracy values are deceptive when compared with other measures, and that more positive documents in a category result in more complex models, i.e. more RVs.

[4] http://www.miketipping.com/index.php?page=rvm

Table 2.4 Performances for RVM with RCV1.

1,000 training documents

Category	RVs	Accuracy	Precision	Recall	F1
CCAT	68	80.18 %	75.26%	85.00 %	79.83%
GCAT	63	88.42%	42.50 %	33.47%	37.45%
MCAT	51	87.49%	73.25 %	68.87 %	70.99%
C15	42	90.89%	69.17 %	70.16%	69.66%
ECAT	45	85.66%	52.70 %	47.50%	49.97%
M14	38	91.14 %	67.56 %	18.03 %	28.46%
C151	14	98.25%	24.79 %	12.61%	16.71%
C152	11	99.02%	13.51 %	2.30%	3.93%
GPOL	23	93.98 %	64.27%	44.78 %	52.79%
M13	22	95.53%	57.30 %	56.45 %	56.87 %
Average	37.70	91.06%	54.03%	43.92%	46.67%

2,000 training documents

Category	RVs	Accuracy	Precision	Recall	F1
CCAT	106	84.04 %	81.66%	84.35 %	82.99%
GCAT	81	88.71%	43.56 %	30.42%	35.83%
MCAT	92	89.46%	76.17 %	76.62 %	76.36%
C15	64	92.55%	72.26 %	81.19%	76.47%
ECAT	88	87.98%	60.01 %	60.86%	60.43%
M14	64	97.61 %	26.27 %	39.26 %	31.48%
C151	28	92.76%	66.30 %	52.69%	58.21%
C152	17	98.84%	8.67 %	3.45%	4.93%
GPOL	52	93.95 %	61.11%	53.54 %	55.07%
M13	41	96.48%	66.45 %	65.79 %	66.12 %
Average	62.50	92.24%	56.24%	54.82%	55.04%

Comparing the performances in Reuters-21578 and RCV1, again RCV1 appears more complex with an average F1 15.91% lower, for the models trained with 2,000 documents.

Moreover, RCV1 has more RVs per model; for Reuters-21578 the average number of RVs is between 19 and 32, whereas for RCV1 the average number of RVs is between 37 and 62. Note that, disregarding this difference between datasets, the number of RVs in RVMs is much smaller than the number of SVs in SVMs, which shows the lesser overall model complexity, making RVMs suitable for real online applications.

2.4.3 Discussion

Table 2.5 presents a summary of average default baseline measures for both datasets and kernel-based machines. Reuters-21578 and RCV1 exhibit similar trends for both

Table 2.5 Comparison of baseline SVM and RVM performances.

	Reuters-21578		RCV1	
	Vectors	F1	Vectors	F1
SVM	861.50	71.77%	3498.3	61.91%
RVM	32.10	70.95%	62.50	55.04%

SVMs and RVMs. SVMs have a better F1 classification performance, but have much more complex models. The performance gains for SVMs range between around 1% and 7%, while the number of vectors is 26 to 56 times higher. Therefore, one can say there is a compromise between performance and complexity that can favor either SVMs or RVMs, depending on the application and objectives. Note also that RVMs were trained with only 2,000 documents, i.e. around 30% of the available training set, and that more computational power should result in better performances. Moreover, RVMs offer other advantages, especially probabilistic outputs.

2.5 Conclusion

This chapter has described the kernel-based learning machines used throughout this book. First the concept of kernel methods was introduced and then the foundations of support vector machines and relevance vector machines were more detailed in a text classification perspective.

Kernel methods constitute state-of-the-art learning algorithms. The concepts of kernels and the kernel trick permit their successful application to practically any given application. For text classification, support vector machines (SVMs), making use of the *Structural Risk Minimization* principle, effectively reduce the training error, bounding the real error obtained when facing new unseen examples. Moreover, SVMs scale independently of the dimensionality of the feature space, since the separating margin defines the complexity.

On the other hand, relevance vector machines (RVMs) offer good generalization performance, together with a fully probabilistic framework that provides sparse models.

To compare the two kernel approaches, baseline classification results in two accepted text classification benchmark corpora were presented and analyzed, allowing for future reference in subsequent chapters. Performance-wise SVMs consistently presented better classification results. Conversely, RVMs were found to have sparser models with probabilistic outputs, which can become relevant in real-world applications.

Chapter 3
Enhancing SVMs for Text Classification

Abstract. The previous chapter introduced kernel-based techniques and their baseline application to text classification. In this chapter we develop and explore learning techniques that integrate knowledge in the classification task to improve the performance of support vector machines (SVMs) in text classification applications.

The introduction of unlabeled data in the learning stage is investigated. With the deluge of digital text data, unlabeled texts are ubiquitous. Whether it is the Internet, email servers, database files or plain file systems, the sources for digital texts are countless. However, such texts are usually unlabeled, and their labeling is mostly manual and costly. Therefore, a research field on the study and use of these unlabeled texts has been emerging. It is further exploited the potential of using several learning machines organized in a committee. Knowing that there is no unique classifier that suits all situations, the focus is on using the diversity of classifiers to enhance performance.

3.1 Incorporating Unlabeled Data

Supervised leaning algorithms for text classification, like SVMs, usually require a large number of labeled documents to achieve a satisfactory performance. However, labeling documents to create a training set is time consuming and costly, since human labor is usually needed. This section demonstrates that supervised learning algorithms that use a small number of labeled examples and many inexpensive unlabeled examples, usually termed as partially supervised learning methods, can create competitive text classifiers.

In general, unlabeled examples are much less expensive and easier to gather than labeled ones. This is particularly true for text classification tasks involving online data sources, such as web pages, emails or news stories, where large amounts of texts are readily available. Labeling examples can also be usefully limited in applications like email routing, Spam filtering or web page labeling, saving time and money for users and companies, while providing a vital tool for dealing with the avalanche of digital texts they are faced with.

C. Silva and B. Ribeiro: Inductive Inference for Large Scale Text Classification, SCI 255, pp. 51–70.
springerlink.com

The unlabeled texts can frequently be collected automatically, so it is feasible to collect a large set of unlabeled examples. If unlabeled examples can be integrated into supervised learning, then building text classification systems will be significantly faster, less expensive and more effective. Consider the following example to give some insight into how unlabeled data can be useful. Suppose we are interested in recognizing web pages about conferences. We are given just a few conference and non-conference web pages, along with a large number of pages that are unlabeled. By looking at just the labeled data, we determine that pages containing the word *paper* tend to be about conferences. If we use this fact to estimate the classification of the many unlabeled web pages, we might find that the word *deadline* occurs frequently in the documents that are classified in the positive class. This co-occurrence of the words *paper* and *deadline* over the large set of unlabeled training data can provide useful information to construct a more accurate classifier that considers both *paper* and *deadline* as indicators of positive examples. To successfully integrate information of unlabeled examples in the learning task, several research techniques have been pursued.

In [97], Nigam et al. propose an algorithm for learning from labeled and unlabeled documents based on the combination of expectation maximization and naïve Bayes classifier where, after an initial model is derived, the probabilistic labels of unlabeled examples are used iteratively until a new model is reached.

Tong & Koller [148] introduce an algorithm for performing active learning with support vector machines, defining the version space as the set of possible consistent hyperplanes that separate the data in the induced feature space, and taking advantage of the duality between the parameter space and feature space. Empirically they show that at each iteration the version space is reduced, thus reducing the need for labeled instances in both the standard inductive and transductive settings.

In [78], Liu et al. address the problem of building text classifiers using positive and unlabeled examples. The key feature of this problem is that there are no negative examples. The authors propose a biased formulation of the SVM, where the number of unlabeled examples classified as positive is minimized, while constraining the positive examples to be correctly classified.

Zhang & Oles [166] focus on a statistical point of view of using unlabeled data for classification, assuming there is a correct model of the underlying distribution, and apply the methodology to partially supervised learning and active learning settings. The authors use a probabilistic framework, using the logistic model as an approximate probability model for SVMs with Fisher information. It is concluded that, using this setting, SVMs are appropriate for active learning, but not for partially supervised learning, since unlabeled data points are likely to cause large changes in parameter estimation once the label is determined.

König & Brill [59] describe a way to reduce the labeling effort, while retaining accuracy, by constructing a hybrid classifier that utilizes human reasoning on automatically discovered text patterns to complement machine learning. Using a standard sentiment-classification dataset and real customer feedback data, it is demonstrated that the technique results in significant reduction of the human effort required to obtain a given classification accuracy. Moreover, the hybrid text

classifier also results in a significant boost in accuracy over machine-learning based classifiers when a comparable amount of labeled data is used.

In [169], Zhou et al. propose a general framework for learning from labeled and unlabeled data on a directed graph. The structure of the graph includes the directionality of the edges and it can be utilized as a spectral clustering method for directed graphs, which generalizes the spectral clustering approach for undirected graphs.

Szummer [144] presents a comprehensive study on the use of learning from partially labeled data. The author introduces three approaches to the problem: a kernel classifier that can be interpreted as a discriminative kernel density estimator, trained via the expectation-maximization (EM) algorithm; a Markov random walk representation that exploits clusters and low-dimensional structure in the data; and a non-parametric information regularization technique based on minimizing information about labels on regions covering the domain.

Specifically for SVMs there is a transductive setting, introduced by Vapnik [152] and Joachims [53]. Transductive support vector machines (TSVMs) take into account a particular testing set and try to minimize missclassifications of just those particular examples. Given that TSVMs are a straightforward improvement of SVMs, they will also be used for baseline comparison in the strategies pointed out below.

3.1.1 Background Knowledge and Active Learning

In the following two techniques that include knowledge in the learning task by incorporating unlabeled examples are proposed and compared, viz. background knowledge and active learning. As for other unlabeled approaches, the underlying idea is that the information contained in unlabeled documents can help to improve the classification performance. Both techniques are margin-based partially labeled approaches, i.e. the significance is put on the separating margin of SVMs and both labeled and unlabeled data are used.

The separating margin, defined by the optimal separating hyperplane (OSH) and the nearest training examples (see Section 2.2, Figure 2.2), is determinant in SVMs and basically defines their performance. When classifying unlabeled examples with an SVM model, the support vectors (SVs) and associated weights are used to determine on which side of the OSH the unlabeled examples fall. Figure 3.1 depicts a two-dimensional example where four unlabeled documents, represented by black dots, are classified. As can be gleaned from the figure, not all unlabeled documents are positioned at the same distance from the OSH. Two are close to the margin (represented with a *small margin* label) and other two are farther away (represented with a *large margin* label).

For both the proposed approaches, first an SVM model is induced and applied to all examples (labeled and unlabeled). Then, the two approaches add examples from the unlabeled/testing set to the training set. The approaches differ in the way the incorporated examples are chosen and in the number of examples added.

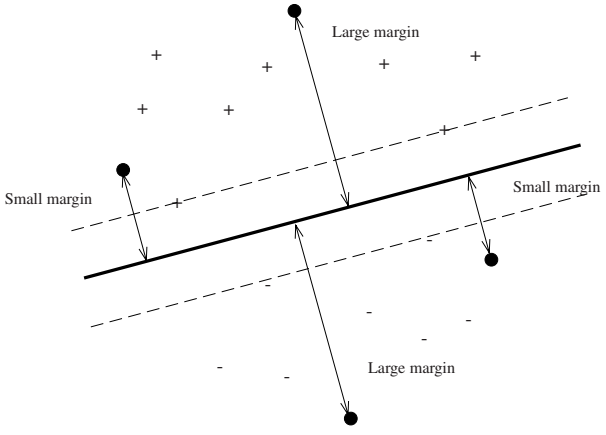

Fig. 3.1 Unlabeled examples (black dots) classified by an SVM model with small and large margins.

3.1.1.1 Background Knowledge

After the classification of unlabeled examples by the initial SVM model, the background knowledge approach incorporates in the training set new examples that are classified farther from the OSH (see Figure 3.1). The rationale is that these examples have a high probability (high confidence) of being well classified, and can thus be interpreted as information underlying the initial problem. The procedure is as follows: examples (only the features, not the label) from the unlabeled set are directly incorporated into the training set as classified by the baseline inductive SVM, i.e. a document $\mathbf{d}_i$ will be chosen if Equation (3.1) holds.

$$\mathbf{d}_i : \rho_i = \frac{2}{\|\boldsymbol{\omega}\|} > \Delta_1, \tag{3.1}$$

where ρ_i represents the margin with which an unlabeled document $\mathbf{d}_i$ is classified, $\boldsymbol{\omega}$ represents the associated weights and Δ_1 is a task dependent parameter. These examples will increase the information about the background of the particular category, and that is the reason for the background knowledge denomination. The SVM model is then retrained with the training set enlarged by the newly added examples, using as target the classification suggested by the initial SVM model.

Incremental background knowledge

An enhancement over background knowledge is an incremental background knowledge technique that alters the definition of Δ_1. Fig. 3.2 illustrates the iterative procedure[138]. As can be gleaned from this figure, the training set is incrementally constructed by iteratively decreasing the value of Δ_1, i.e. reducing the confidence threshold for an unlabeled example to be added as classified by the SVM.

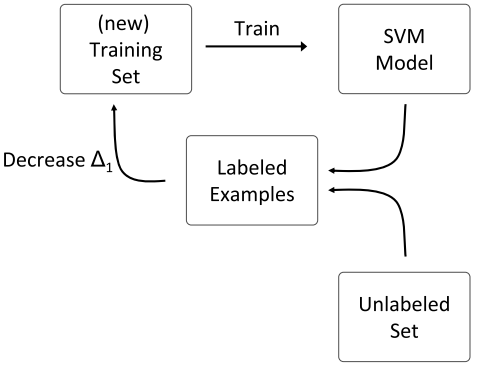

Fig. 3.2 Proposed approach: Incremental Background Knowledge.

Algorithm 1. Incremental Background Knowledge Algorithm.

Current training set ← Initial dataset
$\Delta \leftarrow$ initial Δ value
WHILE not all unlabeled examples added
    Infer an SVM classifier with current training set
    Classify unlabeled examples with the classifier
    Select the newly classified examples with margin larger that Δ
    Add the selected examples to the current training set
    Decrease Δ
ENDWHILE

This approach rational is that as Δ_1 is decreased, the classifiers are also getting better due to the additional information in the training set, thus justifying lowering the confidence threshold. Algorithm 1 below more formally defines the incremental background knowledge procedure.

3.1.1.2 Active Learning

As with the previous approach, initially the SVM model classifies all unlabeled examples. Then, the active learning approach selects a set of examples that the supervisor is asked to classify in an active learning setting. The examples are chosen from the examples with the smaller margin (see Figure 3.1), which offer more doubts in their classification by the initial SVM model. Thus, an example $\mathbf{d}_i$ will be selected for active learning if Equation (3.2) holds.

$$\mathbf{d}_i : \rho_i = \frac{2}{\|\boldsymbol{\omega}\|} < \Delta_2, \tag{3.2}$$

where ρ_i represents the margin with which an unlabeled document $\mathbf{d}_i$ is classified, $\boldsymbol{\omega}$ represents the associated weights and Δ_2 is a task dependent parameter. This number of documents selected cannot be large, since the supervisor will be asked to classify

them manually. Therefore, in addition to the above constraint, an application and user dependent threshold should also be set. After being correctly classified, the selected documents are integrated into the training set with the correct classification given by the supervisor, and the SVM is retrained. This approach can be regarded as a form of active learning, where the information that an example can introduce in the classification task is considered to be inversely proportional to its classification margin.

3.1.2 Experimental Results

The use of both unlabeled examples approaches was evaluated in the Reuters-21578 benchmark data set with the ModApte split. The use of unlabeled examples is more important when few labeled examples are present. Therefore, in addition to the ModApte split, a Small split was also defined to reproduce a real situation in which a user would be asked to provide the examples. In the Small split, the testing set is the same for the sake of comparison, but the training set is defined for each category by randomly selecting 10 positive and 10 negative examples of the category. The baseline of comparison will be the results obtained with the SVM in the inductive setting presented in Chapter 2, and duly repeated in Table 3.1 for the ModApte and Small splits. The unlabeled set was defined as the testing set.

3.1.2.1 Background Knowledge Results

The background knowledge approach has a parameter Δ_1 (see Equation (3.1)) to be defined. Empirically we considered $\Delta_1 = 0.6$, representing 60% of confidence. Table 3.2 presents the background knowledge approach results with both training/testing splits. Analyzing F1 values, there is a slight improvement (from 71.77% to 72.50%) for the ModApte split, but not for the Small split, where there is a decrease (from 32.56% to 20.82%). For this approach to be successful the baseline classifier should not be too weak, since it will be responsible for classifying the unlabeled examples. That is not the case with Small split. With only 20 examples, the initial classifier is not accurate enough to determine new training examples.

Incremental background knowledge results

For incremental background knowledge, the values of Δ were varied starting with no inclusion of new unlabeled examples until practically all available examples were added. In the background knowledge approach, for each value of Δ a new training set was constructed, learned and tested, while for incremental background knowledge the training set in each iteration, corresponding to a value of Δ, was used as baseline for the next iteration, where it was again incremented (see Fig. 3.2).

Table 3.1 Baseline SVM performances with Reuters-21578.

ModApte Split

Category	SVs	Accuracy	Precision	Recall	F1
Earn	1,632	95.92%	93.53%	95.50%	94.50%
Acquisitions	1,751	94.93%	93.09%	85.15%	88.94%
Money-fx	908	96.13%	71.43%	52.80%	60.72%
Grain	771	97.96%	92.55%	63.04%	75.00%
Crude	693	97.04%	84.85%	63.64%	72.73%
Trade	647	97.64%	79.49%	54.87%	64.92%
Interest	742	97.15%	77.03%	47.11%	58.46%
Ship	500	98.45%	89.36%	51.85%	65.62%
Wheat	487	98.77%	84.44%	57.58%	68.47%
Corn	484	99.08%	93.33%	53.85%	68.29%
Average	861.50	97.31%	85.91%	62.24%	71.77%

Small Split

Category	SVs	Accuracy	Precision	Recall	F1
Earn	19	90.32%	90.26%	82.57%	86.24%
Acquisitions	19	49.77%	32.07%	98.24%	48.35%
Money-fx	18	38.33%	8.11%	95.65%	14.95%
Grain	20	81.31%	16.06%	67.39%	25.94%
Crude	18	70.50%	15.52%	84.66%	26.23%
Trade	18	79.41%	15.50%	93.81%	26.60%
Interest	18	54.10%	8.02%	93.39%	14.77%
Ship	19	32.31%	3.90%	96.30%	7.50%
Wheat	19	95.49%	29.61%	68.18%	41.29%
Corn	20	98.20%	52.00%	25.00%	33.77%
Average	18.80	68.97%	27.11%	80.52%	32.56%

Fig. 3.3 shows the F1 performance for both approaches and for the several values of Δ. Notice that the values of Δ in x-axis are decreasing values, reflecting the nature of the incremental background knowledge technique that starts to add large margin classified examples (with high confidence and large Δ values) and proceeds with decreasing values of margin, confidence and Δ. It is clear from this figure that the incremental background knowledge generally surpasses basic background knowledge. However, the most compelling analysis is in its stability to different values of confidence. While basic background knowledge presents sensitivity when the confidence drops below an acceptable threshold, incremental background knowledge remains fairly insensitive and stable, even when all examples are added. It is possible to assert the number of added examples and of these how many are wrongly classified in the graphic in Fig. 3.4. It is interesting to notice that although more examples are added in the basic background knowledge, the number of examples introduced with

Table 3.2 Background knowledge performances with Reuters-21578.

ModApte Split

Category	SVs	Accuracy	Precision	Recall	F1
Earn	1,651	95.85%	93.27%	95.59%	94.42%
Acquisitions	1,800	95.04%	92.71%	86.03	89.25%
Money-fx	928	96.13%	71.07%	53.42%	60.99%
Grain	802	98.03%	92.71%	64.49%	76.07%
Crude	697	97.18%	85.29%	65.91%	74.36%
Trade	661	97.68%	79.75%	55.75%	65.62%
Interest	744	97.22%	76.92%	49.59%	60.30%
Ship	505	98.49%	89.58%	53.09%	66.67%
Wheat	490	98.77%	82.98%	59.09%	69.03%
Corn	505	99.08%	93.33%	53.85%	68.29%
Average	878.30	97.35%	85.76%	63.68%	72.50%

Small Split

Category	SVs	Accuracy	Precision	Recall	F1
Earn	42	90.14%	82.43%	93.01%	87.40%
Acquisitions	92	40.65%	28.63%	99.12%	44.43%
Money-fx	44	25.31%	6.83%	96.27%	12.76%
Grain	23	37.35%	6.74%	92.75%	12.57%
Crude	31	37.63%	8.87%	97.73%	16.26%
Trade	36	21.89%	4.81%	99.12%	9.17%
Interest	33	22.74%	5.11%	97.52%	9.71%
Ship	86	30.83%	3.82%	90.30%	7.35%
Wheat	24	5.03%	2.39%	100.00%	4.67%
Corn	23	9.43%	1.98%	100.00%	3.88%
Average	43.50	32.10%	15.16%	97.18%	20.82%

the wrong classification in the training sets are fairly low and equivalent, despite the difference in classification performance.

3.1.2.2 Active Learning Results

The threshold Δ_2 for active learning in Equation (3.2) was empirically defined as 0.5. Table 3.3 presents the results obtained for both training/testing splits. As already mentioned, in addition to this constraint the number of documents selected to be classified by the supervisor was limited to a maximum of 40. The improvement is more relevant (improvement of 37%, from 32.56% to 44.61%) in the Small split than the ModApte split, a predictable outcome, since the training set was substantially increased (20 initial examples plus 40 examples actively chosen to be classified by the supervisor). Regarding the ModApte split the active approach improves 10% of the baseline results (from 71.77% to 78.61%).

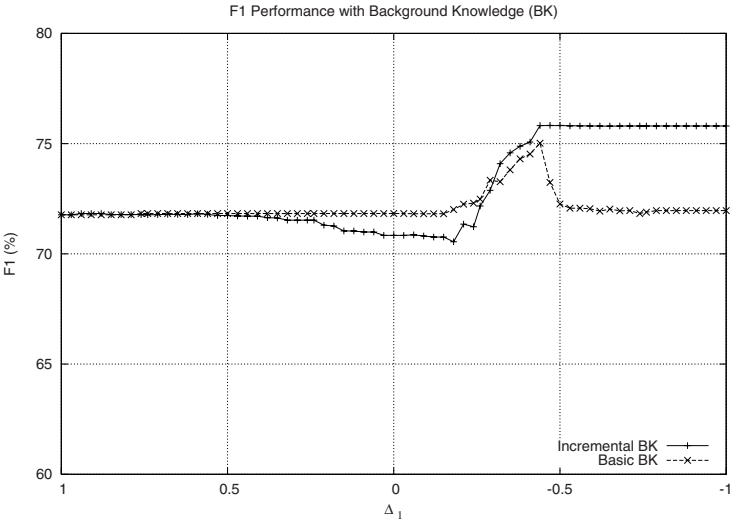

Fig. 3.3 F1 performance for incremental background knowledge and background knowledge for different Δ_1 values.

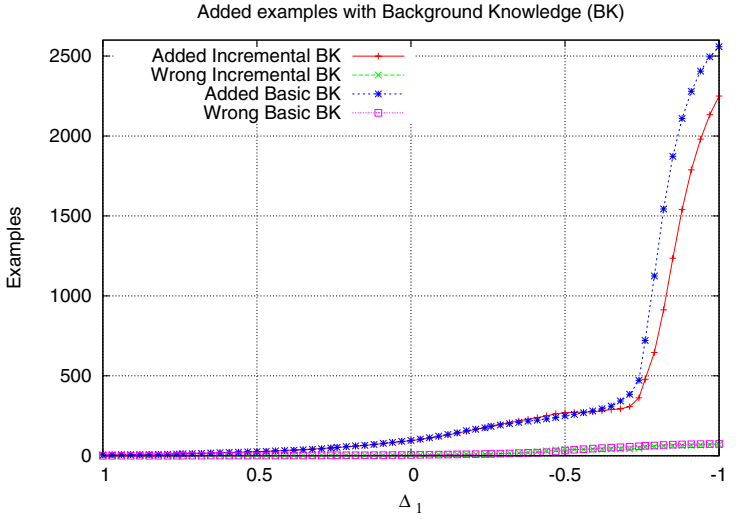

Fig. 3.4 Total added examples and wrongly added examples for incremental background knowledge and basic background knowledge for different Δ values.

3.1.3 Combining Background Knowledge and Active Learning

Table 3.4 presents the performances resulting from the combination of the two approaches for both training/testing splits. The combination of both techniques was

Table 3.3 Active learning performances with Reuters-21578.

ModApte Split

Category	SVs	Accuracy	Precision	Recall	F1
Earn	1,662	96.27%	94.00%	95.98%	94.98%
Acquisitions	1,788	95.28%	94.03%	85.74%	89.69%
Money-fx	947	96.59%	76.67%	57.14%	65.48%
Grain	791	98.28%	94.95%	68.12%	79.33%
Crude	719	97.43%	88.72%	67.05%	76.38%
Trade	676	98.20%	83.70%	68.14%	75.12%
Interest	777	97.64%	84.62%	54.55%	66.34%
Ship	545	98.84%	92.86%	64.20%	75.92%
Wheat	482	99.19%	90.57%	72.73%	80.68%
Corn	537	99.44%	97.37%	71.15%	82.22%
Average	892.40	97.72%	89.75%	70.48%	78.61%

Small Split

Category	SVs	Accuracy	Precision	Recall	F1
Earn	54	92.64%	86.78%	94.35%	90.41%
Acquisitions	56	57.76%	35.92%	97.50%	52.50%
Money-fx	56	93.95%	48.15%	88.82%	62.45%
Grain	55	66.42%	12.01%	93.48%	21.29%
Crude	54	57.48%	11.83%	90.91%	20.94%
Trade	55	95.04%	43.81%	87.61%	58.41%
Interest	53	75.99%	13.71%	87.60%	23.71%
Ship	47	70.47%	8.62%	97.53%	15.84%
Wheat	52	95.39%	32.43%	90.91%	47.81%
Corn	56	97.92%	45.21%	63.46%	52.80%
Average	53.80	80.31%	33.85%	89.22%	44.61%

expected to take advantage of their benefits. In these circumstances, the training sets (ModApte and Small) were enriched with items classified both by the baseline SVM (background knowledge) and by the user (active learning). The combined approach performs better with ModApte split. It surpasses the baseline results (71.77%) around 11%, reaching 79.66%. The combined approach with Small split presents poorer results than the single active learning approach, 34.29% compared with 44.61%, confirming that the baseline classifier is too weak to be used as a background knowledge incorporator.

3.1.4 Analysis of Results

Table 3.5 presents a summary of the F1 average performance for the baseline and unlabeled approaches with both training/testing splits. The active learning approach offers enhancements for both splits, while the background knowledge technique

Table 3.4 Combined background knowledge and active learning performances with Reuters-21578.

ModApte Split

Category	SVs	Accuracy	Precision	Recall	F1
Earn	1,682	96.30%	94.00%	96.07%	95.02%
Acquisitions	1,836	95.46%	93.52%	87.06%	90.17%
Money-fx	961	96.69%	77.24%	59.01%	66.91%
Grain	806	98.28%	94.95%	68.12%	79.33%
Crude	721	97.71%	89.36%	71.59%	79.49%
Trade	690	98.20%	83.70%	68.14%	75.12%
Interest	780	97.61%	83.54%	54.55%	66.00 %
Ship	547	98.98%	91.94%	70.37%	79.72 %
Wheat	491	99.26%	90.91%	75.76%	82.65 %
Corn	537	99.44%	97.37%	71.15%	82.22%
Average	905.10	97.79%	89.65%	72.18%	79.66%

Small Split

Category	SVs	Accuracy	Precision	Recall	F1
Earn	61	92.50%	87.13%	93.39%	90.15%
Acquisitions	105	44.84%	30.10%	98.68%	46.13%
Money-fx	78	34.25%	7.73%	96.89%	14.32%
Grain	57	54.17%	9.07%	93.48%	16.54%
Crude	67	51.14%	10.55%	92.05%	18.93%
Trade	70	90.50%	28.25%	90.27%	43.03%
Interest	65	47.45%	7.07%	93.39%	23.71%
Ship	130	34.60%	4.08%	97.53%	7.83 %
Wheat	56	93.59%	25.42%	90.91%	39.73 %
Corn	59	98.13%	49.18%	57.69%	53.46%
Average	74.80	64.12%	25.86%	90.43%	34.29%

Table 3.5 F1 average performance for the baseline and unlabeled approaches with Reuters-21578 ModApte and Small splits.

	Baseline SVM	Background Knowledge	Active Learning	Combined Approaches
ModApte	71.77%	72.50%	78.61%	79.66%
Small	32.56%	20.82%	44.61%	34.29%

should only be applied when the baseline classifier has a reasonably acceptable performance, which only happens for ModApte split.

Following the comparison with the baseline SVMs, we now present a comparison with Transductive SVM (TSVM) [152, 53]. Table 3.6 presents the comparison of Background Knowledge, Active Learning and their combination, presented in [128].

Analyzing Table 3.6, it can be concluded that the proposed approaches are an improvement over TSVM, and thus constitute a valid learning approach.

Table 3.6 Comparison of the proposed approaches with TSVM using Reuters-21578 ModApte Split.

Category	Transductive SVM	Background Knowledge	Active Learning	Combined approaches
Earn	94.46%	94.42%	94.98%	95.02%
Acquisitions	89.38%	89.69%	98.24%	90.17%
Money-fx	70.22%	60.99%	65.48%	66.91%
Grain	78.49%	76.07%	79.33%	79.33%
Crude	75.63%	74.36%	76.38%	79.49%
Trade	69.88%	65.62%	75.12%	75.12%
Interest	76.28%	60.30%	66.34%	66.00%
Ship	79.49%	66.67%	75.92%	79.72%
Wheat	78.96%	69.03%	80.68%	82.65%
Corn	67.24%	68.29%	82.22%	82.22%
Average	78.00%	78.61%	80.52%	79.66%

To graphically compare the proposed approaches, a ROC graph was constructed using the true positive rates and false positive rates in Table 3.7, and this is presented in Figure 3.5. As can be gathered from the ROC graph, the Small split consistently shows a better true positive rate, gained at the expense of an also larger false positive rate. This difference appears due to the balanced training set that models a real user-provided set of classified examples. For the ModApte splitt, all settings present a very precise behavior (low false positive rate), and the combined approach presents the best true positive rate [126].

Table 3.7 False positive rate (FPR) and true positive rate (TPR) for the unlabeled approaches with Reuters-21578 ModApte and Small splits.

	Baseline SVM		Background Knowledge		Active Learning		Combined Approaches	
	FPR	TPR	FPR	TPR	FPR	TPR	FPR	TPR
ModApte	0.01	0.63	0.01	0.64	0.01	0.70	0.01	0.72
Small	0.33	0.81	0.73	0.97	0.21	0.89	0.39	0.90

Finally, for a straight comparison of the two proposed approaches, we defined the following criteria [129, 130]:

1. **User interaction:** while the background knowledge is automated, active learning needs some user interaction, since the selected items must be classified by the supervisor;
2. **Correctness of training set:** background knowledge does not guarantee its correctness, since the added examples are classified by the inductive SVM, whereas for active learning all examples in the training set are (correctly) classified by the supervisor;

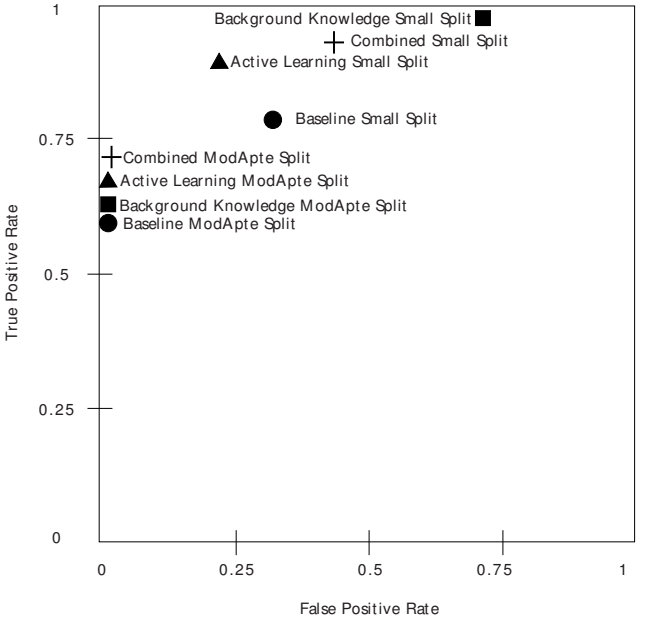

Fig. 3.5 ROC graph for the baseline SVM inductive classifiers (black circles), background knowledge (black squares), active learning (black triangles) and combined approaches (plus signs).

3. **Computational time:** there is no significant difference in the computational time taken, although the background knowledge approach can take longer, because the examples are automatically classified and there is no limit on the number of examples added;
4. **Performance:** active learning has greater potential, since the information added is more reliable, but there are limitations on the number of items the supervisor can tolerate/is able to classify.

The background knowledge method has the advantage of being completely automated. However, it should not be used with small training sets, i.e. with too weak initial classifiers. When this is not the case it can introduce an improvement. The proposed margin-based active learning method has potential to substantially improve performance when small training sets are available. This conclusion is very important in text mining tasks, since usually there is a small number of classified examples and a huge number of unlabeled ones.

3.2 Using Multiple Classifiers

There are circumstances in which the outputs of all algorithms solving a particular type of problem are statistically identical. A way of describing this situation, introduced in [112] for machine learning and in [160] in connection with the problems

of search and optimization, is to say that there is *no free lunch*. In a particular problem, different algorithms may yield different results, but over all problems, they are indistinguishable. It follows that if an algorithm achieves superior results in some problems, it must pay with inferiority on other problems. More formally, there is *no free lunch* when the probability distribution on problem instances is such that all problem solvers have identically distributed results. Consequently it would be of little use to optimize a classifier to a specific problem, since there would be no guarantee of generalizing its capabilities, or as stated in plain language by Wolpert and Macready themselves, "any two algorithms are equivalent when their performance is averaged across all possible problems" [161]. Using several classifiers is therefore appropriate, when it is possible to take advantage of each classifier's benefits and avoid the errors.

Given a machine learning problem, if we have some knowledge of the problem background, we may be able to exploit and integrate that knowledge in the selection or parameterization of the algorithms. This knowledge integration can therefore enhance the performance on that particular problem. Ensembles of learning machines, introduced in Section 1.5.7, are an excellent technique to exploit this knowledge integration by the use of multiple different classifiers. Diversity can be assured in several ways, particularly via different training sets, different training parameters, different types of classifiers or different features in the data representation. Moreover, different combinations of the diverse classifiers can be used [100].

The simplest way to combine diverse classifiers in text classification is majority voting, where the outputs of the k classifiers are joined together, and the classification decision that reaches half of the votes plus one is considered as output [75, 76]. This method is particularly suited to the case in which the committee is composed of binary classifiers, like SVMs.

Another combination method is weighted linear combination. The output is the sum of the decision functions produced by the k classifiers. The weighting of each classifier reflects its expected relative effectiveness, and is usually optimized on a validation set [64].

Another technique is dynamic classifier selection, where the most effective classifier on the validation examples closer to the testing document at hand is selected, and its judgment adopted by the committee [75].

Yet another method is adaptive classifier combination. In this case the judgments of all the classifiers in the committee are added together, but their individual contribution is weighted by their effectiveness in the validation examples most similar to the testing document at hand [75].

Classifier committees have had mixed results in text classification so far [122]. Larkey & Croft [64] have used combinations of Rocchio method, naïve Bayes, and k-NN, all together or in pairwise combinations, using weighted linear combination. In their experiments the combination of any two classifiers outperformed the best individual classifier (k-NN), and the combination of the three classifiers improved on all three pairwise combinations.

On the other hand, Li & Jain [75] have tested a committee formed of (various combinations of) a naïve Bayes classifier, an example-based classifier, a decision

tree classifier, and a classifier built by means of their own subspace method. To combine these classifiers they used majority voting, dynamic classifier selection and adaptive classifier combination. The best results were achieved with the adaptive classifier combination.

Based on AdaBoost [38], BoosTexter [113] constitutes a state-of-the-art algorithm for text classification. BoosTexter is a general purpose machine learning program based on boosting for building a classifier from text and/or attribute-value data. Boosting is deployed with very simple classifiers; specifically the weak hypotheses have the same basic form as a one-level decision tree. The test at the root of this tree is a simple check for the presence or absence of a term in the given document.

All words and pairs of adjacent words are potential terms. Based only on the outcome of this test, the weak hypothesis outputs the predictions and confidences that each label is associated with the document. And then, a boosting process similar to AdaBoost takes place (see Section 1.5.7.1).

3.2.1 SVM Ensembles

In this section the ensemble structure designed for the text classification problem is presented. The goal is to improve classification performance, which is possible when the base classifiers show different patterns of errors, since the errors made by one of them can be compensated by the correct output of others.

SVMs and ensemble methods exhibit state-of-the-art results in text classification. Therefore is it more than reasonable to try to join these two strategies to enhance performance. Since SVMs can handle large training sets and there is a reduced number of positive examples in the training set, data partitioning techniques would penalize performance and were therefore not considered.

To create the base SVM classifiers, we have explored different learning parameters. As SVMs are strong classifiers, and the number of learning parameters is reduced, we have defined four settings for the ensemble base classifiers, using the SVM light software package:

- SVM_1: Linear kernel with default parameters;
- SVM_2: RBF kernel with default parameters;
- SVM_3: Linear kernel with the trade-off between training error and margin set to 100;
- SVM_4: Linear kernel with the cost-factor, by which training errors on positive examples outweigh errors on negative examples set to 2.

This choice of base classifiers was not random. The SVM_1, despite being the simplest and fastest classifier, gives competitive results. The rationale for this result is that text classification problems, although highly dimensional, are nevertheless essentially linearly separable. As for SVMs, the complexity depends on the margin and not on the dimensionality, baseline SVMs have *per se* acceptable performance. RBF kernel SVMs are a more complex classifier, but for some specific applications constitute a major improvement over linear SVMs. Regarding SVM_3, the trade-off

between training error and margin was increased to 100 to try to control the errors. With respect to SVM_4, the false positives were the main concern in this setting, since their weight was doubled when compared with false positives.

The four SVM base classifiers were combined using the two step method proposed and described next. In the following equations, the classification margin with which an SVM_s model, $s \in \{1,2,3,4\}$, classifies a testing document $\mathbf{d}_i$, is represented by ρ_{si}.

Step 1

Give more weight to the positive classifications. Since there are few positive examples in the data set, the false negative rate is more critical and this step tackles this problem. The classification margin, (ρ_{si}), of classifier SVM_s in a testing document $\mathbf{d}_i$, is updated according to (3.3)

$$\begin{aligned} \forall_{s\in\{1,2,3,4\}} \quad & if \quad \rho_{si} > 0 \quad then \\ & \rho_{si} := K \times \rho(SVM_s). \end{aligned} \tag{3.3}$$

Step 2

Use the SVM presenting the largest margin, which implies more confidence in the result. For each testing example, the output classification of the ensemble learning machine, y_{ens}, will be given by the output of the base SVM ($y_{s^*}, s^* \in \{1,2,3,4\}$) with the maximum margin for each testing document $\mathbf{d}_i$ (3.4):

$$y_{ens} := y_{s^*} \quad with \quad s^* : \rho_{s^*i} = max(\rho_{si}), s \in \{1,2,3,4\} \tag{3.4}$$

3.2.2 Experimental Results and Analysis

The four base SVM classifiers were initially tested separately. Table 3.8 presents the F1 macro-averaged performances for the Reuters-21578 benchmark data set's ten most frequent categories, and Table 3.9 presents their possible combinations in the proposed ensemble setup.

The average base performances range between 79.65% and 83.38%. The linear base classifier presents the lower performance, while the classifier that emphasizes false negative errors (SVM_4) has the best results. Notice that in the less represented categories, such as ship, wheat and corn, the improvement is more noticeable. This is probably because these categories have very few positive examples, and the struggle against false negatives is more critical [133]. Regarding the proposed settings for combinations, for instance, A refers to an ensemble using only SVM_3 and SVM_4. Note that of the 2^4 possible combinations we left out those that used only one or none base classifier, since they can not be considered ensembles, given that they correspond to the base classifiers or no classifier at all. In the next results, the constant K in Equation (3.3) was empirically set to 100. Tables 3.10 and 3.11 present

Table 3.8 F1 performance for SVM ensemble base classifiers.

Category	SVM_1	SVM_2	SVM_3	SVM_4
Earn	98.14%	98.09%	98.14%	97.91%
Acquisitions	95.59%	95.43%	95.69%	95.75%
Money-fx	74.64%	75.81%	75.18%	77.35%
Grain	82.76%	82.76%	85.23%	85.71%
Crude	84.87%	84.87%	84.44%	87.12%
Trade	69.57%	72.12%	69.27%	74.21%
Interest	74.73%	75.41%	75.68%	76.68%
Ship	70.59%	71.53%	76.39%	79.19%
Wheat	75.63%	75.44%	78.69%	83.72%
Corn	70.00%	68.42%	74.70%	76.19%
Average	79.65%	79.99%	81.34%	83.38%

Table 3.9 Combinations of SVM base classifiers on ensembles.

	SVM_1	SVM_2	SVM_3	SVM_4
A	No	No	Yes	Yes
B	No	Yes	No	Yes
C	No	Yes	Yes	No
D	No	Yes	Yes	Yes
E	Yes	No	No	Yes
F	Yes	No	Yes	No
G	Yes	No	Yes	Yes
H	Yes	Yes	No	No
I	Yes	Yes	No	Yes
J	Yes	Yes	Yes	No
K	Yes	Yes	Yes	Yes

the F1 macro-averaged performances for each combination of base classifiers in the ensembles for the Reuters-21578 ten most frequent categories.

It is observed that the best stand-alone result SVM_4 with 83.38% (see Table 3.8) was surpassed by 6 (C, D, F, G, J, K) of the 11 possible combinations. In spite of having only four ensemble base classifiers, generated by different SVM parameterization, we have a positive outcome, since we have almost effortlessly improved the overall performance, i.e. from Table 3.1 for the ModApte split, the improvement is from 71.77% to 83.81%. To better analyze the ensemble combination, Table 3.12 presents the 11 settings ordered by F1 performance. One remarkable fact is that the best base classifier is not present in the best ensemble setting, but is, on the contrary, more present in some of the worst ones. A possible reasoning is that diversity is more important than individual performance. SVM_3 seems to have a crucial importance, i.e. seems to be the most different of the other base classifiers, since it is present in all the top settings. More generally, settings with three or four classifiers perform better, which indicates the generation of more base classifiers.

Table 3.10 F1 performances for SVM ensembles (A-F).

Category	A	B	C	D	E	F
Earn	97.91%	98.10%	97.91%	97.91 %	98.10%	97.91%
Acquisitions	95.75%	95.69%	95.75%	95.75 %	95.77%	95.75%
Money-fx	77.24%	76.49%	77.78%	77.24 %	76.06%	77.51%
Grain	85.71%	85.71%	86.18%	86.18 %	85.71%	86.18%
Crude	87.12%	84.44%	87.12%	87.12 %	84.44%	87.12%
Trade	73.87%	71.09%	75.00%	74.67 %	69.86%	74.21%
Interest	76.68%	75.68%	76.68%	76.68 %	75.68%	76.68%
Ship	79.19%	76.39%	79.19%	79.19 %	76.39%	79.19%
Wheat	83.72%	78.69%	83.72%	83.72 %	78.05%	83.72%
Corn	76.19%	74.70%	76.19%	76.19 %	77.65%	79.07%
Average	83.34%	81.70%	83.55%	83.47 %	81.77%	83.73%

Table 3.11 F1 performances for SVM ensembles (G-K).

Category	G	H	I	J	K
Earn	97.91%	98.09 %	98.10%	97.91%	97.91%
Acquisitions	95.75%	95.44 %	95.69%	95.75%	95.75%
Money-fx	77.24%	75.99 %	76.49%	77.51%	77.24%
Grain	86.18%	82.76 %	85.71%	86.18%	86.18%
Crude	87.12%	85.25 %	84.44%	87.12%	87.12%
Trade	73.87%	71.70 %	71.36%	75.00%	74.67%
Interest	76.68%	75.41 %	75.68%	76.68%	76.68%
Ship	79.19%	71.53 %	76.39%	79.19%	79.19%
Wheat	83.72%	75.63 %	78.05%	83.72%	83.72%
Corn	79.07%	70.00 %	77.65%	79.07%	79.07%
Average	83.67%	80.18 %	81.96%	83.81%	83.75 %

Table 3.12 Ordered F1 performances of the SVM ensembles.

	SVM_1	SVM_2	SVM_3	SVM_4	F1
J	Yes	Yes	Yes	No	83.81%
K	Yes	Yes	Yes	Yes	83.75%
F	Yes	No	Yes	No	83.73%
G	Yes	No	Yes	Yes	83.67%
C	No	Yes	Yes	No	83.55%
D	No	Yes	Yes	Yes	83.47%
A	No	No	Yes	Yes	83.34%
I	Yes	Yes	No	Yes	81.96%
E	Yes	No	No	Yes	81.77%
B	No	Yes	No	Yes	81.70%
H	Yes	Yes	No	No	80.18%

Table 3.13 Average false positives (FP) and false negatives (FN) for SVM ensembles.

	FP	FN
A	19.40	22.30
B	16.30	26.90
C	19.20	22.00
D	19.50	22.00
E	16.30	27.00
F	19.20	22.00
G	19.40	22.00
H	14.70	30.00
I	16.50	26.60
J	19.30	21.80
K	19.50	21.80

Focusing on the two best settings (J and K), Table 3.13, which shows the false positive and false negative macro-averaged values for each combination of classifiers tested, can aid the understanding of their ordering. The difference is not so much in the false negative values (as might be expected), but in the false positive values. The reason for this is related to the first step of the proposed ensemble algorithm which gives more weight to positive classifications, transferring the focus from false negatives to false positives, and thereby improving the breakeven point [125].

3.3 Conclusion

This chapter discussed and proposed techniques to enhance the classification of text using SVMs. Specifically, the incorporation of unlabeled data and the use of multiple learning machines were investigated.

Two margin-based approaches to introduce unlabeled documents information into the learning stage were presented: background knowledge and active learning. The background knowledge method has the advantage of being completely automated. However, it should not be used with small training sets, i.e. with too weak initial classifiers. When this is not the case it can lead to improvement. The proposed margin-based active learning method has potential to substantially improve performance when small training sets are available. This conclusion is very important in text mining tasks, since usually there are a small number of classified examples and a huge number of unlabeled ones. The enhancements achieved by both methods amount to encouraging results.

Concerning the use of multiple classifiers, an SVM ensemble with a two-step learning strategy using the separating margin as a differentiating factor in positive classifications was proposed. First, the positive classifications given by the base classifiers were enhanced and then the maximum margin classifier was used for each example. This strategy proved to be robust and led to the conclusion that diversity in the base classifier is a more important factor than individual performance.

Both the proposed enhancements to SVMs in text classification integrate new knowledge into the learning procedures and show improvements over the baseline SVMs, presented in the previous chapter. The way the SVM separating margin played a crucial role in the described techniques and how it can be used in further enhancements should be emphasized. Eventually it ough be possible to infer probabilistic information that would approximate SVMs to relevance vector machines. This is discussed in the next chapter.

Chapter 4
Scaling RVMs for Text Classification

Abstract. In the previous chapter we investigated learning techniques to improve support vector machines' (SVMs) performance in text classification. We turn our attention in this chapter to relevance vector machines (RVMs) and their application to text classification. RVMs' probabilistic Bayesian nature allows both predictive distributions on testing instances and model-based selection that yields a parsimonious solution. However, scaling up the algorithm is not viable in most digital information processing applications.

4.1 Introduction

As was noted in Chapter 2, RVMs suffer from poor scaling capabilities when faced with high-dimensional data sets with a large number of examples, like text classification training sets. Figure 4.1[1] depicts a model for the computing time fitted to the experimental data, obtained using the text classification benchmark Reuters-21578, with different training set sizes. Observing this model, one can see the importance of limiting the training set size to a minimum.

Given RVMs' Bayesian roots, all training set documents must be evaluated in the first iteration, and only then are unnecessary documents pruned out. Therefore, unlike SVMs, it is difficult to use decomposition methods that would make the learning faster and computationally less demanding. Hence, efforts to applying RVMs to large scale sets have met with limited success in the past, due to computational constraints. Nevertheless, RVMs have interesting properties, like sparsity and probabilistic outputs, that motivate further investigation to surpass their scalability issues when applied to text classification. Different approaches to tackle these scaling problems are proposed. First, the focus is on techniques that reduce the scale of the problem by selecting the learning instances with active learning and determining the learning examples with a similitude measure. Second, a set of divide-and-conquer

[1] This figure has already been presented in Chapter 2, Figure 2.5, but is again presented to ease the understanding.

C. Silva and B. Ribeiro: Inductive Inference for Large Scale Text Classification, SCI 255, pp. 71–91.
springerlink.com

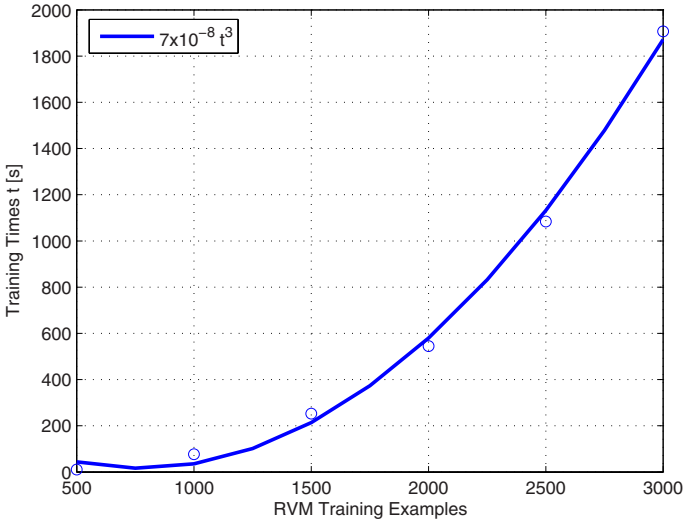

Fig. 4.1 Model of RVMs training times on Reuters-21578 benchmark corpus.

strategies is introduced to cope with the high-dimensional feature space, with incremental, boosting and ensemble RVM methods being put forward. Finally, a hybrid RVM-SVM combination is presented, substantially improving baseline results.

4.2 Scale Reduction Approaches

The straightforward approach for solving scalability problems is to try to reduce the number of examples. Here, it is investigated first how to reduce the number of training documents using an active learning strategy, and second how to determine the most relevant examples with a similitude-based technique.

Several studies have looked at active learning for text classification. Lewis and Gale examine uncertainty sampling and relevance sampling [70]. These pool-based techniques select queries based on only a single classifier instead of a committee. Liere and Tadepalli [76] use committees of Winnow learners for active text learning. They select documents for which two randomly selected committee members disagree on the class label. Nigam and MacCallum show that EM (Expectation-Maximization) with unlabeled data reduces text classification error by one-third [86].

Active learning methods can be grouped according to the active instances selection strategy: committee-based and certainty-based. The first group, Query-By-Committee (QBC) determines the active examples combining the outputs of a set of committee members. In the QBC approach most effort is made to determine samples where members disagree the most as the potentially more informative ones [86]. The second group, on the contrary, tries to determine the most uncertain cases

and indicate them as active ones to be labeled. In certainty-based methods, the certainty measure depends on the learning method used. In [116], Schohn & Cohn suggest a method for active learning, exploiting the samples that are orthogonal to the space spanned by the training set in order provide the classifier with information about dimensions not yet explored. Furthermore, in [148] Tong & Koller introduce a method that considers the examples which better split the current version of the space into equal parts, making them more informative for the model.

4.2.1 Active Learning

A general RVM active learning technique is proposed and tested for text classification. This framework is intended to tackle two major difficulties prevailing in most supervised learning tasks: the overload of data, prohibitive of manual labeling, and the growing complexity of machine learning algorithms, specifically of RVMs.

Optimally, an active learner selects those documents which, when labeled and incorporated into training, minimize classification error over the distribution of future documents. When determining and adding new active elements to the training set, complexity increases in two directions: first within the active search algorithm for new training elements, and second with the size of the training set.

In RVMs, relevance vectors (RVs) RVM active learning and weights define the model (see Section 2.3). RVs can be viewed as cluster centers, i.e. paradigmatic instances of the data (see Figure 2.4). According to this interpretation, the most informative unlabeled examples are potentially those farther away from any of the existing RV of the model, given a measure of distance. To define these examples, we use a kernel approach that defines a design matrix Ψ, assessing the distances between the existing RVs and the set of unlabeled examples available. Given an initial RVM model (2.33), induced using a set of input-output labeled training documents $(\mathbf{d}_1,t_1),...(\mathbf{d}_l,t_l) \in \mathbb{R}^M \times \{\pm 1\}$, resulting in a number r of RVs, $\boldsymbol{\varphi}$, $(\boldsymbol{\varphi}_1,...,\boldsymbol{\varphi}_r) \in \mathbb{R}^M$ and also unlabeled data $\mathbf{U}$, $(\mathbf{u}_1,...,\mathbf{u}_h) \in \mathbb{R}^M$, the distance between an RV, $\boldsymbol{\varphi}_i$, and an unlabeled document, $\mathbf{u}_j$, is defined as

$$\Psi_{ij} = k(\boldsymbol{\varphi}_i, \mathbf{u}_j), \tag{4.1}$$

where k represents the kernel used to define a higher dimension space where points can be compared. For a generic kernel function, Φ, Ψ_{ij} is the dot product

$$\Psi_{ij} = \langle \Phi(\boldsymbol{\varphi}_i), \Phi(\mathbf{u}_j) \rangle. \tag{4.2}$$

Assuming a linear kernel, Ψ_{ij} is simplified

$$\Psi_{ij} = \langle \boldsymbol{\varphi}_i, \mathbf{u}_j \rangle. \tag{4.3}$$

Using a cosine kernel, where the dot product is normalized by the norm

$$\Psi_{ij} = \frac{\langle \boldsymbol{\varphi}_i, \mathbf{u}_j \rangle}{\mid \boldsymbol{\varphi}_i \mid\mid \mathbf{u}_j \mid}, \tag{4.4}$$

the resulting design matrix Ψ is

$$\Psi = \begin{bmatrix} k(\boldsymbol{\varphi}_1, \mathbf{u}_1) & k(\boldsymbol{\varphi}_1, \mathbf{u}_2) & k(\boldsymbol{\varphi}_1, \mathbf{u}_3) & & k(\boldsymbol{\varphi}_1, \mathbf{u}_h) \\ k(\boldsymbol{\varphi}_2, \mathbf{u}_1) & k(\boldsymbol{\varphi}_2, \mathbf{u}_2) & k(\boldsymbol{\varphi}_2, \mathbf{u}_3) & & k(\boldsymbol{\varphi}_2, \mathbf{u}_h) \\ \cdot & \cdot & \cdot & & \cdot \\ \cdot & \cdot & \cdot & \cdots & \cdot \\ \cdot & \cdot & \cdot & & \cdot \\ k(\boldsymbol{\varphi}_r, \mathbf{u}_1) & k(\boldsymbol{\varphi}_r, \mathbf{u}_2) & k(\boldsymbol{\varphi}_r, \mathbf{u}_3) & & k(\boldsymbol{\varphi}_r, \mathbf{u}_h) \end{bmatrix} \tag{4.5}$$

For a linear kernel the design matrix is simplified

$$\Psi_{linear} = \boldsymbol{\varphi} \,.\, \mathbf{U}'. \tag{4.6}$$

And for a cosine kernel the formulation is

$$\Psi_{cosine} = \frac{\varphi \,.\, \mathbf{U}'}{\mid \varphi \mid\mid \mathbf{U} \mid}. \tag{4.7}$$

After the design matrix is constructed, it remains to be determined which unlabeled examples are potentially more informative, i.e. which ones are farther away from any current RV. The procedure is easily implemented as follows. First, the closest RV to any given unlabeled document is determined taking the minimum value of each column of the design matrix

$$\left[\min(k(\boldsymbol{\varphi}_i, \mathbf{u}_1)) \; \min(k(\boldsymbol{\varphi}_i, \mathbf{u}_2)) \; \ldots \; \min(k(\boldsymbol{\varphi}_i, \mathbf{u}_h))\right]. \tag{4.8}$$

Second, L unlabeled examples with larger minimum distance to an RV (the L larger values in (4.8)) are chosen and added to the training set, constituting what will hereunder be termed as the added active learning examples. Although this procedure might incur the choice of some outliers, it still ensures that relevant facets of the underlying data are not neglected.

4.2.1.1 Experimental Results and Analysis

To evaluate the classification performance the macro-averaged F1 values are used, while the number of relevance vectors (RVs) is adequate to analyze solution complexity. For the two tested similitude measures, the complexity is comparable, given the number of RVs in each setting (40.40 and 37.10 average number of RVs). The classification performance is slightly better for the linear kernel, due to its superior recall values.

To further evaluate the active learning framework, it is compared with an active learning scheme based on the random selection of unlabeled examples, i.e. the active examples added to the baseline training set are randomly chosen from the unlabeled set, $\mathbf{U}$. Table 4.3 shows the averaged performance results for the proposed active learning strategy with linear and cosine kernels, for the random active

Table 4.1 Performances for RVM active learning with linear kernel.

Category	RVs	Accuracy	Precision	Recall	F1
Earn	46	95.51%	95.45%	84.91%	89.88%
Acquisitions	59	98.18%	97.15%	98.08%	97.62%
Money-fx	48	96.57%	67.67%	63.83%	65.69%
Grain	32	97.88%	78.36%	78.36%	78.36%
Crude	40	96.54%	72.92%	65.22%	68.85%
Trade	52	96.64%	58.06%	64.29%	61.02%
Interest	44	96.72%	54.46%	61.00%	57.55 %
Ship	32	97.70%	67.74%	49.41%	57.14%
Wheat	22	98.83%	71.79 %	84.85%	77.78%
Corn	29	99.02 %	71.43%	72.92%	72.16%
Average	40.40	97.36%	73.50%	72.29%	72.60%

Table 4.2 Performances for RVM active learning with cosine kernel.

Category	RVs	Accuracy	Precision	Recall	F1
Earn	46	97.74%	96.94%	97.13%	97.03%
Acquisitions	67	95.37%	93.29%	86.47%	89.75%
Money-fx	41	96.64%	73.33%	54.61%	62.60 %
Grain	29	97.81%	84.26%	67.91%	75.21%
Crude	38	96.68%	74.65%	65.84%	69.97%
Trade	43	96.57%	59.57%	50.00%	54.37%
Interest	37	97.19%	63.53%	54.00%	58.38 %
Ship	28	98.10%	90.24%	43.53%	58.73%
Wheat	20	99.12%	82.81 %	80.30%	81.54%
Corn	22	99.27 %	73.68%	73.68%	73.68%
Average	37.10	97.45%	79.23%	67.35%	72.13%

Table 4.3 Summary of RVM active learning averaged results.

			Active Learning					
	Baseline		Random		Cosine Kernel		Linear Kernel	
	F1	RV	F1	RV	F1	RV	F1	RV
500	53.91%	13	60.20%	16	54.21%	15	63.13%	21
1,000	62.40%	19	62.80%	20	63.75%	25	67.14%	26
2,000	70.95%	32	71.57%	34	72.13%	37	72.60%	40

learning scheme and for the baseline RVM for settings using 500, 1,000 and 2,000 documents.

Experimental results with RVM active learning use at most 2,000 documents in the baseline training set, i.e. tests with 500, 1,000 and 2,000 instances were pursued. Settings using more training examples were not attainable in reasonable computational time, since CPU times increase supra-linearly. The remaining training

examples were considered as unlabeled. Both in baseline models and after new active examples were added, the RVMs were trained with linear kernels.

To test the active learning strategy, the design matrix was constructed with linear (4.3) and with cosine (4.4) kernels. The number of active examples added ranged from 200 to 300. Tables 4.1 and 4.2 present the detailed performances for the proposed RVM active learning technique with 2,000 base documents for linear and cosine kernels respectively.

The analysis presented for the 2,000 documents setting is still valid for the other two settings, and the performance, as expected, improves with the size of the training set. The enhancement with the active learning strategy is more significant for smaller baseline training sets, with a 10% average improvement, when compared with larger training sets, where an improvement of around 2% was observed.

Baseline results are generally outperformed by all active learning settings tested, even by the simple random selection, which was predictably found to yield the poorest active learning results.

In fact, random selection of the active examples is very unstable, making it difficult to determine when it would be advantageous. Although average random values are similar to baseline performance, individual results per category exhibit a strong variation, especially the less represented ones. For instance *interest* and *corn* present a reduction of performance for random selection, from 62.63% to 59.18% and 45.40% to 44.97% respectively, despite the expansion of the training set [135]. This undesirable behavior is completely avoided in the proposed kernel-based technique.

Overall, the linear kernel distance metric presents the best performances without penalizing complexity. Cosine kernels most likely introduce a degree of complexity in the distance metric unsuitable for text classification problems that, despite their high dimensionality, are usually linear.

4.2.2 Similitude Measure

Another possibility for scale reduction is to find similarities between documents that allow the reduction of the training set. Here a similitude measure is proposed to determine which documents are considered redundant and can be removed from the training set.

A new approach is provided, which consists of a two-step RVM classifier, able to achieve a competitive processing time, using all available training elements and improving RVM classification performance.

The first stage selects which training documents should be used in the next level. For each pair of documents $\{\mathbf{d}_i, \mathbf{d}_j\}$ a similitude measure S_{ij}, based on the co-occurrence of words, is calculated using the following method.

S_{ij}=0;
for each word
 if word is not in both documents $\{\mathbf{d}_i, \mathbf{d}_j\}$
  $S_{ij} = S_{ij} + A$;

```
else
if word is in only one of the documents {d_i, d_j}
   S_ij = S_ij − B;
else S_ij = S_ij + C;
```

This similitude measure captures the similarities and dissimilarities between documents, using the co-occurrence patterns of words. When a word either occurs or does not occur simultaneously in both documents, the similitude is strengthened. On the other hand, when a given word occurs in only one of the documents, the similitude is weakened. `A` and `C` reflect similitude between the documents, due to co-absence and co-occurrence, respectively. `B` reflects the occurrence of a word in only one of the documents, i.e. a difference that diminishes the similitude measure. Empirically and intuitively `A` was set to 1, `B` was set to 5 and `C` was set to 10. The intrinsic idea is that the most important similitude is found when a term appears in both documents (`C` $= 10$). The fact that a term does not appear in neither document is positive, but not as relevant (`A` $= 1$) and when a term occurs in only one of the documents lies somewhere in-between (`B` $= 5$).

4.2.2.1 Experimental Results and Analysis

To make this method computationally effective, it was not applied just once to the whole collection, but to chunks of 500 documents. This procedure not only makes the process swifter, but also parallelizable, making the classification system scalable.

The result of processing each chunk is thus a set of pairs of similar documents (considered similar when above an empirically defined threshold of $S_{ij} >$`100`), after removing the similar redundant documents. If two documents $\mathbf{d}_i$ and $\mathbf{d}_j$ are similar, the document with fewer words is removed. A further protection is imposed on not removed documents: they may not be removed later, since they were already responsible for the elimination of one document. The second step of the method gathers all the remaining documents from all the chunks of 500 documents and infers an RVM classifier. Table 4.4 summarizes the results obtained for this two-step RVM classifier.

For each category it includes the number of documents used, the resulting number of RVs and the classification performance metrics. Comparing these results with the baseline RVM in Table 2.2 we can observe the improvement from 70.95% to 74.57% in F1 values, while the sparsity is not heavily penalized (32.10 to 37.60 average RVs), despite the much larger training set used with the proposed two-step RVM approach.

Figure 4.2 shows the ROC curves for the baseline RVM trained with 2,000 documents and for the RVM trained with the proposed two-step method, both for the Reuters-21578 trade category. It substantiates the better performance of the proposed RVM classifier over the baseline RVM, since the baseline curve is always under the two-step RVM approach curve. In addition, the AUC (Area Under the

Table 4.4 Performances for RVM using a similitude measure to select training documents.

Category	Documents	RVs	Accuracy	Precision	Recall	F1
Earn	2724	48	96.02%	97.66%	91.76%	94.62%
Acquisitions	2804	45	94.71%	93.68%	83.05%	88.05%
Money-fx	2762	52	95.84%	58.18%	68.09%	62.75%
Grain	2809	34	98.25%	85.25%	77.61%	81.25%
Crude	2804	42	97.05%	73.53%	77.64%	75.53%
Trade	2801	35	97.05%	63.25%	66.07%	64.63%
Interest	2846	38	97.23%	64.29%	54.00%	58.70 %
Ship	2855	39	98.07%	72.86%	60.00%	65.81%
Wheat	2793	15	99.27%	81.08 %	90.91%	85.71%
Corn	2791	28	98.87 %	66.67%	70.83%	68.69%
Average	2798.90	37.60	97.24%	75.64%	74.00%	74.57%

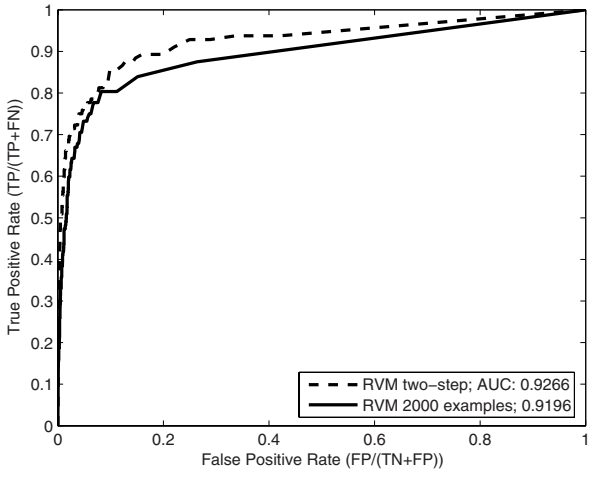

Fig. 4.2 ROC curves for RVMs with similitude measure for trade category.

ROC Curve) values also confirm this result, exhibiting an improvement from 0.9196 to 0.9266.

4.3 Divide-and-Conquer Approaches

Here three divide-and-conquer methods to scale up RVMs to high-dimensional text data sets are investigated (see Figure 4.3). Results of RVMs obtained using the limited number of examples computationally feasible are already competitive, but there is potential to improve the performance if more training examples can be used.

The proposed set of divide-and-conquer approaches employs decomposition techniques to promote the definition of smaller working sets that permit the use

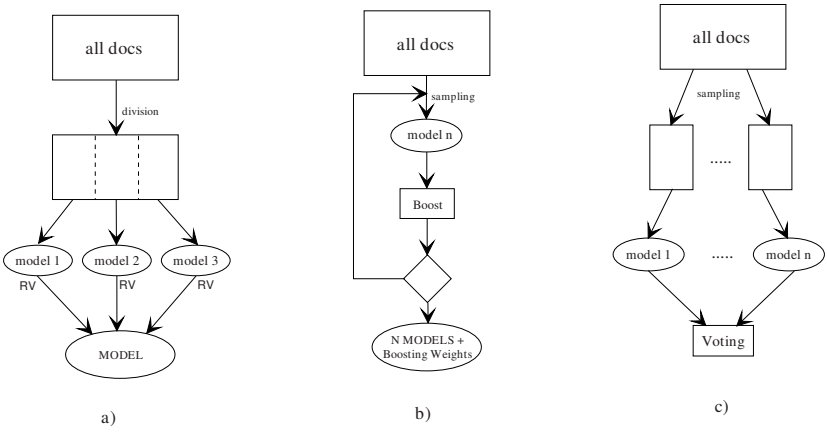

Fig. 4.3 a) Incremental RVM; b) RVM Boosting ; c) RVM Ensemble.

of all training examples. The rationale is that, by exploring incremental, ensemble and boosting strategies, it is possible to improve classification performance taking advantage of the large training set available. Figure 4.3 is a schematic representation of the three methods detailed in the next sections.

4.3.1 Incremental RVM

The first method is an incremental approach to scale RVMs, where the final training set is incrementally constructed. The dataset is divided into evenly sized smaller subsets (or chunks). Each chunk is then trained independently, possibly in parallel, resulting in a set of RVM models. The relevance vectors yielded by each model are gathered and constitute the training set of a new RVM model, which will be the final incremental RVM.

Figure 4.3 a) illustrates the procedure of the proposed incremental RVM technique. The size of each chunk and the number of chunks should be determined according to the available computational power. Note that, given the independence of the initial models, if there is a distributed platform with available resources, the procedure can be speeded up. Moreover, this method essentially scales linearly with the number of chunks, thus taking advantage of the entire training set.

Table 4.5 shows the incremental RVM results for chunks of 1,000 and 2,000 documents. Larger chunks were not considered since their computational burden would jeopardize algorithm scalability.

Comparing the incremental RVM average results with the average baseline (see Table 2.2), there is a classification performance improvement of between 3% and 5%, due to fewer false negatives, causing an improvement in recall values [131, 137]. Note that, in text classification applications, recall values, associated with false negatives, are extremely important because of the small number of positive examples.

Table 4.5 Performances for incremental RVM learning.

Chunks of 1,000 documents

Category	RVs	Accuracy	Precision	Recall	F1
Earn	33	95.08%	95.04%	91.86%	93.42%
Acquisitions	43	94.31%	88.35%	87.25%	87.79%
Money-fx	27	95.99%	59.75%	67.38%	63.33%
Grain	37	97.59%	73.94%	78.36%	76.09%
Crude	23	95.81%	68.55%	52.80%	59.65%
Trade	20	96.83%	60.68%	63.39%	62.01%
Interest	31	95.99%	47.02%	79.00%	58.96 %
Ship	25	96.61%	46.15%	56.47%	50.79%
Wheat	22	98.03%	56.25 %	81.82%	66.67%
Corn	24	98.40 %	53.33%	66.67%	59.26%
Average	28.50	96.46%	64.91%	72.50%	67.80%

Chunks of 2,000 documents

Category	RVs	Accuracy	Precision	Recall	F1
Earn	41	97.48%	95.01%	98.56%	96.76%
Acquisitions	55	95.92%	92.48%	89.89%	91.17%
Money-fx	39	96.46%	62.09%	80.14%	69.97%
Grain	35	97.59%	73.61%	79.10%	76.26%
Crude	38	96.46%	69.51%	70.81%	70.15%
Trade	44	96.75%	57.72%	76.79%	65.90%
Interest	31	97.19%	64.94%	50.00%	56.50 %
Ship	40	97.99%	66.30%	71.76%	68.93%
Wheat	15	98.87%	74.65 %	80.30%	77.37%
Corn	28	98.58 %	57.89%	68.75%	62.86%
Average	36.60	97.33%	71.42%	76.61%	73.59%

Concerning solution complexity, the incremental approaches have slightly more RVs, i.e. there is a difference of 10 and 4 RVs for the 1,000 and 2,000 baseline training documents settings respectively.

4.3.2 RVM Boosting

This section presents a boosting method applied to RVMs and text classification. The main idea of boosting is to generate several relatively weak classification rules and to combine them into a single highly accurate classification rule [113]. First, the standard algorithm is introduced, and afterwards the proposed RVM boosting method.

The boosting algorithm assigns different importance weights to different training examples. The algorithm proceeds by incrementally increasing the significance of the examples which are hard to classify, while easier training examples get lower weights. This weighting strategy is the basis of the weak learners evaluation. The

Algorithm 2. AdaBoost algorithm.

Input:
N training labeled examples $< (\mathbf{d}_1, t_1), ..., (\mathbf{d}_N, t_N) >$, $t_i \in \{-1, +1\}$
integer T specifying the number of iterations

Initialize $X_1(i) = \frac{1}{N}$

for s=1,2,...,T **do**
Call weak learner and get weak hypothesis h_s
Calculate the error of h_s: $\varepsilon_s = \sum_{i:h_s(\mathbf{d}_i) \neq t_i} X_s(i)$
Set $\delta_s = \frac{1}{2} ln\left(\frac{1-\varepsilon_s}{\varepsilon_s}\right)$
Update distribution:

$$X_{s+1}(i) = \frac{X_s(i)\ e^{-\delta_s t_i h_s(x_i)}}{Z_s}$$
$$= \frac{X_s(i)}{Z_s} \times e^{-\delta_s}, \text{ if } h_s(\mathbf{d}_i) = t_i$$
$$= \frac{X_s(i)}{Z_s} \times e^{\delta_s}, \text{ if } h_s(\mathbf{d}_i) \neq t_i$$

where Z_s is a normalization factor.

end for

Output: the final hypothesis:
$h_{fin}(\mathbf{d}) = sign\left(\sum_{s=1}^{T} \delta_s h_s(\mathbf{d})\right)$.

final combined hypothesis classifies a new testing example by computing the prediction of each of the weak hypothesis and taking a vote on these predictions. Algorithm 2 presents the AdaBoost Algorithm [38].

The algorithm starts with N input-target pairs $< (\mathbf{d}_1, t_1), ..., (\mathbf{d}_N, t_N) >$, where $\mathbf{d}_i$ is a training example and $t_i \in \{-1, +1\}$ is the associated label, usually defined by a human expert. Initially the importance weights of the examples are uniformly distributed ($X_1(i) = \frac{1}{N}$). Then, the AdaBoost algorithm repeatedly retrieves *weak hypotheses* that are evaluated and used to determine the final hypothesis. On iteration s, using the set of importance weights determined on iteration $s-1$, the hypothesis error is computed and δ, which corresponds to the weight or importance of that *weak classifier*, is determined. The expression for δ assigns larger weights to *good classifiers*, i.e., classifiers with low error, whereas lower weights (even with negative values) are assigned to *bad classifiers*. Although RVMs may not be considered *weak learners*, we show that the boosting concept can be applied to avoid RVM scaling problems. Figure 4.3 b) illustrates the innovations performed on the AdaBoost procedure and Algorithm 3 shows the changes made to obtain the RVM boosting algorithm.

The main idea in the RVM boosting approach is to use all the training examples, by sampling them into small working sets, making each classifier much weaker than it would be if trained with all available training examples. If enough models are generated, all distinctive aspects of the class can be captured and represented in the final classifier. By dividing the huge data set into smaller tractable chunks, the computational load usually associated with training RVMs is mitigated. Moreover,

Algorithm 3. RVM boosting algorithm.

Input:
N training labeled examples $< (\mathbf{d}_1, t_1), ..., (\mathbf{d}_N, t_N) >$, $t_i \in \{-1, +1\}$
N_{boost} boosting labeled examples:
$< (\mathbf{d}_{N+1}, t_{N+1}), ..., (\mathbf{d}_{N+N_{boost}}, t_{N+N_{boost}}) >$, where $t_i \in \{-1, +1\}$
integers NC and T specifying the number of classifiers and iterations

Initialize $X_1(i) = \frac{1}{N_{boost}}$

for $s = 1, 2, ..., T$ **do**

$c = s$ **mod** NC

if $c = 0$ **then**

c=NC

end if

Call weak learner and get weak hypothesis h_c

Calculate the error of h_s: $\varepsilon_s = \sum_{i:h_c(\mathbf{d}_i) \neq t_i} X_s(i)$

Set $\delta_s = \frac{1}{2} ln \left(\frac{1-\varepsilon_s}{\varepsilon_s} \right)$

Update distribution:

$X_{s+1}(i) = \frac{X_s(i)\ e^{-\delta_s t_i h_s(x_i)}}{Z_s}$

$= \frac{X_s(i)}{Z_s} \times e^{-\delta_s}$, if $h_c(\mathbf{d}_i) = t_i$

$= \frac{X_s(i)}{Z_s} \times e^{\delta_s}$, if $h_c(\mathbf{d}_i) \neq t_i$

where Z_s is a normalization factor.

end for

Output: the final hypothesis:
$h_{fin}(\mathbf{d}) = sign \left(\sum_{s=1}^{T} \delta_s h_c(\mathbf{d}) \right)$.

due to the independence of the *not so weak* RVM classifiers, it is possible to distribute the computational burden within a cluster or other distributed environment.

First, instead of using the training set for training and for boosting, a separate boosting set was defined. The training set is used to learn the base RVM models, while the boosting set is used to define the weights of the classifiers and the documents.

Second, considering that the RVM classifiers are in fact *not so weak classifiers*, as the AdaBoost assumes, the same set of classifiers was presented repeatedly to the boosting algorithm, i.e. the number of iterations is not equal to the number of classifiers, but it is proportional to it. This way for each model the boosting algorithm was run several times updating the weights iteratively. Considering different training and boosting sets is justifiable in large scale problems, since there are enough examples. Also, when the training sets are large and sparse, convergence problems may occur by boosting the classifier with the same set. These convergence problems can lead to the algorithm being unable to determine which are the harder examples, i.e. those

having greater weight in the classifier evaluation, given that using the same learning and boosting sets may result in insufficient diversity.

To test the RVM Boosting approach, 20 classifiers were trained by randomly sampling 2,000 documents from the training set. For each classifier, the rest of the training set was used for boosting. Tests were performed with 20, 40 and 60 iterations, i.e. each of the 20 classifiers was used to update the weights 1, 2 or 3 times (more runs were tried with no significant improvement). Table 4.6 shows the F1 performance results for the three settings.

Table 4.6 F1 performance for RVM boosting learning with 20 classifiers of 2,000 examples.

Category	20 iterations	40 iterations	60 iterations
Earn	97.40%	97.26%	97.35%
Acquisitions	92.05%	91.95%	91.87%
Money-fx	66.84%	64.39%	64.39%
Grain	83.27%	83.27%	83.27%
Crude	73.87%	75.00%	75.00%
Trade	66.03%	66.99%	66.99%
Interest	65.14%	65.91%	65.91%
Ship	70.75%	70.75%	70.75%
Wheat	85.08%	84.81 %	84.21%
Corn	71.26 %	71.26%	71.26%
Average	76.77%	76.99%	77.39%

Comparing RVM boosting results with the baseline RVM performance (see Table 2.2), we observe an improvement of 7% [142]. Comparing the three proposed settings, there is a small but consistent improvement when each classifier is repeatedly presented to the RVM boosting algorithm.

4.3.3 RVM Ensemble

When a scalability problem exists, as with RVMs, ensemble strategies should also be considered as possible solutions. An ensemble can solve a number of learning problems, particularly parameter tuning, testing set definition and cross-validation. The idea of combining multiple classifiers is based on the observation that achieving overall optimal performance is not necessarily consistent with obtaining the best performance for an individual (base) classifier. The rationale is that it might be easier to optimize the design of a combination of relatively simple classifiers than to optimize the design of a single complex classifier.

The RVM ensemble strategy depicted in Figure 4.3 c) starts by constructing several smaller evenly sized training sets, randomly sampled from the entire available training set. The size and number of the training sets depend on the available computational power, but more training examples usually result in more diversity and better performance. Then, a model is learned from each training set. These models will constitute the ensemble individual classifiers. After this learning phase, a

majority voting scheme is implemented to determine the ensemble output decision, taking as output value the average value of the classifiers that corroborated the majority decision.

To define the ensemble, 40 classifiers were trained by randomly choosing 2,000 examples from the training set. Table 4.7 presents, for each category of Reuters-21578, the best performing element of the ensemble (Maximum), the mean of the 40 elements (Mean) and finally the result of the ensemble majority voting scheme (Ensemble). For instance, for the earn category the best of the 40 models achieved an F1 of 96.77%, the F1 average of the 40 models was 96.12% and the ensemble majority voting scheme resulted in a classification performance of 97.69%. Analyzing these values, it can be seen that the ensemble strategy outperforms the average result as expected, and also the best of the individual base classifiers. Comparing the baseline RVM performance (see Table 2.2), an improvement of around 9% is observed [132, 136].

Table 4.7 F1 performance for RVM ensemble learning with 40 classifiers of 2,000 examples.

Category	Maximum	Average	Ensemble
Earn	96.77%	96.12%	97.69%
Acquisitions	90.36%	88.84%	94.97%
Money-fx	68.29%	60.54%	71.81%
Grain	84.83%	79.16%	81.62%
Crude	75.08%	70.13%	78.34%
Trade	66.67%	62.46%	70.25%
Interest	67.02%	62.10%	70.47%
Ship	66.67%	60.43%	77.99%
Wheat	85.71%	80.83 %	81.48%
Corn	70.83 %	64.06%	66.67%
Average	77.22%	72.46%	79.13%

4.3.4 Analysis of Results

In this section three divide-and-conquer strategies proposed for scaling RVMs are analyzed and compared (see Figure 4.3) - the incremental, boosting and ensemble approaches. The three methods are distinguished mostly by the way they combine the chosen training chunks. After the detailed results presented in the previous sections, Table 4.8 summarizes the macro-averaged F1 results of the three proposed RVM scaling methods together with baseline results for the Reuters-21578 benchmark dataset. The trend of increasing performance is: Baseline, Incremental, Boosting, Ensemble.

Figures 4.4 and 4.5 present ROC curves for the acquisitions and trade categories from Reuters-21578, including the area under the curve (legend in the bottom right corner), depicted both for the baseline results and for the three methods proposed. The ROC curves are generally consistent with F1 results, showing that the proposed

Table 4.8 F1 performance for divide-and-conquer RVM approaches.

Baseline	Incremental	Boosting	Ensemble
70.95%	73.93%	77.30%	79.13%

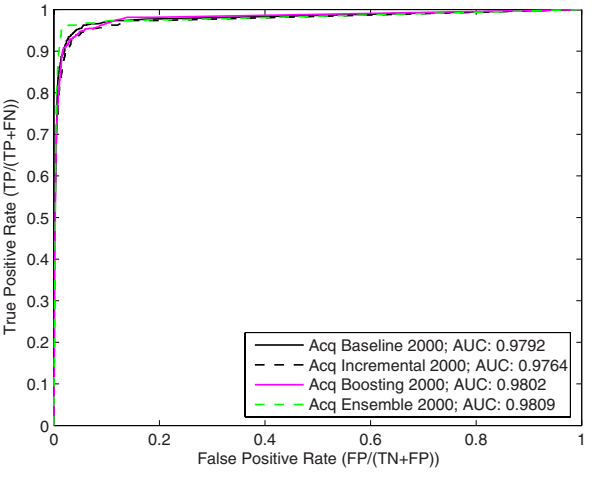

Fig. 4.4 ROC curves for RVMs with divide-and-conquer strategies for acquisitions (Acq) category.

techniques, especially boosting and ensemble, improve classification performance compared with baseline results.

The incremental approach, using RVs gathered from models trained with subsets of the training set, shows the first evidence that using the entire training information can be useful, improving the average performance by 2% to 3%. This average improvement is small, so in some specific situations, it may not improve, owing to the nature of this learning machines. RVMs try to discover typical, paradigmatic instances in the whole dataset, and when using subsets, the final relevance vectors may not always be consistent with the training set. The boosting approach, that ranks the classifiers according to their performance on harder-to-classify examples, allows a 6% to 7% surplus, constituting an important improvement. The ensemble approach presents the higher performance improvement (9% to 12%), by using majority voting. This result strengthens the *no free lunch theorem* [161], corroborating that there is no individual better classifier, but the effort should be put into combining different models to achieve a better generalization capability.

The proposed incremental approach presents the poorest results of the three strategies, most likely because it ignores the dependence on the whole scenario of data exhibited by the relevance vectors. Comparing boosting and ensemble strategies, a considerable difference is the use of the probabilistic RVMs' output. While

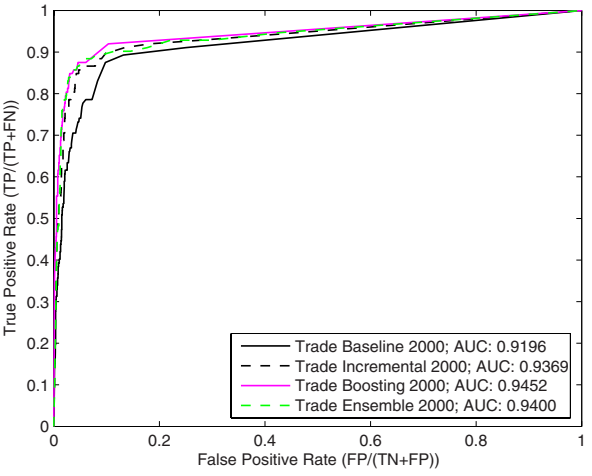

Fig. 4.5 ROC curves for RVMs with divide-and-conquer strategies for trade category.

boosting uses the original method for ranking the classifiers, the ensemble approach takes as output value the average value of the classifiers that corroborated the majority decision. This constitutes a leading edge that allows the ensemble to achieve a slight improvement over boosting.

4.4 Hybrid RVM-SVM Approach

Support Vector Machines (SVMs) and Relevance Vector Machines (RVMs) are two state-of-the-art learning machines that are currently the focus of cutting-edge research. SVMs offer better accuracy and complexity, but are surpassed by RVMs when it comes to probabilistic outputs or kernel selection.

In the following a two-level hierarchical hybrid SVM-RVM model is proposed to combine the best of both learning machines. The first level of the proposed model uses an RVM to determine the less confident classified examples and the second level uses an SVM to learn and classify the tougher examples. The hierarchical approach outperforms both baseline learning machines. Figure 4.6 depicts the proposed two-level hierarchical hybrid model in the training and in the testing settings.

The first level of the two-level hierarchical approach uses the probabilistic nature of several RVM models to determine which examples will be retained and which will be transferred to the second level. To cope with RVM scalability problems, the training set was divided into chunks of 1,000 examples and an RVM model was induced for each chunk. By interpreting the RVM probabilistic output classification as a confidence measure, we defined a symmetrical interval centered on the origin ($[-range, range]$), where the RVM presented lower confidence. This interval

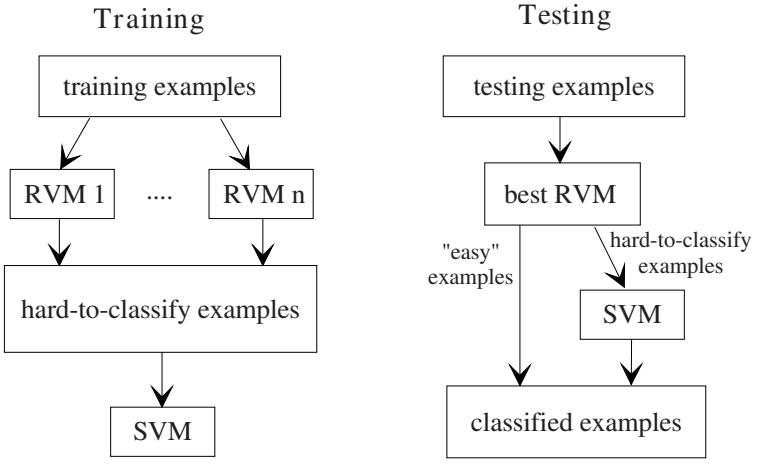

Fig. 4.6 Two-level hierarchical RVM-SVM hybrid model.

or range was inspired by the logistic function, $p(t=1|\mathbf{d}) = \frac{1}{1+exp(-f_{RVM}(\mathbf{d}))}$, where $\mathbf{d}$ represents the document, t is the target and f_{RVM} is the RVM classifier function. Thus, training examples that fall in the area under the logistic curve are selected to be labeled as hard-to-classify examples. These examples classified within the defined range will constitute the training set of the second level SVM, which has presented the best standalone results so far. Testing procedure is similar to the training phase. Table 4.9 presents the performances for the hybrid RVM-SVM using different confidence intervals.

The confidence intervals were determined according to the logistic function and range from $[-5,5]$ to $[-15,15]$ (other values with poorer performances were also tested). A testing example is first classified by the RVM model that exhibited most confidence in the training phase, i.e., with fewer examples being transferred to the second level. If it is classified outside the interval $[-range, range]$ the testing phase ends, otherwise the second level SVM will be responsible for its classification. As the range value increases, less examples will be classified by the first level RVM, meanwhile more examples will be classified by the second level SVM.

Figure 4.7 allows a graphical comparison of the F1 performances obtained for the tested range intervals independently for the first (RVM) and second (SVM) levels. For small ranges, the second level SVM performs better than the first level RVM. However, as the range increases, positions are changed and the first level RVM becomes superior. As the range value increases, fewer and simpler examples will be classified by the first level RVM, meanwhile more and harder examples will be classified by the second level SVM. Thus, the first and second levels cannot be directly compared, since each only provides partial classifications and only their merging will provide complete classification performance.

Figure 4.8 summarizes all the results achieved with the two-level hierarchical model. It shows the average F1 performances for different range values for the

Table 4.9 Performances for hybrid RVM-SVM with different ranges.

	Range=5			Range=7		
Category	1^{st} level	2^{nd} level	Total	1^{st} level	2^{nd} level	Total
Earn	98.30%	99.14%	98.43%	98.65%	91.01%	97.41%
Acquisitions	94.01%	87.82%	92.48%	94.85%	92.07%	93.86%
Money-fx	42.22%	84.21%	49.54%	46.67%	74.75%	68.22%
Grain	77.78%	83.76%	80.66%	84.97%	80.00%	83.00%
Crude	47.31%	84.95%	72.40%	83.72%	76.84%	79.74%
Trade	62.96%	79.75%	75.58%	71.79%	74.03%	72.64%
Interest	69.05%	76.00%	72.83%	70.97%	77.86%	75.65%
Ship	49.06%	82.35%	69.57%	57.14%	78.90%	73.61%
Wheat	79.37%	80.00 %	79.66%	78.57%	77.42%	77.97%
Corn	73.68 %	68.97%	72.09%	68.57%	76.92%	73.56%
Average	69.37%	82.69%	76.32%	75.59%	79.97%	79.66%

	Range=10			Range=15		
Category	1^{st} level	2^{nd} level	Total	1^{st} level	2^{nd} level	Total
Earn	99.03%	94.47%	97.82%	99.26%	96.22%	97.77%
Acquisitions	95.82%	93.01%	94.38%	97.52%	94.49%	95.34%
Money-fx	80.46%	73.27%	75.43%	50.00%	75.18%	74.47%
Grain	87.88%	82.35%	85.26%	91.67%	83.87%	86.85%
Crude	77.55%	81.68%	81.03%	95.38%	78.40%	81.90%
Trade	66.67%	74.37%	73.73%	87.50%	74.64%	75.56%
Interest	86.15%	70.77%	75.90%	91.89%	70.87%	75.00%
Ship	81.25%	76.92%	77.85%	70.59%	81.20%	80.00%
Wheat	83.72%	81.01 %	81.97%	80.00%	80.77%	80.65%
Corn	83.33 %	73.47%	77.65%	82.35%	81.16%	81.40%
Average	84.19%	80.13%	82.10%	84.62%	81.69%	82.89%

two-level hierarchical hybrid SVM-RVM classification model, together with the F1 values (constant with respect to the range) for RVM and SVM baseline classifiers. F1 average values show an improvement over baseline SVMs and RVMs, made possible by the hierarchical association of both learning machines. While the best baseline results, achieved by SVMs, were 78.99% the best hierarchical result is 82.89%, achieved with the range $[-15, 15]$.

The larger the range considered, the larger the percentage of testing examples classified by the second level. Four range settings were tested, covering the possible splits of testing examples between the two levels. The percentages of testing examples classified by the second level SVM were 8%, 18%, 46% and 73% for ranges $[-5, 5]$, $[-7, 7]$, $[-10, 10]$ and $[-15, 15]$ respectively. As the number of examples in the second level grows, SVM performance degrades, since more examples including the more difficult-to-classify are assigned to it [134].

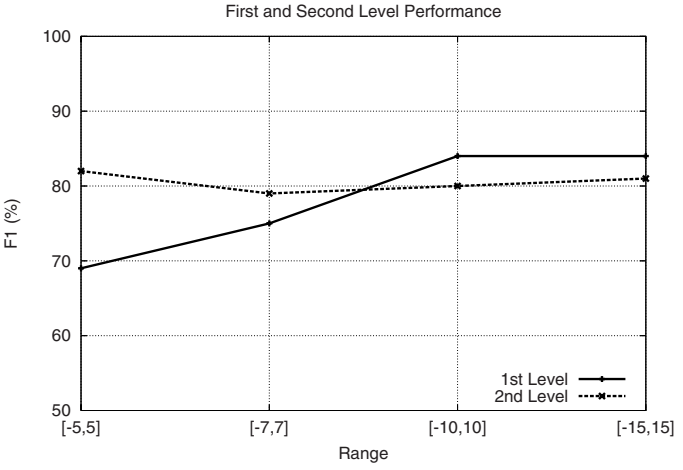

Fig. 4.7 F1 performance comparison of hybrid RVM-SVM first and second levels.

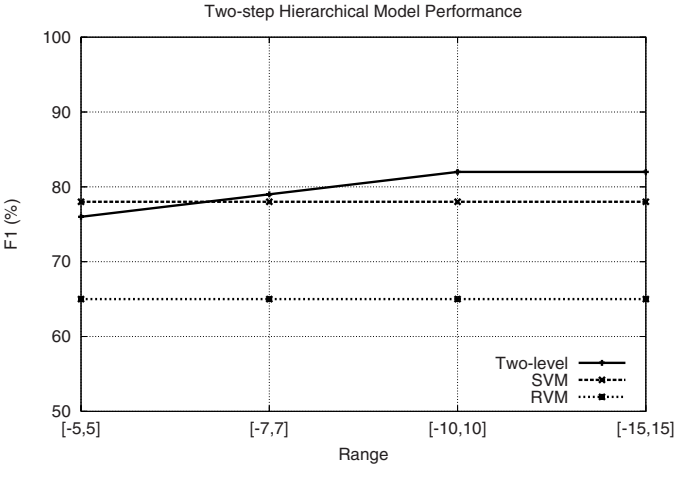

Fig. 4.8 F1 performance comparison of hybrid RVM-SVM with two-level hierarchical model and SVM and RVM baselines.

4.5 Conclusion

This chapter has introduced several techniques to scale RVMs to applications with large data sets, such as text classification. First, ways to reduce the size of the problem were investigated, viz. active learning and similitude measure between terms. Then the focus was on three divide-and-conquer methods, viz. incremental, boosting and ensemble strategies. Finally, a hybrid RVM-SVM approach was proposed to deal with the scale of text classification problems.

Concerning active learning, we introduced an active learning RVM method based on the kernel trick. The underlying idea is to define a working space between the relevance vectors (RVs) initially obtained in a small labeled data set and the new unlabeled examples, where the most informative instances are chosen. Using a kernel distance metric, a higher-dimensional space can be defined where the selection can take place. The proposed active learning method not only surmounts the problem of overload of unlabeled examples available in learning tasks like text classification, but also overcomes the scalability problems posed by RVM learning machines and selects an optimized working set of training documents. Improvements of 2% to 10% over baseline performance were achieved, without severely affecting the size of the problem. Complexity escalation was controlled, since the number of added documents was fixed and the best method of the experiments, the linear kernel, provides a simple strategy to determine those active documents.

In this chapter methods to reduce the number of documents in the training set to cope with the large scale of the problem were also studied. A two-step RVM that is able to manage large datasets in the setting of text classification was presented. The first stage selects which training documents go to the next level, using a similitude measure between documents, based on the co-occurrence of words. The second step of the method gathers all remaining documents from all the chunks of documents and infers an RVM classifier. The approach tends to maintain the sparse solutions given by RVMs, while improving classification performance by around 4%, with some penalization on training time. However, the number of RVs is kept remarkably small, making the recall phase much faster than with SVMs. These values show that RVMs can show competitive accuracy while maintaining their capacity for sparseness, as long as training examples are carefully established.

Concerning divide-and-conquer approaches, in this chapter a set of methods was proposed, where decomposition techniques promote the definition of smaller working sets that permit the use of all training examples in RVM expansion to large datasets. It was demonstrated that, by exploring incremental, ensemble and boosting strategies, it is possible to make use of RVMs' advantages, such as predictive distributions for testing instances and sparse solutions, while maintaining and even improving the classification performance. The resulting models allow the use of information from the entire training set, yielding significant improvement (9%) over baseline RVM performance. The outlined methods rely on a selection of small working chunks from the training set and then explore different combining strategies. By dividing the huge data set into smaller tractable chunks, the computational load usually associated with training RVMs is mitigated. Moreover, due to the independence of the RVM classifiers, it is possible to distribute the computational burden within a cluster or other distributed environment, as will be addressed in the following chapter.

Finally, a two-level hierarchical hybrid SVM-RVM model was exploited to combine the best of both SVMs and RVMs. The first level exploits RVMs' probabilistic nature to define the second-level training set, where SVMs' accuracy properties are utilized. Experimental results show that the proposed model has the potential to

outperform existing approaches, presenting improvements of around 12% over RVMs and around 3% over SVMs.

This chapter has demonstrated the potential of RVMs in text classification tasks, when appropriate scaling strategies are put forward. The combination of RVMs and SVMs was shown to be extremely beneficial to text classification performances.

Chapter 5
Distributing Text Classification in Grid Environments

Abstract. The previous chapters looked at several ways to improve the performance of support vector machines (SVMs) and relevance vector machines (RVMs) in text classification applications.

Most data mining problems are nowadays faced with two great challenges. First, the volume of digital data available is growing massively in almost all application areas. Second, state-of-the-art learning machines are becoming increasingly demanding in terms of computing power. This chapter establishes a high-performance distributed computing environment model where the learning techniques proposed in the previous chapters are efficiently deployed and tested in large scale corpora.

5.1 Introduction

Nowadays we have to deal with an overwhelming amount of data. New communication scenarios, notably the Internet, have appeared and deliver large units of textual information, which must be properly managed. The many scientific and industrial fields generate enormous amounts of text data, such as news wires, microarray gene data or web pages. On the other hand, centralized solutions to machine learning and data mining problems are not suitable for many current enterprises, considering that data sources are often distributed and the complexity of tasks and techniques is increasing. In fact, an important challenge for cutting-edge research in machine learning is integrating knowledge and learning, i.e. knowledge discovery has moved on to the incorporation of knowledge into the learning process. The high complexity of such methods [36] and the high dimensionality of text and its representation have generated interest in distributing text classification. The challenge, therefore, is to tackle the synergies made possible by linking content, knowledge and learning so as to make content and knowledge abundant, accessible, interactive and usable over time [35].

Efforts made towards distributing several tasks in cluster and grid environments have met with some success, as is shown in Section 5.2. These applications have greatly benefited from the availability of inactive computational resources over long

C. Silva and B. Ribeiro: Inductive Inference for Large Scale Text Classification, SCI 255, pp. 93–115.
springerlink.com

periods of time, not only on the Internet, but more especially in educational institutions and companies.

In this chapter the deployment of text classification tasks on two distributed platforms in a cluster environment is proposed, using code developed for a sequential implementation [140, 80]. A direct acyclic graph (DAG) is used to define the tasks and dependencies, and a model is built to describe the task executions and determine the graph optimizations. The first step is speeding up text classification by distribution and, then more complex and demanding knowledge-integrating learning techniques, such as Bayesian methods and ensemble approaches are pursued. Testing is carried out on the standard Reuters-21578 corpus (see Section 1.6.2.1 and Appendix A) and on the 35 times larger RCV1 corpus (see Section 1.6.2.2 and Appendix B).

5.2 Related Work

Many earlier studies have looked at distributed approaches to data mining applications. Most of them require a distributed computing platform which supports the deployment of data mining procedures. Regrettably, often such approaches are application specific and not easy to compare directly.

5.2.1 Distributed Computing Platforms

TeraGrid[1] was one of the first projects in the area of grid-based knowledge discovery. It is an open scientific discovery infrastructure combining leadership-class resources at eleven sites in the United States of America to create an integrated, lasting computing resource. Using high-performance network connections, the TeraGrid integrates high-performance computers, data resources and tools, and high-end experimental facilities. TeraGrid resources currently include more than 750 teraflops of computing capability and more than 30 petabytes of online and archival data storage, with rapid access and retrieval over high-performance networks.

The Algorithm Development and Mining System (ADaM)[2] was developed by the Information Technology and Systems Center at the University of Alabama in Huntsville. It is used to apply data mining technologies to remotely-sensed and other scientific data. The mining and image processing toolkits consist of interoperable components that can be linked together in a variety of ways for application to diverse problem domains. ADaM has over 100 components that can be configured to create customized mining processes. Pre-processing and analysis utilities help users to apply data mining to their specific problems. New components can easily be added to adapt the system to different science problems. ADaM is the first data mining platform to execute on the NASA (National Aeronautics and Space Administration) Information Power Grid.

[1] http://www.teragrid.org/

[2] http://datamining.itsc.uah.edu/adam/

NaCTeM[3] (The National Centre for Text Mining) is the first publicly-funded text mining centre in the world. They provide text mining services in response to the requirements of the United Kingdom academic community. The goal of this project is to investigate needs and to develop an infrastructure that will enable various text mining applications to work in the grid environment. This topic includes research into the roles of text mining for the Semantic Web and the Semantic grid, and vice versa [111].

SETI@home[4] is a scientific experiment that uses Internet-connected computers in the search for extraterrestrial intelligence (SETI). Anyone can participate by running a free program that downloads and analyzes radio telescope data.

The Globus Toolkit[5] is an open source software toolkit used for building computational grids and grid-based applications, letting people share computing power, databases, and other tools securely online across corporate, institutional, and geographic boundaries without sacrificing local autonomy. The toolkit includes software services and libraries for resource monitoring, discovery, and management, plus security and file management. The Globus Toolkit has grown through an open-source strategy similar to the Linux operating systems, and distinct from proprietary attempts at resource-sharing software. This encourages broader, more rapid adoption and leads to greater technical innovation, as the open-source community provides continual enhancements to the product.

Two freely available general purpose middleware platforms for grid computing are tested: Condor[6] and Alchemi[7]. The goal of the Condor project is to develop, implement, deploy, and evaluate mechanisms and policies that support High Throughput Computing (HTC) on large collections of distributed computing resources. Condor consists of a set of software tools that enable engineers to increase their computing throughput. It is supported by many platforms, including Linux and Windows XP. Condor's task is to automatically schedule and deploy jobs specified by users and summarize the results. The applications deployed through Condor should have a console (command line) interface.

Alchemi, on the other hand, consists of a set of libraries and tools enabling grid computing on the Microsoft .NET platform. It supports a traditional approach to scheduling jobs (like Condor) and a more native approach to the Microsoft .NET platform, enabling multi-threaded programming to be extended from one computer to the computing grid. An advantage of Alchemi is native support for graphical user interface applications.

5.2.2 Distributed Applications

Efforts at distributing several tasks in cluster and grid environments have met with some success. In [2] a fire prediction problem is tackled, accelerating and enhancing

[3] http://www.nactem.ac.uk/
[4] http://setiathome.berkeley.edu/
[5] http://www.globus.org/toolkit
[6] http://www.cs.wisc.edu/condor
[7] http://www.alchemi.net

predictions. In [61] a parallel evolutionary algorithm applied in an Alchemi framework optimizes the speedup of a two-stage forging anvils system. In [88] large-size combinatorial bioinspired algorithms are deployed in a pool of resources using a Condor grid-enabled framework to select features in a near-infrared spectroscopic data mining system.

Text mining applications have also seen an initial effort to take advantage of the available distributed environment resources. In [165] Yu et al propose and implement a framework for text mining on a web platform. The framework includes a core network, a router, grid members and grid groups. It proceeds by considering four main phases of text mining, viz., text collection phase, feature extraction phase, structure analyzing phase and text classification, which are deployed using a defined protocol between the framework elements.

Sarnovský & Butka [111] give a theoretical description of processing textual documents and their classification or clustering, describing their processing, testing and evaluation of experimental results in a grid environment. In addition, the authors find possible ways of improving the results of classification and clustering problems. The learning algorithms used were decision trees, self-organizing maps and the centroid clustering algorithm k-means. The deployment was achieved with available distributed software systems, such as JBOWL[8] (Java Bag-Of-Words Library) and GRIDMINER[9].

In [87, 167] a framework is used to provide all the computational facilities needed to solve a text categorization problem, which is decomposed in two stages: construct the text classifier and classify new texts. The tasks are dealt with using web services, segmenting the classification task into five sub services: Text Representation, Feature Extraction, Calculation Features Weight, Training Service and Threshold Value Selection.

We note two important drawbacks of the methods presented in this section. First, they constitute frameworks where several different applications can be deployed. Although this has the benefit of flexibility, a certain effort is usually needed to customize the platform to respond to the text classification application. Second, they focus on parallelizing and distributing the learning algorithm, relegating the text classification process to a minor role. Even though the parallelization and distribution of algorithms is very interesting and can yield considerable gains, it is again a generic approach that can greatly improve if the specific application is considered. Therefore, in this chapter a framework that allows the deployment of text classification in a generic distributed environment is proposed, with no constraints on the environment or platform employed. Kernel-based learning algorithms, state-of-the-art in text classification [122], are used, but other supervised learning techniques could be used. Furthermore, it is shown how is it both possible and advantageous to deploy text classification in a cluster environment, notwithstanding the use of available middleware distributed platforms and existing sequential code. This kind of effort is rarely made, and to the best of our knowledge this is one of the first, if not the first, for text classification.

[8] http://sourceforge.net/projects/jbowl
[9] http://www.gridminer.org/

5.3 Deployment in the Distributed Environment

Parallelization of tasks is nowadays one of most popular ways to minimize the execution time on one hand, and on the other, to increase the accuracy of the solution in a given amount of time. The efforts towards parallelization have increased with the availability of middleware platforms, capable of exploiting inactive computational resources in institutions, which are enabling parallelization techniques by using existing sequential code. A methodology for describing task scheduling and for deploying tasks in distributed environments is given below. Later, the distributed environment used in the experiments is presented together with a simple mathematical model which adequately describes it.

5.3.1 Task Scheduling and Direct Acyclic Graphs

An important issue in distributed computer systems is task scheduling [15]. The scheduling scheme, composed of many tasks, is commonly represented by a direct acyclic graph or DAG, with nodes representing tasks, and arrows between them representing task dependencies and underlying dataflow. More precisely, any task connected to some other (dependent) task should be completed before another task can be started. Scheduling of tasks in grid environments is usually regulated through a central manager, which ensures the correct execution of DAGs.

It is also common in loosely coupled distributed environments for the central manager to take care of communication between tasks, which is accomplished by transferring files from and to the manager. Execution of each task thus consists of transferring input and executable files from the manager to an executing node, running executable files on that node and transferring the task's output files from the executing node back to the manager. In the case of a dependent task, the manager takes care of information exchange by setting the output files of preceding tasks as input files for the dependent task. Data are usually exchanged by middleware native mechanisms. When large chunks of data are to be exchanged, a shared file system hosted on the central manager is preferred.

5.3.2 DAG Design in a Distributed Environment

DAGs can be efficiently optimized by following the generally adopted methodology of Ian Foster [103] which will be carried out in a distributed environment (see Section 5.3.3). There are four design steps: partitioning, communication, agglomeration and mapping. The focus of the first two is to find as much parallelism as possible, while the latter two consider the capabilities of the underlying environment.

In the partitioning step, the data and the computation of a task are divided into small parts that can be computed in parallel. To assure the scalability of the task, divisions where the number of small parts increases with problem size, are preferred. When an objective is to use sequential code in parallel design, the data partitioning is more feasible.

Communication between tasks, which is not needed in sequential designs, represents the overhead of parallel designs. In centralized task scheduling systems the only possible communication is point-to-point distribution and collection of files at the start and completion of tasks, respectively. In this case all communication goes through the manager, which can become a bottleneck.

In the agglomeration step, the parts identified in previous steps are grouped into agglomerated tasks in order to improve performance. Grouping should be performed so as to maintain the scalability of the design. Two aspects should be considered to avoid expensive communication. The first is to group the previously identified parts of each task in order to optimize communication; this is preferably achieved by sending the same executables and supporting files to each task. The second is to merge consecutive tasks to avoid expensive communication with the master. This is only possible when a certain task needs only intermediary files produced by the preceding task. It is very important to evenly balance the load around computing nodes by creating agglomerated tasks with roughly the same computational and communication complexity.

The mapping step assigns tasks to the computing nodes. When data decomposition is in question, the agglomerated tasks usually have very similar complexity, meaning the computational load is balanced among tasks. A good strategy in this case is to create as many agglomerated tasks as there are available processors.

In applications where communication takes a significant portion of total execution time, special attention should be given to the optimization of scheduling schemes in terms of the number of processors used for each task. To avoid time-consuming experimentation, a model of the distributed system can be built, following certain assumptions on the communication and computation phases of the process.

The execution of a task can be split into three phases:

1. the distribution of subtasks to the computing nodes;
2. the computation itself;
3. the result collection.

According to this split, the total time needed to complete a set of subtasks can be expressed as

$$\tau_{\text{total}}(p, s_1, s_2, s_3) = \tau_{\text{distribute}}(p, s_1) + \tau_{\text{compute}}(p, s_2) + \tau_{\text{collect}}(p, s_3) \quad , \qquad (5.1)$$

with p representing the number of concurrently available computing nodes and s_1, s_2 and s_3 being complexity parameters for the distribution, computation and collection phases, respectively. In many cases, as in the text classification presented later, the size of files can be used as a complexity parameter, since it satisfactorily reflects the complexity of corresponding tasks. In order to minimize the total execution time of a task, Equation (5.1) should be minimized with respect to the number of acquired nodes p and can differ from task to task in the same DAG.

In the further analysis it is assumed that the computing nodes are homogeneous, exhibiting the same computing and communication capacity. According to the grid-based environment presented it is necessary to ensure that each subtask gets its own

copy of the executable files and the supporting files of size s_{exe}. However, each subtask only needs a proportional part of input files s_{in}. The model simply assumes that the connections between the master and each of the computing nodes are established sequentially, whereupon the file transfer is performed in parallel. Therefore, the distribution time can be modeled as

$$\tau_{\text{distribute}}(p, s_1) = \tau_{\text{data}}(s_1) + p \cdot \tau_{\text{service}}(s_1) \quad , \quad s_1 = s_{exe} + \frac{s_{in}}{p} \quad . \tag{5.2}$$

When the input data is split in even partitions, the contribution of the computation phase to the total time becomes

$$\tau_{\text{compute}}(p, s_2) = T(s_2) \quad , \quad s_2 = \frac{s_{in}}{p} \quad , \tag{5.3}$$

with $T(\cdot)$ representing the task dependency.

In the collection phase we can expect that each of the p subtasks contributes approximately the same to the total output file s_{out}. Thus, the collection time can be expressed as

$$\tau_{\text{collect}}(p, s_3) = \tau_{\text{data}}(s_3) + \tau_{\text{service}}(s_3) \quad , \quad s_3 = \frac{s_{out}}{p} \quad . \tag{5.4}$$

Figure 5.1 shows an example of a task distribution to one node ($p = 1$) and to four nodes ($p = 4$). For comparison, the computing time of a task in a centralized environment (stand-alone machine) of equal capacity is given to emphasize the communication costs.

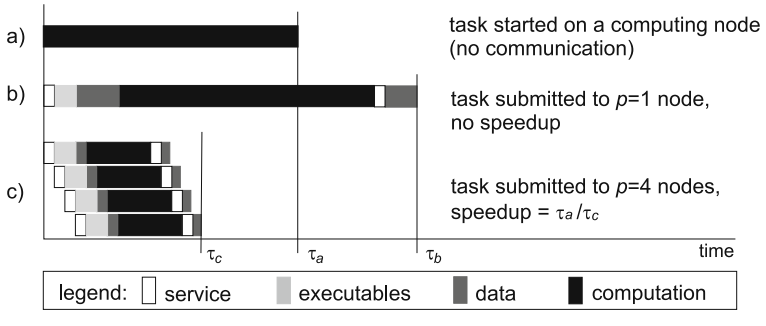

Fig. 5.1 Example of task distribution. Service time τ_{service} is modeled as constant and the computation phase is linearly dependent on the file size.

The main goal of the methodology proposed in this section is to optimize the DAG to obtain significant speedups in data mining tasks. Guidelines that can be followed are now given to help systematize the procedure, despite the fact that they constitute a simplification and so omit many details:

- Time consuming tasks (bottlenecks) should be identified, parallelized and distributed (partitioning);
- The number of subtasks for a task is defined both by an evaluation of the task complexity (building a model) and by the number of available processors (communication and mapping);
- Partitioning can be accomplished functionally or by data splitting and, depending on the constraints of the specific application, additional processes that join partial results may be needed (partitioning and communication);
- When dependent tasks exchange large files not needed subsequently, they should be joined (communication and agglomeration);
- When tasks need few computational resources and have the same input files, they can be joined (agglomeration, communication and mapping).

5.3.3 Distributed Environment for the Experimental Setup

The grid environment used in the experiments consists of a cluster of 16 machines with 3 GHz Pentium 4 processors and 1 GB RAM each. The computers are connected over a 1Gb local area network through a fast Ethernet switch. Desktop computers outside the cluster can join the pool over a 100 Mb local area network. Two freely available middleware platforms for grid computing are set up in the cluster: Condor[10] and Alchemi[11]. One of the machines is the master of the cluster and runs Linux. Besides standard services, e.g. http and ftp, it also serves as the Condor manager. The remaining 15 machines are computing nodes dedicated to job execution and running Windows. In the case of Alchemi a maximum of 14 computing nodes can be used, since one node has to serve as the Alchemi manager.

5.3.4 Model of the Environment

To effectively deploy a text classification task in a distributed environment, first it is needed to construct a model of the environment, analyzing the tasks and determining a procedure to represent them. Each task is thus divided in its fractions, namely distribution, execution and collection times so that a generic model can be inferred.

Optimization of scheduling schemes depends on the underlying architecture of the distributed environment, therefore it is of vital importance to consider the communication as well as computation costs. To estimate the communication costs an experiment consisting of a set of sequential tasks mostly involved in file transfer was performed. The dependency between the execution time of the transfer tasks and the actual file sizes is represented in Figure 5.2a) for Condor and in Figure 5.2b) for Alchemi.

According to the behavior of the middleware platforms, pure file transfer was treated separately from all other essential processes involved in communication,

[10] http://www.cs.wisc.edu/condor
[11] http://www.alchemi.net

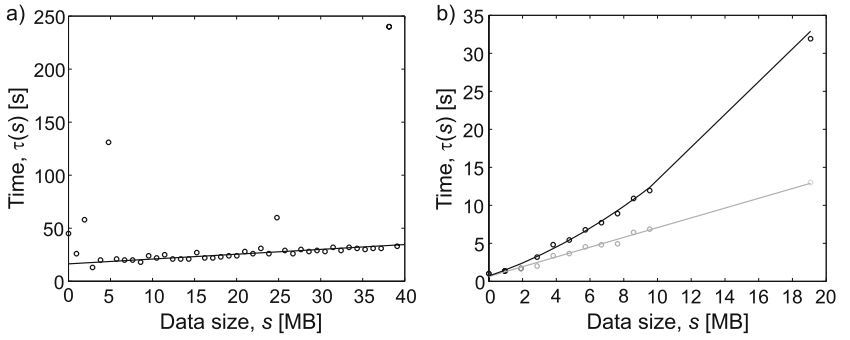

Fig. 5.2 Time to transfer data in a) Condor and b) Alchemi middleware. In the latter, pure data transfer was measured separately (gray color). Measurements are represented by circles and the model equation by solid lines.

such as middleware service processes, submission process and network latency, resulting in an overall time for a task in the experiment

$$\tau_{\text{communication}}(s) = \tau_{\text{transfer}}(s) + \tau_{\text{service}}(s) \quad , \tag{5.5}$$

where s represents the file size, τ_{transfer} is the time needed for file transfer and τ_{service} relates to all other processes. As can be seen in Figure 5.2a), omitting the random spikes caused by Condor not responding due to its services, results in a linear relation composed of pure data transfer $\tau_{\text{transfer}}(s) = C_T^{-1}\, s$, $C_T = 2.2$ MB/s and constant service requirements $\tau_{\text{service}}(s) = C_S$, $C_S = 16.4$ s.

According to the tests on Alchemi, presented in Figure 5.2b), a linear model is also used for the file transfer $\tau_{\text{transfer}}(s) = A_T^{-1}\, s$, $A_T = 1.59$ MB/s. Other processes exhibit quadratic dependency, $\tau_{\text{service}}(s) = A_{S0} + A_{S1}\, s + A_{S2}\, s^2$ with $A_{S0} = 0.69$ s, $A_{S1} = 0.16$ s/MB and $A_{S2} = 0.043$ s/MB^2, mainly due to the poor memory management when preparing files for transfer.

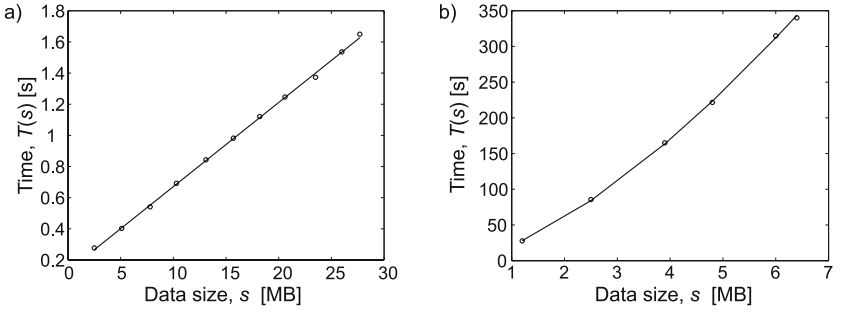

Fig. 5.3 Computation time dependence on input file size for a) acquisition task and b) dictionary task. Measurements are designated with circles and the models with solid lines. The model for the first is $T(s) = 0.131 + 0.054s$, and for the second $T(s) = 21.3\, s^{3/2}$.

In addition, the computing complexity of the tasks can be easily estimated by running programs on a stand-alone computer. For example, the tasks involved in text classification manifest linear or power-law dependence on the input data size, as shown in Figure 5.3.

5.4 Design of Distributed Text Classification Scheduling Schemes

Text classification consists of many consecutive pre-processing, training and testing tasks, as explained in Chapter 1. To efficiently port a sequential algorithm to a distributed environment, it is important to be aware of dataflow between its tasks. This dataflow for text processing tasks is described next. Then, based on the dataflow and the concepts of the design methodology described in the previous section, text classification tasks are distributed in the given environment.

5.4.1 Dataflow in Text Classification

In a sequential implementation, each task is carried out by stand-alone executable code, capable of processing predefined input files and storing results in corresponding output files. However, when a distributed deployment is intended, dependencies between tasks must be identified. Analysing a text classification flow, we can define the following subtasks, graphically represented with a DAG in Figure 5.4.

A. Acquisition task, which scans the input files to retrieve the documents, identify them with their topics (classes) and their role (training or testing). Reuters-21578 dataset is organized in eleven input files, whereas RCV1 dataset has ten input files. Hence, the initial task, denoted with letter A, is composed of eleven (or ten) subtasks A0 - A10 (A0 - A9), where each is processing one of the input files. To organize data for further processing, the partial outputs from individual runs are gathered. Depending on the arguments the executable code can separately gather training topics in subtask A'0, testing topics in subtask A'1, training documents in subtask A'2 and testing documents in subtask A'3. Whilst the first two are only needed in training and testing, the latter two are further pre-processed.
B. Task B parses, i.e. applies stemming and removes stopwords from the files representing training and testing documents. Therefore, task B can be treated as two subtasks, B0 and B1, using the same executable code but different input files, one containing training and the other testing documents.
C. In the dictionary task the distinctive words from parsed training and testing documents are indexed and their document frequency is calculated.
D. Words appearing in less than a definable threshold number of documents are removed (cleaned) from the obtained dictionary.

E. Afterwards, document-word matrix representation is obtained from parsed training and testing documents based on the cleaned dictionary. The values in matrices are also linearly scaled to avoid overflow or underflow in the learning machines. The task is divided into two subtasks, E0 responsible for training documents and E1 for testing documents.
F. Training and testing sets are constructed according to the specifications of publicly available benchmark corpora splits. Each line in a training or testing set represents one document with as many inputs as there are words in the cleaned dictionary and one output, identifying the document as belonging or not to the given category. Whilst inputs are taken from the document-word matrix representation of testing and training documents, the outputs are obtained from training and testing topics resulting from tasks A'0 and A'1. The classification into ten categories results in ten training sets (F0 - F9) and ten testing sets (F10 - F19) being prepared.
G. Finally, models for classification into ten predefined categories are built. The training algorithm is run ten times, one for each category in a binary classification scheme (one-against-all), suggesting that the adjustment of model free parameters is treated as a training subtask G0 - G9.
H. Similarly, the ten models are tested in ten subtasks H0 - H9 of the testing task. Each subtask reveals the performance and generalization capabilities of one of the models obtained in the previous task on corresponding testing set.

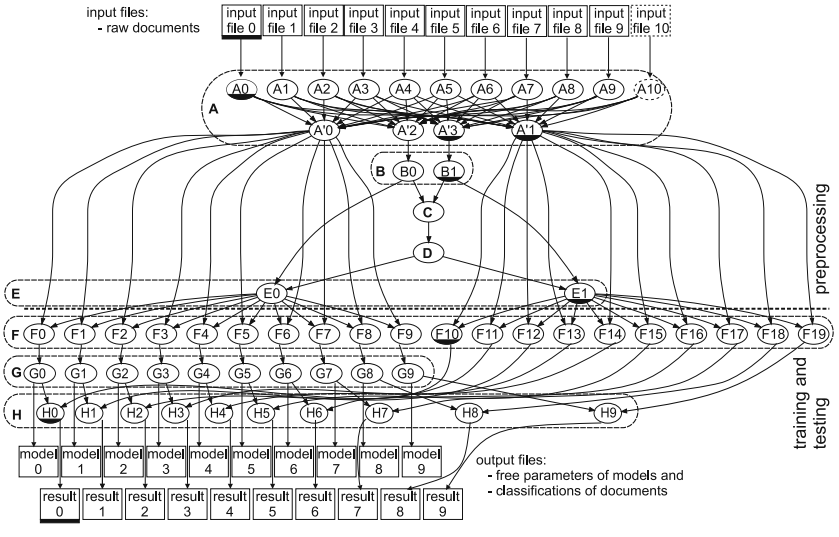

Fig. 5.4 Text classification dataflow represented using a DAG with arrows identifying task dependencies with underlying file exchange. Dashed lines indicate optional input file and subtask. When a new input document is presented to the system and there is no need to update the models, only the files and the subtasks with a colored bottom part are involved in processing.

When a new document is presented to the system and there is no need to update the classification models, only the files and subtasks indicated by the colored bottom part are involved in processing (see Figure 5.4).

5.4.2 Optimization of Scheduling Schemes

The text classification dataflow given in Figure 5.4 itself already has an underlying parallelism of subtask execution on distributed nodes. But this distribution is far from optimum. To better join an application and a distributed environment the procedure described in Section 5.3.2 can be followed.

Bearing in mind the approach of using sequential code to the greatest possible extent, the tasks identified in the previous section represent the most detailed partitioning in terms of computation. Fortunately, the dataflow itself indicates data partitioning in tasks A, B, E, F, G and H. One of the most critical tasks in text classification is task C (followed by D), which has to gather all partial results to build a global dictionary and as such represents a significant bottleneck. Below, the explanation of DAG optimization is split into two parts due to the two main learning machines, SVMs and RVMs. In both cases, the designs of DAGs for the two datasets, Reuters-21578 and RCV1, are further detailed. Afterwards, the optimization of ensembles of learning machines on the given datasets is discussed.

5.4.2.1 SVMs in a Distributed Environment

The first step is to optimize the initial DAG deployment using SVM models. Task A (acquisition) can be partitioned into as many subtasks as there are documents in a given dataset. In the case of Reuters-21578 dataset, the split would lead to as many as 21,578 subtasks. Furthermore, parsing in task B can work on subtasks consisting of an arbitrary number of documents. Building a dictionary of words in task C is very time-consuming and needs to be parallelized. Instead of gathering the documents before task C, we can create dictionaries of documents in an arbitrary number of subtasks together with their partial document frequencies, and afterwards combine them into a global dictionary. These modifications require undemanding programming of the procedure for gathering of partial dictionaries.

Tasks A, B and C exchange many intermediate files that are not needed in other tasks. By agglomerating them into a joined ABC task, the overall file transfer is significantly reduced. By considering the computation and communication costs in the process of data agglomeration, the optimal number of subtasks can be obtained. To avoid time-consuming experimentation, the model of the environment presented was used to determine the suitable number of subtasks, resulting in the dependency shown by the solid line in Figure 5.5.

According to the model, the optimal number of subtasks in Condor middleware is four. However, the natural organization of the dataset into 11 files led us to five subtasks, without heavily degrading computing performance. More precisely, by assigning three smallest input files to one subtask and pairs of remaining files to the

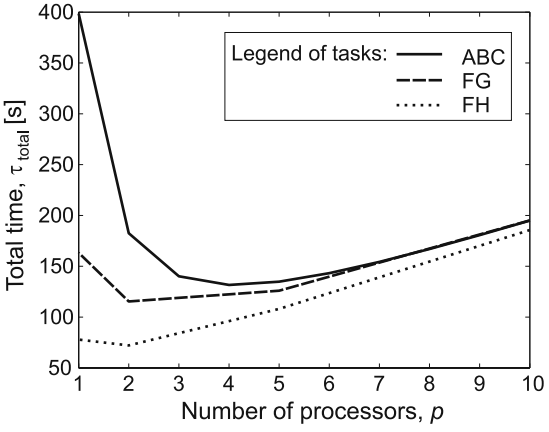

Fig. 5.5 Model of execution times vs. number of processors for text classification tasks.

other four subtasks, subtasks of balanced complexity were obtained. Task ABC is followed by a new gathering task, which is split into five subtasks exhibiting functional and data parallelization. And so, subtasks A'0 and A'1 still gather training and testing topics, whilst tasks B'0 and B'1 combine parsed training and testing documents, respectively. There is also a joined task C'D that is used for combining partial dictionaries and their document frequencies into the global dictionary and further cleaning it. In this case, the functional agglomeration was performed to reduce unneeded file transfer. Following a similar approach we obtained an optimal number of eleven subtasks for Alchemi. However, for easier comparison of the DAG execution in different middleware environments, the Condor settings were also used for Alchemi.

The task of building training and testing sets (F) with SVMs creates large intermediate files for each category, only needed by the subsequent training (G) or testing (H) tasks. For that reason, new joined tasks (FG and FH) were created, reducing overall file transfer.

Similar to the procedure used with the ABC tasks, the processing times of FG and FH tasks were evaluated to determine a suitable number of subtasks, resulting in the dependency shown in Figure 5.5 by dashed and dotted lines, respectively. In both cases the optimal number of FG and FH subtasks for Condor is two, each one taking care of building or testing five classification models. As a result of these modifications, we finally reached the refined DAG shown in Figure 5.6.

By following the described optimization procedure, for the 35-times larger RCV1 dataset and the same learning machines, a rather different partitioning of tasks was obtained. For the ABC task, instead of five subtasks, ten subtasks are suggested and for the FH task, five subtasks are found.

As the gathered training and testing sets are much larger than for the Reuters-21578 dataset, task E was identified as a bottleneck, so additional steps were taken to permit its distribution. More precisely, the dictionary gathering was reprogrammed

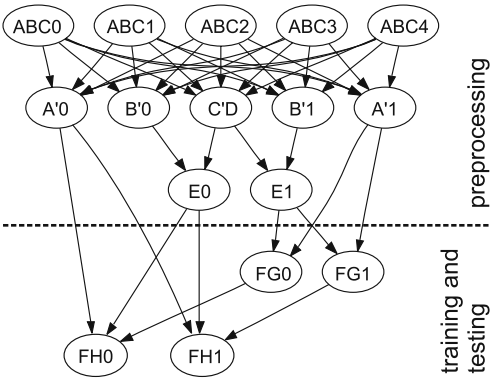

Fig. 5.6 Text classification DAG for the SVM model with the Reuters-21578 dataset.

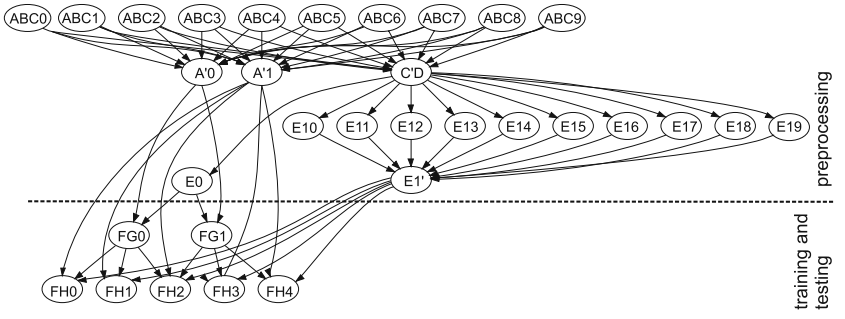

Fig. 5.7 Text classification DAG for the SVM model with the RCV1 dataset.

to take into account the parsed training and testing documents, and a new gathering task was programmed considering document-matrix representation peculiarities. Furthermore, in the case of the RCV1 dataset there are far more testing documents than training documents, suggesting a split of subtask E1 into many smaller subtasks, which require a new gathering task, E1'. Taking into account the proposed split and gathering required, the model suggests using ten subtasks E10-E19. Figure 5.7 shows the DAG achieved for SVM learning.

5.4.2.2 RVMs in a Distributed Environment

Besides the SVM classification models, RVM classification models were also used for classification of the datasets. Since the datasets are the same, there are no changes in the pre-processing tasks.

Contrary to SVMs, the learning times for RVMs are much longer. Using all available training documents for the RCV1 dataset is not computationally feasible. Therefore, to achieve comparable computational complexity of the training task in both datasets, training sets with 2,000 training documents were defined.

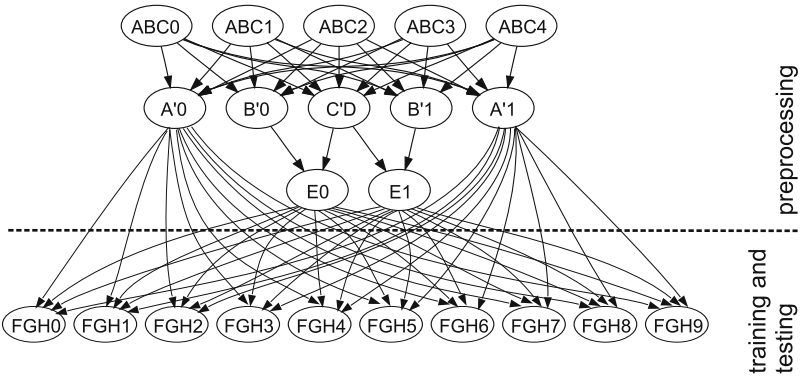

Fig. 5.8 Text classification DAG for the RVM model with the Reuters-21578 dataset.

Additional reorganization of training and testing tasks is necessary, since the software package used[12] requires the simultaneous input of training and testing sets. Thus a joined training and testing task, named FGH, is defined for RVMs. Since the computation largely prevails over communication, it is reasonable to arrange subtasks in such a way that each of them handles training and testing of one of the ten classification models. The refined DAG for Reuters-21578 is given in Figure 5.8. In the case of the RCV1 dataset, the training and testing parts of the DAG remain unchanged, whereas the initial pre-processing part is replaced by the one given in Figure 5.7, with the task E1' connecting to the FGH subtasks instead of subtask E1.

5.4.2.3 Ensembles of Learning Machines

Along with kernel-based machines, ensemble techniques present state-of-the-art results in several applications, including text classification [122], which was corroborated by the previous two chapters. Therefore, they were chosen for implementation in the distributed environment to improve classification performances. In fact, with an ensemble structure, profiting from a distributed environment setup and available computing cycles, classification performance can be improved without penalizing processing time.

In the following the ensemble strategies undertaken for SVMs and RVMs are outlined. For both, the elected voting algorithm is majority voting, where each base classifier votes on the class the document should belong to and the majority wins. The differences between SVM and RVM ensembles lie in the choice of the k experts that are constrained by the learning algorithms. While the SVM ensemble is created using different learning parameters, the RVM ensemble is defined using data partitioning.

[12] http://www.miketipping.com/index.php?page=rvm

SVM Ensemble

The purpose of this approach was to develop homogeneous ensembles, as detailed in Chapter 3, i.e. using the same learning algorithm. SVMs are sufficiently scalable to use all training examples for each model (which is not true for RVMs). As indicated in Section 3.2.1, we explore different parameters for SVMs[13], resulting in four different learning machines: (i) linear default kernel, (ii) RBF kernel, (iii) linear kernel with trade-off between training error and margin set to 100, and (iv) linear kernel with the cost-factor, by which training errors in positive examples outweigh errors in negative examples, set to 2.

To adapt the optimized DAG presented in Figure 5.6, the learning and evaluation components had to be altered. The training and testing tasks (FG and FH) were thus quadrupled to represent the four different learning machines that constitute the ensemble. Moreover, after these tasks, a new dependent ensemble task (I) was created to implement the majority voting among the SVM classifiers.

RVM Ensemble

RVMs have the drawback of low scalability. As mentioned in Chapter 4, for the baseline RVM setting, training sets of up to 2,000 examples can be learned in reasonable computing time. For the distributed approach followed in this chapter we have therefore devised an RVM ensemble strategy that could take advantage of the predominance of training examples in the text classification benchmarks. The size and number of the training sets used in the RVM ensemble modeling depend on the available computational power, but the more training examples used usually results in more diversity and better performance. We have constructed seven smaller evenly sized training sets, each consisting of 1000 randomly disjointed sample documents from the available training set. Then, from each training set a model is learned, and these models constitute the ensemble individual classifiers. After this learning phase, a majority voting scheme is implemented in the testing phase to determine the ensemble output decision, taking as output value the average value of the classifiers that corroborated the majority decision.

As with the SVM ensemble, the changes on the DAG presented in Figure 5.8 lie mainly in the learning and testing tasks (GH) to accommodate the seven models for each category, and in adding an extra ensemble phase that depends directly on the testing tasks.

5.5 Experimental Results

This section presents the results obtained in terms of processing time and classification performance. The proposed framework for distributed text classification is tested on the Reuters-21578 and RCV1 datasets with the SVM and RVM models.

[13] http://svmlight.joachims.org/

The study also includes the ensemble strategies for the two approaches. Experiments are conducted in both Condor and Alchemi distributed environments, while a centralized approach is used for comparison.

Even though the processing times vary slightly in a centralized approach, differences between runs can be significant in distributed environments. Therefore, to establish confidence bounds for thetests and thus ensure statistical significance, each experiment was repeated 30 times, and average results are presented.

5.5.1 Processing Time

Table 5.1 presents the processing times for the initial DAG, emerging directly from dataflow (see Figure 5.4), and the optimized DAGs (see Figures 5.6, 5.7 and 5.8) for Alchemi and Condor using SVM models for Reuters-21578.

Table 5.1 Processing times using SVM models for Reuters-21578 (times in seconds).

	Sequential	Condor	Alchemi
Initial DAG	516	1,487	1,075
Optimized DAG	516	543	242

The results for the initial DAG in both Condor and Alchemi platforms show how a straightforward deployment in a distributed environment can have adverse effects on processing times, since the distributed text classification systems approximately double the centralized approach sequential time.

In previous sections significant effort has been devoted to the analysis of the distributed environment model. Based on a clear understanding of the problems involved, i.e. communication overhead and unbalanced node complexity, we proposed the refined DAG (Figure 5.6). In this case, an improved solution is reached, taking full advantage of the computational resources available in the cluster environment. Figure 5.9 shows the processing times for both datasets using SVM and RVM models with optimized DAGs in both centralized and distributed environments. To emphasize the cost of communication the processing times are split into two parts - communication including middleware service requirements and pure computation. The small horizontal lines on each bar represent the standard deviations for the 30 runs.

The four arrangements exhibit the same trend, regardless of the tested setting. Alchemi always presents a better processing time than Condor, and Condor better than the sequential approach. It was expected that the distributed platforms would exhibit a better performance than the sequential approach, since they take advantage of a far greater availability of computational power. Alchemi excels because it is much simpler, having less demanding services, thus dealing with task management more efficiently. The results are far more expressive for the RCV1 dataset, since it is 35 times larger than the Reuters-21578, which affects both communication and execution.

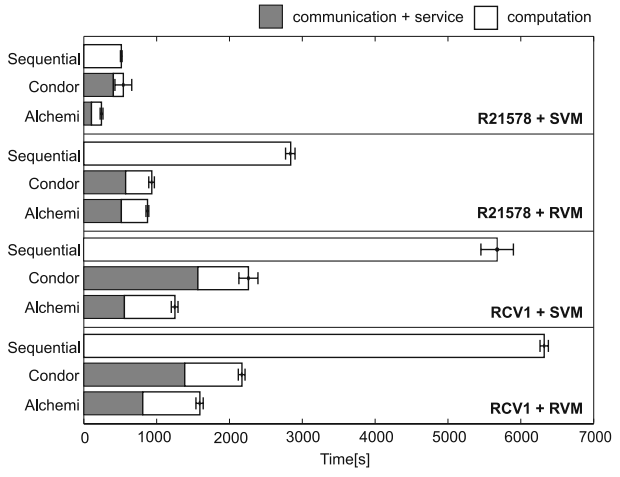

Fig. 5.9 Processing times for SVMs and RVMs models with optimized DAGs.

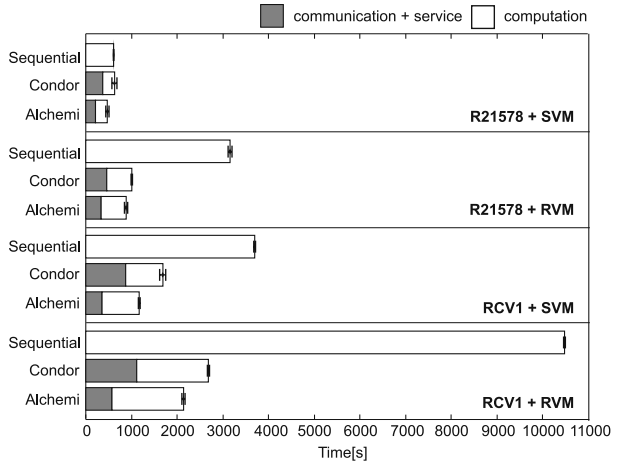

Fig. 5.10 Processing times for ensemble models with optimized DAGs.

For the ensemble strategies, as Figure 5.10 shows, similar trends to the single learning machine approach are observed. As the learning burden is much larger, the effect of communication and service requirements is less pronounced, leading to more significant improvement in processing times. The standard deviations resulting from the 30 experiments agree with the previous results.

There is however a difference when comparing SVMs and RVMs processing times for Figures 5.9 and 5.10. For the ensemble setting, the RVM is the more time-consuming learning machine, as was expected, since RVMs are known to have scaling issues (see Chapter 4).

One of the frequently used measures for a distributed system evaluation is speedup, defined as the ratio between processing times in the sequential approach and the distributed platforms,

$$\text{Speedup} = \frac{\text{Sequential processing time}}{\text{Parallel processing time}} \quad . \tag{5.6}$$

The speedups for Condor and Alchemi compared with the sequential setting in the several arrangements are presented in Table 5.2.

Table 5.2 Speedups obtained for Condor and Alchemi with both datasets.

	SVM		SVM Ensemble	
	Condor	Alchemi	Condor	Alchemi
R21578	0.95	2.13	0.97	1.29
RCV1	2.51	4.54	2.19	3.16

	RVM		RVM Ensemble	
	Condor	Alchemi	Condor	Alchemi
R21578	3.04	3.25	3.13	3.57
RCV1	2.91	3.97	3.90	4.89

As could be predicted from the processing times, there is generally an improvement in speedup with the deployment in the distributed environment. Alchemi yields a better speedup than Condor in most settings, since Condor is more concerned with high execution burdens, whereas Alchemi deals better with frequent file transfer, as observed for text classification applications. Compared with SVMs, RVMs also present a greater potential for parallelization, explaining their better performance in distributed environments. Given that RCV1 is much larger than Reuters-21578, its speedup also tends to be larger.

For SVMs, the difference between Condor and Alchemi speedups is more evident, almost doubling (from 0.95 to 2.13 and 2.51 to 4.54) for the SVM baseline setting and increasing by 50% (from 0.97 to 1.29 and 2.19 to 3.16) for the SVM ensemble setting.

RVMs show more potential for speedup, with speedups around 3 in the RVM baseline setting and between 3 and 5 in the RVM ensemble setting. Differences between Condor and Alchemi for RVMs are minor in both the RVM baseline (from 3.04 to 3.25 and 2.91 to 3.07) and the RVM ensemble (from 3.13 to 3.57 and 3.90 to 4.89) settings.

Generally, speedups are more pronounced for the RCV1 dataset and for the ensemble approach. As these are the most demanding settings in terms of computing power, this outcome was expected. Considering the number of processors available and that a very complex and not fully parallelizable problem was undertaken, the resulting speedups represent a real improvement in pursuit of the initial goal, i.e., to deploy a complex text classification task efficiently in a cluster environment.

In addition to speedup, efficiency is commonly used in order to measure utilization of processors. Given as

$$\text{Efficiency} = \frac{\text{Sequential processing time}}{\text{Processors used} \times \text{Parallel processing time}} = \frac{\text{Speedup}}{\text{Processors used}} \tag{5.7}$$

it indicates the fraction of time the processors are used for a given computation. In the DAGs presented the number of processors used is changing dynamically from task to task. In this situation the worst case efficiency can be calculated by using the maximal number of processors simultaneously involved in the computation. Of course, this reflects a poor utilization of processors in a distributed system. However, the released processors can be employed in some other applications. The efficiencies of the presented designs are summarized in Table 5.3.

Table 5.3 Efficiencies obtained with Condor and Alchemi setups on both datasets with maximal number of used processors indicated in parenthesis.

	SVM		SVM Ensemble	
	Condor	Alchemi	Condor	Alchemi
R21578	0.19 (5)	0.43 (5)	0.19 (5)	0.26 (5)
RCV1	0.25 (10)	0.45 (10)	0.22 (10)	0.32 (10)

	RVM		RVM Ensemble	
	Condor	Alchemi	Condor	Alchemi
R21578	0.30 (10)	0.32 (10)	0.45 (7)	0.51 (7)
RCV1	0.29 (10)	0.40 (10)	0.39 (10)	0.49 (10)

In all cases the efficiency of parallel designs under Alchemi is higher than under Condor which indicates that Alchemi has more efficient task management, including middleware service requirements and file transfer.

It can also be seen that the efficiencies in the larger RCV1 dataset compared with the smaller Reuters-21578 dataset are largely maintained at the same level or even improved upon when more processors are used. This indicates the scalability of the suggested text classification solutions. Since the number of processors in DAG processing changes dynamically from task to task, and since the DAGs are specially optimized for each dataset, a deeper analysis is not really feasible.

5.5.2 Classification Performance

In Table 5.4 the summary of $F1$ classification results is presented. Concerning classification performance, the comparison of the SVM and RVM approaches per se has already been presented in previous chapters. Here we show a comparison of approaches. SVMs generally provide better average F1 performance results than

Table 5.4 $F1$ performance results for both datasets and learning machines.

Reuters-21578

	SVM	SVM Ensemble	RVM	RVM Ensemble
Earn	98.14%	97.91%	97.52%	97.03%
Acquisitions	95.59%	95.75%	91.57%	93.59%
Money-fx	74.64%	77.24%	57.14%	69.62%
Grain	82.76%	86.18%	76.52%	80.30%
Crude	84.87%	87.12%	69.51%	70.86%
Trade	69.90%	74.67%	50.27%	68.57%
Interest	74.73%	76.68%	58.29%	62.92%
Ship	70.59%	79.19%	66.21%	62.77%
Wheat	75.63%	83.72%	75.18%	77.17%
Corn	70.00%	79.17%	67.37%	66.67%
Average	79.68%	83.91%	70.96%	74.95%

RCV1

	SVM	SVM Ensemble	RVM	RVM Ensemble
CCAT	92.78%	92.27%	82.99%	81.55%
GCAT	56.63%	62.40%	35.83 %	40.86%
MCAT	90.72%	91.40 %	76.36%	77.78%
C15	71.92%	78.66 %	76.47%	75.13%
ECAT	74.97%	80.11 %	60.43%	65.68%
M14	72.39%	78.33%	58.72 %	71.43%
C151	16.01%	47.43%	31.48%	32.95%
C152	5.18%	6.13%	4.93%	3.07%
GPOL	68.78%	75.24%	57.07 %	64.37 %
M13	69.69%	85.52%	66.12%	75.02 %
Average	61.91%	69.45%	49.64%	58.78%

RVMs for the same datasets. In [54, 148] detailed studies on SVMs applied to text classification can be found. Baseline results presented here concur with these studies. For RVMs, the published research is usually not baseline but the result of specific algorithms optimizations, such as in reference [36]. As can be seen in Table 5.4, the average performance differs by around 10% (79.68% to 70.96% for Reuters-21578 and 61.91% to 49.64% for RCV1). It must be stressed, however, that an RVM uses only a fraction of the training examples owing to computational constraints.

The ensemble strategy in most cases improves the classification performance compared with a single learning machine (see Table 5.4), while maintaining relative performance (around 10% difference from 83.75% to 69.45% for Reuters-21578 and from 69.45% to 58.78% for RCV1). The four ensemble SVM machines were built by varying parameters over all training samples, while the same learning parameters and different training samples were used for all RVM machines. Considering that SVM and RVM present state-of-the-art results for text classification, their improvement with the ensemble strategy constitutes progress in terms of classification performance.

5.5.3 Discussion of Results

In this chapter an architecture for the seamless deployment of data mining tasks in any given distributed environment was proposed, using state-of-the-art learning algorithms and making it possible to speedup pre-processing, learning and evaluation processes.

To exploit the proposed techniques, a high-dimensional challenging problem of text classification was used, where there are tens of thousands of learning documents and each document can also have thousands of features. To cope with this complexity the kernel-based learning machines employed throughout this book, i.e. SVMs and RVMs, where used. SVMs, while exhibiting extraordinary generalization capabilities, also scale linearly with the training set. On the other hand, RVMs are capable of introducing Bayesian priors which, using a priori knowledge, make the model generalize well and incorporate thepreferences into the learning algorithm. Furthermore, ensembles of kernel-based machines were employed to improve classification performances. Therefore, the results relate to both processing time and classification performance.

The speedup of a text classification problem depends on the size of datasets, on the complexity of classification algorithms and on the deployment in the distributed environment. Generally, the processing times are improved with the deployment in the distributed environment.

Comparing the speedups between the distributed middleware platforms Condor and Alchemi, we can conclude that for Alchemi the distributed approach reduces the processing time more significantly. Alchemi is internally much simpler, having less demanding services, and so deals better with task management, which is especially pronounced in frequent handling of tasks when file transfer prevails over the execution burden. One of its advantages in terms of better task management is the support and adaptation to only one operating system (i.e. Windows with .NET framework). On the other hand Condor has to take care of different operating systems and different architectures increasing service requirements which are probable reasons for occasional internal delays. The importance of selecting the right platform is also demonstrated in Table 5.2. Despite of the fact that direct acyclic graphs optimized for Condor were used on both platforms, Alchemi outperforms Condor. In terms of learning algorithms, RVMs, when compared with SVMs, also show better potential for parallelization. This result was foreseeable since RVMs have noticeably larger computational burdens and were in fact parallelized using data partitioning of the inputs. In addition, ensemble strategies overload the cluster environment with processing demands resulting from the several models they must build, therefore constituting a great source of parallelism. Hence, ensemble techniques offer the larger speedups.

As illustrated in Table 5.2, the speedups are more noteworthy for larger datasets, like RCV1. This is mainly due to the fact that the computation in the nodes prevails over the communication between them. In the case of smaller datasets like Reuters-21578, acquaintance with the problem and its distribution are more important to improving the performance, which is shown in Table 5.1.

Analyzing the $F1$ classification performance results presented in Table 5.4, it is evident that SVMs generally perform better than RVMs, probably because of the RVMs' scaling problems that do not allow the use of all training examples. Comparing baseline kernel-based machines with ensemble strategies is, as expected from previous chapters, favorable to ensemble strategies, reinforcing the idea that no single classifier is always best. In SVM this improvement is achieved through classifier diversity, a fundamental characteristic of ensembles, while for RVM the profit is accomplished by subsampling the training set.

5.6 Conclusion

In this chapter a new architecture for deploying text mining in a cluster computing environment was developed. The framework employs a combination of the text classification task (and data) decomposition, configuration evaluation through the modeling of the design phases, and the high performance distributed computing model. The main features of the kernel-based learning machines, upgraded with the ensemble strategies, make it possible to create models for text data mining and knowledge-discovery in texts which are generally used aliases for the process of extracting relevant and non-trivial information from text. In this way, it was possible to integrate content, knowledge and learning, which is of growing importance as an effective answer to the challenge posed by the deluge of information available in digital form.

A methodology to deploy a data mining problem in a generic distributed environment was described, indicating possible optimization steps. The proposed technique is based on four steps: partitioning, communication, agglomeration and mapping. In addition, a direct acyclic graph (DAG) is used to define the tasks and dependencies, and a model is built to describe the task executions and determine the direct acyclic graph optimizations. The latter are based on a simple mathematical model of the distributed environment which takes into account experimentally obtained communication and computation costs.

It was shown that it is not only possible, but even advantageous to deploy text classification in a cluster environment, notwithstanding the use of available middleware distributed platforms and sequential code. Two benchmark datasets from Reuters database are used in the experimental work. The task and the data distributions are performed with Condor and Alchemi platforms. The results are compared with the sequential approach in terms of speed and performance and we concluded that the Alchemi platform provides larger speedups in all tests with different databases and learning models, while the SVM ensemble classification results outperform other combinations like SVM, RVM, and RVM ensemble. Furthermore, the improvement in performance achieved by the ensemble strategies has no drawback in terms of processing time, thanks to its distribution in the cluster environment.

Chapter 6
Framework for Text Classification

Abstract. The previous chapters presented a number of novel techniques to tackle a variety of problems encountered in real-world text classification settings. The common underlying thread has been the integration of knowledge in the inference of inductive learning models without penalizing processing time. This chapter unifies the main topics of this book into a framework. An inductive inference-based text classification framework will provide basic generic tools that are appropriate for a broad range of applications. New research trends in text classification are highlighted towards the end. We will focus on the particular developments in kernel methods triggered by new problems in text mining and on how to extract useful knowledge by mining relationships between data. We include a few promising research directions that are likely to expand in the future.

Inductive inference-based text classification framework

The main focus of this book has been to exploit ways to integrate knowledge in the inductive learning task, considering all available, sometimes not tangible, knowledge. We presented different approaches that exploit information offered in many formats, specifically, unlabeled examples, expert classification of active learning examples and synergies of knowledge obtained from different classifiers. Here, a unified view of all the techniques is presented and a framework for inductive text classification systems is proposed. Figure 6.1 illustrates the components of the proposed framework. A text classification system was divided into different stages: sources, pre-processing, learning and evaluation. The most important techniques have been outlined for each stage.

The sources of texts to classify are numerous and evolving every day, and there is a wide variety of formats and contexts from which a text can be supplied. However, databases and the web cover most possible supplies. In fact, web-based scenarios are becoming ubiquitous even though offline applications are still plentiful. With the boost in text sources it is envisaged that effort will be required in the pre-processing step to deal with heterogeneous, structured and multimodal data, whether in a supervised or unsupervised manner.

C. Silva and B. Ribeiro: Inductive Inference for Large Scale Text Classification, SCI 255, pp. 117–128.
springerlink.com

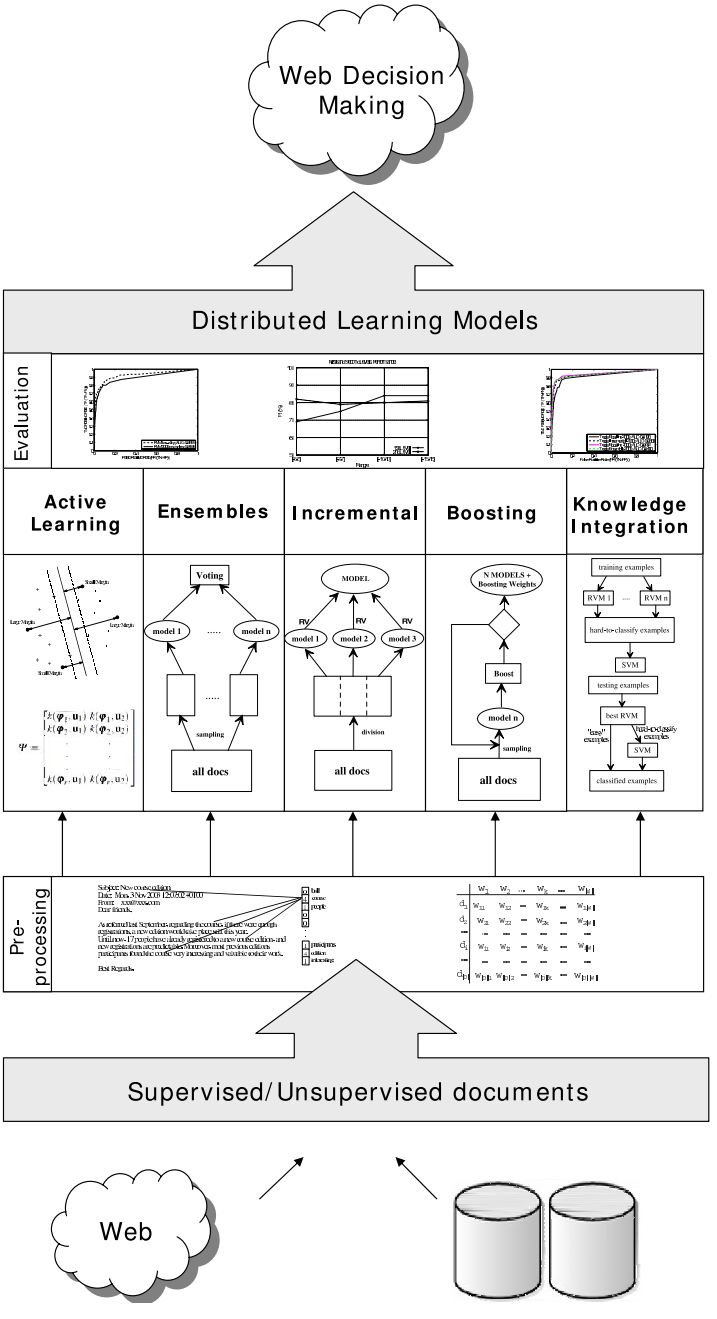

Fig. 6.1 Framework for text classification.

In addition to the representation problems, and given that text classification is a high dimensional task no matter how one looks at it, dimensionality reduction techniques will remain an imperative focus of research, especially nonlinear approaches like latent semantic indexing [29] and manifold learning [145]. Today it is argued that the pre-processing stage usually accounts for up to 80% of both time and computational efforts, and with the increasing overload of digital texts and text classification applications, it will stay that way. Research tends to focus on scaling properties of both data and methods to cope with such huge data sets. A number of attempts to achieve this goal are pursued throughout this book, following paths that lead to the use of distributed environments, an unavoidable landmark in the text classification scenario. These ideas will be detailed further in the next section.

The learning stage plays a central role in any text classification system, and was in fact the main concern of the material found throughout this book. The milestones in the learning stage are Occam's razor combined with the no free lunch theorem, i.e the simpler solution is usually better, but there is no best algorithm for all situations. To implement these ideas, techniques that both adopt simplicity and combine different characteristics of classifiers have been proposed. Together these techniques allow a comprehensive influence on all learning aspects, namely scaling, active learning, ensembles, knowledge integration and hybrid systems.

The evaluation is an important stage of text classification since it constitutes the interface with the end user who has to have confidence in the text classification system. Accuracy and error metrics are nowadays insufficient to deal with text classification evaluation. Combined measures that take into account errors in both positive and negative examples are a reality. Moreover, techniques that permit graphical visualization of performance, of which ROC curves are a paradigmatic example, are evolving rapidly and thus becoming even more important.

The framework proposes a series of techniques to enhance learning performance. Some of those important tools are outlined below.

Active learning

Cutting-edge techniques were set up with both Support Vector Machines (SVMs) and Relevance Vector Machines (RVMs). An SVM margin-based method was proposed with potential to substantially improve performance when small training sets are available. This conclusion is very important in text mining tasks since usually there are a small number of classified examples and a huge number of unlabeled ones; for RVMs we introduced an active learning RVM method based on the kernel trick. The underlying idea is to define a working space between the relevance vectors initially obtained in a small labeled data set and the new unlabeled examples, where the most informative instances are chosen. Using a kernel distance metric, a higher-dimensional space can be defined where the selection can be accomplished. The proposed active learning method not only surmounts the overload of unlabeled examples available in learning tasks like text classification, but also overcomes the scalability problems posed by RVM learning machines, selecting an optimized working set of training documents. Complexity escalation was controlled,

since the number of added documents was fixed and a simple strategy to determine those active documents was provided.

Ensembles

Ensembles have become a mainstay in the machine learning and data mining literature. The use of several models organized in a committee can be exploited both for SVMs and RVMs. A robust SVM ensemble with a two-step learning strategy using the separating margin as differentiating factor in positive classifications has been proposed. First the positive classifications given by the base classifiers were enhanced and then the maximum margin classifier was used for each example. This strategy proved to be robust and led to the conclusion that diversity in the base classifier is a more important factor than individual performance. For RVMs the ensemble strategy was applied to define smaller working sets that permit the use of all training examples in RVMs expansion to large datasets, obtaining performance and speedup gains.

Incremental

A set of divide-and-conquer methods was proposed, with decomposition techniques that explore different approaches, take advantage of RVM characteristics such as predictive distributions for testing instances and sparse solutions, while maintaining and even improving the classification performance. In the incremental approach the final training set is incrementally constructed. The dataset is divided into evenly sized smaller subsets (or chunks). Each chunk is then trained independently, possibly in parallel, resulting in a set of RVM models. The relevance vectors yielded by each model are gathered and constitute the training set of a new RVM model, which is the final incremental RVM.

Boosting

Another divide-and-conquer method was an RVM Boosting approach where several relatively weak classification rules are generated and combined into a single highly accurate classification rule. The main idea in the RVM boosting approach is to use all the training examples by sampling them into small working sets, making each classifier much weaker than it would be if trained with all available training examples. If enough models are generated, all distinctive aspects of the class can be captured and represented in the final classifier. By dividing the huge data set into smaller tractable chunks the computational load usually associated with training RVMs is mitigated. Two main distinctive steps were involved. First, instead of using the training set for training and for boosting a separate boosting set was defined. Second, considering that the RVM classifiers are in fact *not so weak classifiers*, as the boosting assumes, the same set of classifiers was presented repeatedly

to the boosting algorithm, i.e. the number of iterations is not equal to the number of classifiers, but it is proportional to it.

Knowledge Integration

Different methods for integrating knowledge into the learning task were exploited. A completely automatic SVM margin-based method called background knowledge was introduced. It incorporates into the training set new examples that are classified with larger margins. After the classification of unlabeled examples by an initial SVM model, the background knowledge approach incorporates into the training set new examples that are classified with larger margins. The rationale behind this is that these examples have a high probability (high confidence) of being well classified, and can thus be interpreted as information underlying the initial problem. However, this method should not be used with small training sets, i.e. with too weak initial classifiers. In other cases it introduces improvements.

For RVMs we proposed a two-step technique RVM based on a new similitude measure between documents that is able to manage large datasets by chunking dissimilar documents and reducing complexity. The first stage selects which training documents go to the next level, using a similitude measure between documents that is based on the co-occurrence of words. The second step of the method gathers all remaining documents from all the chunks of documents and infers an RVM classifier. The approach tends to maintain the sparse solutions given by RVMs, while improving classification performance, with some penalization on training time. However, the number of RVs is kept remarkably small, making recall phase much faster than with SVMs.

Another knowledge integration approach was a two-level hierarchical hybrid SVM-RVM model to exploit the best of SVMs and RVMs, the two state-of-the-art kernel-based learning machine currently the focus of cutting-edge research. SVMs offer better accuracy and complexity, but are surpassed by RVMs when it comes to probabilistic outputs or kernel selection. The first level of the proposed model uses an RVM to determine the less confident classified examples and the second level uses an SVM to learn and classify the tougher examples. The hierarchical approach outperforms both baseline learning machines. While the first level exploits RVMs probabilistic nature, the second-level takes advantage of SVMs accuracy properties.

Deployment in distributed environments

The computational feasibility of inductive kernel-based techniques raises a major concern in real world text applications. When applied to large amounts of data, e.g. in the magnitude of several million documents, the process can be too time consuming. Therefore, we deployed text classification in distributed environments; we combined task (and data) decomposition, configuration evaluation through the modeling of the design phases, and a high performance distributed computing model. Testing was carried out on a large-scale corpus and the results revealed that the tested

techniques can be implemented in a high-dimensional real-world situation, especially if a distributed environment is available.

6.1 Novel Trends in Text Classification

Besides the important developments mentioned in the previous section, several other attempts to import ideas from kernel methods into text mining have emerged recently. For large scale text classification inductive inference models undoubtedly need approaches to be tackled in terms of the engineering of new kernels, the development of methods to handle multiple kernels, and the use of kernel methods for graphs in text mining systems [153].

In this context text classification points to countless interesting research possibilities. In Figure 6.2 text classification is framed in terms of information semantics (knowledge, ontology and indexing), information extraction (information retrieval and search engines), and information distributed systems (GRID technology). These areas will provide promising avenues of research. In this section we highlight promising directions that are likely to develop quickly in the near future.

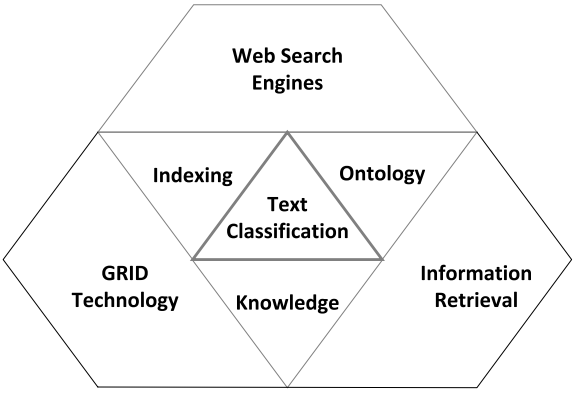

Fig. 6.2 Novel Trends in Text Classification.

We are undoubtedly in the era of the Web. As such, different facets of text classification are emerging including web search, social networks, online media (music,video, etc), content aggregation and syndication (RSS), etc. Currently the Web is still mostly accessed via a PC, but mobile devices are becoming increasingly popular, which is another major challenge for text classification.

We can expect a boom in text classification applications in the next decade [65, 168, 12] not only in relation to the common PC interfaces, but also by mixing online activity with the detection of presence, mobility or other objects.

6.1.1 Information Semantics

Semantic Web

The semantic web is as an evolving extension of the World Wide Web in which the semantics of information and services on the web are defined. Its importance to the extraction of knowledge brings up the need for research focusing on semantics, particularly on innovative approaches for text classification. Traditional algorithms must adapt to understand and respond to the user's and machine's requests to use the web content. The most usual way to solve this problem is to enrich document representation with the background knowledge in an ontology, but this presents different problems despite the potential advantages. In particular, the ontology may not cover the entire knowledge space and so some information loss may ensue. In [46], a novel text clustering method to address these issues by enriching document representation with Wikipedia concepts and category information is presented, by mapping text documents to Wikipedia concepts, and further to Wikipedia categories. The inevitable thriving of (web) ontology will empower web intelligent systems based on robust text classification systems [19, 105].

The work presented throughout this book has revolved around discriminative approaches to learning. Generative approaches such as graphical models are nevertheless acquiring visible potential. The last ten years have in fact witnessed rapid growth in the popularity of graphical models, most notably Bayesian networks, as a tool for representing, learning, and computing complex probability distributions. Graphical models provide an explicit representation of the statistical dependencies between the components of a complex probability model, making it possible to detect semantic dependencies.

Social Networks and Community Mining

A social network is considered a descriptive framework to study and analyze relations and behavior, and measure social metrics. A text classifier framework to approximately predict social relations using web documents was proposed in [83]. By employing text mining techniques the social network generation system may learn non-trivial patterns, similarities and associations among the individual persons. Considerable resources are needed to extract information about people, such as news, personal homepages, web logs, publications and resumes. Another interesting problem to address is leadership discovery in an organization based on email communication history among people. Two heuristic metrics are proposed in [143] for evaluating pair-wise leadership factors in a group of employees. Several issues in discovering the organization's structure through mining leadership graphs constructed from the leadership factors have to be addressed. In both application scenarios, extraction of information from the massive amounts of data is a non-trivial, highly challenging task. Suitable and powerful text classifier methods are needed in order to deal with the huge (and heterogeneous) amount of information data.

Personalization

Personalization is becoming a strong theme, including tasks such as personalized search, automatic taxonomy generation, topic-based document classification and also the spreading Google applications [102]. The personalization can be made explicit or implicit, usually maintaining a user profile [8]. The profiling can be done automatically based on the link structure of a user [146], also referred as topic continuity [98], or on different ranking algorithms. There can be various levels of personalization, from simply detecting the country where a query is being submitted to forcing users to sign up and provide personal information. An important issue with personalization is privacy. A user (or system) must balance the degree of private information they share with the degree to which the service is tailored. For instance, with Google applications personalization is said to continue to be a gradient: the more a user shares, the more tailored the results will be. If one of the authors of this book enters RVM on a search engine, she is not searching for Research Virtual Machines, but almost certainly for Relevance Vector Machines, despite the accuracy of both results. From the above, it is easy to see the enormous challenges faced by text classification systems. Users are becoming increasingly demanding and text classification models will be expected to provide high-quality predictions [47] that are straightforward to use, to provide a better customized service to users.

6.1.2 Information Extraction

Text Categorization-based Ranking

More powerful categorization-based algorithms to enhance the score of Cross-Subject web page lists in search engines should be developed. Traditional ranking algorithms such as Page Rank [18] do not rank web pages as a whole; instead the result is determined for each page individually. The main drawbacks of such approach are related to the fact that only the importance of web pages, the links relation and the category which the web page belongs to are considered. When a page belongs to more than one category traditional ranking computes only the similarity between query keywords and web pages and ignores the page's category. In [47] an algorithm containing diverse information about the web page, the page rank and the category the web page belongs to is said to boost the score of subject web pages. The empirical evidence provided suggests, however, the use of downsized samples in the experiments. Further research is needed to optimize the ranking algorithms able to deal with terabytes of data that are generated every day in applications (and made available on the web), e.g. in medicine, life sciences, physics, engineering and financial markets. The traditional ranking methods are thus suitable for a general search engine, but not for the focused search engine, and the search engines based on categorization should be strengthened.

Structured, Heterogeneous, Multimedia Data

The role of structure in the data is becoming more and more important. A major driving force is the explosive growth in the amount of web heterogeneous data that is also being generated in the business and scientific world [110, 156]. The widespread diffusion of various kernels (and kernel methods) and their ongoing integration led to the development of learning algorithms to cope with abstract data types. Their ability to work in high dimension, to process non-vectorial data, and the natural framework they provide to integrate heterogeneous data are particularly relevant to various problems arising in text classification [3]. In addition, easy-to-use software was applied to structured objects simply by plugging in a suitable kernel function for the data type at hand. Yet research has mainly focused on independent and identically-distributed (i.i.d.) examples.

In most domains there is interesting knowledge to be mined from the relationships between data entities. Extracting this knowledge requires the data to be represented in a form that not only captures the relational information but supports efficient and effective mining of this data and comprehensibility of the resulting knowledge [25]. Dealing with inter-related examples that are linked together in complex graphs or hypergraphs remains a major challenge.

Similarly, multimedia covers for text, like images or videos, are substantially more difficult problems than standard text classification since they are a source of heterogeneous data. Operations on kernels provide simple, powerful tools to integrate heterogeneous data or multiple kernels; this is particularly relevant in text classification, where objects can typically be described by heterogeneous representations, and the availability of a large number of possible kernels for even a single representation raises the question of a choice or combination of kernels. A natural approach with kernel methods is to start by defining one or several kernels for each sort of data. The apparent heterogeneity of data types then vanishes as one simply obtains a family of kernel functions [154].

Dimension Reduction

Today, digital repositories accommodate vast amounts of text in electronic format and the access to it is becoming more necessary and widespread [94]. While differing in volume and characteristics, this data deluge has implications for the way text classification is conducted to meet the needs of individual users and companies. This includes two important issues that we need to highlight: scaling and reducing the dimensionality in vector space based methods for text classification [50, 99].

It has been conjectured that an aggressive dimension reduction may result in a significant loss of information, and therefore result in poor classification results. On the other hand, with dimension reduction computational complexity can be dramatically reduced for all classifiers including support vector machines. Even though the learning ability and computational complexity of training in support vector machines may be independent of the dimension of the feature space, reducing

computational complexity is an essential issue for the efficient handling of a large number of terms in practical applications of text classification [58].

In [89] Wikipedia articles are incorporated into the system to give word distributional representation for documents. The extension with this new corpus causes dimensionality increase. Therefore methods such as latent semantic indexing (LSI), kernel principal component analysis (KPCA) and kernel canonical correlation analysis (KCCA) have been used to tackle this problem.

In addition to low-rank approximation using LSI, other nonlinear techniques for reduction have now acquired a novel interest such as manifold learning where a geometric framework and kernel methods are used [139]. Another example is non-negative matrix factorization [104] which allows graphical visualization and captures the semantics through data rank embedding, which is being increasingly investigated and developed.

Dynamic Mining Large Web Repositories

Web text categorization must adapt to the dynamic change over time as web text documents continue to increase rapidly [49]. Mining large scale repositories requires leveraging both the textual and structural aspects of information data, as well as any relevant metadata. The methods developed in [77] to facilitate the mining of large-scale software repositories are applicable at multiple scales, from single projects to Internet-scale repositories. Combining term-based information retrieval techniques with graphical information derived from program structure resulted in significantly improved software search and retrieval performance. In [79] classifiers are developed to create dynamic category profiles with respect to the document, and accordingly make proper decisions in filtering and classification. To this end, suitable features are selected to construct category profiles to distinguish relevant documents from irrelevant documents. In [57] a novelty-based document clustering method clusters documents based on similarity and novelty. The method assigns higher weights to recent documents and generates clusters with the focus on recent topics. The similarity function is derived probabilistically, extending the conventional cosine measure of the vector space model by incorporating a document forgetting model to produce novelty-based clusters. The clustering method is an incremental update facility, which is applied when new documents are incorporated into a document repository. Dynamic mining of large web repositories is a cutting-edge topic requiring more effective incremental categorization methods to cope with the web dynamic changes. In particular, special focus on automatic reassignment of categories (and hierarchies) needs further research efforts.

6.1.3 Information Distributed Systems

The web is an inherently distributed system. Data and services are available from anywhere to everyone. Nevertheless, users expect transparency from this complex maze and so it is crucial to develop and deploy distributed environments for

researchers and practitioners to better make use of such exciting resources. The challenge, therefore, is to tackle the synergies made possible by linking content, knowledge and learning to make content and knowledge abundant, accessible, interactive and usable over time.

Attempts to distribute several tasks in cluster and grid environments have met with some success [2]. In [61] a parallel evolutionary algorithm applied is applied in an Alchemi framework to optimize the speedup. In [88] large-size combinatorial bioinspired algorithms are deployed in a pool of resources using a Condor grid-enabled framework to select features in a data mining system.

Text mining applications have also seen an initial effort made to take advantage of the available distributed environment resources. In [165] a framework for text mining is proposed and implemented on a web platform. The framework includes a core network, a router, grid members and grid groups. It proceeds by considering four main phases of text mining, viz., text collection phase, feature extraction phase, structure analyzing phase and text classification, which are deployed using a defined protocol between the framework elements.

In [111] a theoretical description of processing textual documents and their classification or clustering is given, and their processing, testing and evaluation of experimental results in a grid environment are described. In addition, the authors find possible ways of improving the results of classification and clustering problems. In [87, 167] a framework is used to provide all the computational facilities needed to solve a text categorization problem, which is split in two stages: constructing the text classifier and classifying new texts. The tasks are dealt with using web services and segmenting the classification task.

Note that other distributed systems, especially distributed databases, are also the working environment of many users, as when a company is spread across a country or the globe for instance. Another very interesting and booming area of distributed applications is that of mobile devices, which presents even more challenges.

These systems can be made feasible with distributed computing systems, provided that there is a serious effort to bring the distributed platforms closer to data miners and researchers in general [141].

Hence, distributed systems will merit a great deal of attention as solutions to speed and scale problems in different applications.

6.2 Conclusion

In previous chapters of the book we dealt with large scale text classification methods and several algorithms and techniques based on inductive inference and kernel machines were approached. The book presents a series of new techniques to enhance, scale and distribute text classification tasks. The approaches combine high performance and robustness with an extensive assessment of the suggested techniques. They have proven to be essential to personal usage for managing (and structuring) documents in an intelligible way.

In this chapter, we gave an overview of how to integrate knowledge into a text classification framework and focused on current trends for dealing with the

overload of digital information using text mining. More specifically, we presented a unified framework whose components incorporate the stages, workflow and procedures needed to further enhance the text classification task. We especially focused on highlighting the conceptual similarities and differences in the various approaches. From a web usage point of view, novel trends have been identified for new challenges in this broad area. Moreover, the links and relations between new avenues of research were embedded in a diamond figure symbolizing the intricate nature of information data and sources.

The potential of applications such as web semantics, text classifiers with multimedia and heterogeneous data, distributed text classifiers frameworks and providing users with personalized content in digital libraries is far from being fully exploited. And so it is our hope that we have provided a unified view of inductive learning techniques in text classification that is valuable and maybe inspiring for different groups of readers: scientists and students, obviously, but also for practitioners looking for solutions in a concrete text classification application. Finally, we would remind readers that the scope of the book only provides an incomplete snapshot of text classification, focusing on some current and most emerging trends in text categorization.

Appendix A
REUTERS-21578

A.1 Introduction

This chapter is a condensed version of the information available about the Reuters-21578 collection.

Reuters-21578 text categorization test collection is a resource for research in information retrieval, machine learning, and other corpus-based research.

The copyright for the text of newswire articles and Reuters annotations in the Reuters-21578 collection resides with Reuters Ltd. and Carnegie Group, Inc. that have agreed to allow the free distribution of this data for research purposes only.

The Reuters-21578, Distribution 1.0 test collection is available from http://www.daviddlewis.com/resources/testcollections/reuters21578

A.2 History

The documents in the Reuters-21578 collection appeared on the Reuters newswire in 1987. The documents were assembled and indexed with categories by personnel from Reuters Ltd. (Sam Dobbins, Mike Topliss, Steve Weinstein) and Carnegie Group, Inc. (Peggy Andersen, Monica Cellio, Phil Hayes, Laura Knecht, Irene Nirenburg) in 1987.

In 1990, the documents were made available by Reuters and CGI for research purposes to the Information Retrieval Laboratory (W. Bruce Croft, Director) of the Computer and Information Science Department at the University of Massachusetts at Amherst. Formatting of the documents and production of associated data files was done in 1990 by David D. Lewis and Stephen Harding at the Information Retrieval Laboratory.

Further formatting and data file production was done in 1991 and 1992 by David D. Lewis and Peter Shoemaker at the Center for Information and Language Studies, University of Chicago. This version of the data was made available for anonymous FTP as "Reuters-22173, Distribution 1.0" in January 1993. From 1993 through 1996, Distribution 1.0 was hosted at a succession of FTP sites maintained by the

Center for Intelligent Information Retrieval (W. Bruce Croft, Director) of the Computer Science Department at the University of Massachusetts at Amherst.

At the ACM SIGIR '96 conference in August, 1996, a group of text categorization researchers discussed how published results on Reuters-22173 could be made more comparable across studies. It was decided that a new version of collection should be produced with less ambiguous formatting, and including documentation carefully spelling out standard methods of using the collection. The opportunity would also be used to correct a variety of typographical and other errors in the categorization and formatting of the collection.

Steve Finch and David D. Lewis did this cleanup of the collection September through November of 1996, relying heavily on Finch's SGML-tagged version of the collection from an earlier study. One result of the re-examination of the collection was the removal of 595 documents which were exact duplicates (based on identity of timestamps down to the second) of other documents in the collection. The new collection therefore has only 21,578 documents, and thus is called the Reuters-21578 collection.

A.3 Formatting

The Reuters-21578 collection is distributed in 22 files. Each of the first 21 files (`reut2-000.sgm` through `reut2-020.sgm`) contain 1000 documents, while the last (`reut2-021.sgm`) contains 578 documents.

The files are in SGML format. Rather than going into the details of the SGML language, it is described how the SGML tags are used to divide each file, and each document, into sections.

Each of the 22 files begins with a document type declaration line:
`<!DOCTYPE lewis SYSTEM "lewis.DTD">`

A.4 The REUTERS Tag

Each article starts with an "open tag" of the form:
`<REUTERS TOPICS=?? LEWISSPLIT=?? CGISPLIT=?? OLDID=?? NEWID=??>`
where the `??` are filled in an appropriate fashion. Each article ends with a "close tag" of the form:`</REUTERS>`

In all cases the `<REUTERS>` and `</REUTERS>` tags are the only items on their line.

Each `REUTERS` tag contains explicit specifications of the values of five attributes, `TOPICS, LEWISSPLIT, CGISPLIT, OLDID,` and `NEWID`. These attributes are meant to identify documents and groups of documents, and have the following meanings:

1. TOPICS: The possible values are YES, NO and BYPASS:
   a. YES: indicates that in the original data there was at least one entry in the TOPICS fields;
   b. NO: indicates that in the original data the story had no entries in the TOPICS field;
   c. BYPASS: indicates that in the original data the story was marked with the string BYPASS (or a typographical variant on that string).

   This poorly-named attribute unfortunately is the subject of much confusion. It is meant to indicate whether or not the document had TOPICS categories in the raw Reuters-22173 dataset. The sole use of this attribute is to defining training set splits similar to those used in previous research. (See the section on training set splits.) The TOPICS attribute does not indicate anything about whether or not the Reuters-21578 document has any TOPICS categories. That can be determined by actually looking at the TOPICS field. A story with TOPICS="YES" can have no TOPICS categories, and a story with TOPICS="NO" can have TOPICS categories.

   A reasonable (though not certain) assumption is that for all TOPICS="YES" stories the indexer at least thought about whether the story belonged to a valid TOPICS category. Thus, the TOPICS="YES" stories with no topics can reasonably be considered negative examples for all 135 valid TOPICS categories.

   TOPICS="NO" stories are more problematic in their interpretation. Some of them presumedly result because the indexer made an explicit decision that they did not belong to any of the 135 valid TOPICS categories. However, there are many cases where it is clear that a story should belong to one or more TOPICS categories, but for some reason the category was not assigned. There appear to be certain time intervals where large numbers of such stories are concentrated, suggesting that some parts of the data set were simply not indexed, or not indexed for some categories or category sets. Also, in a few cases, the indexer clearly meant to assign TOPICS categories, but put them in the wrong field. These cases have been corrected in the Reuters-21578 data, yielding stories that have TOPICS categories, but where TOPICS="NO", because the the category was not assigned in the raw version of the data.

   "BYPASS" stories clearly were not indexed, and so are useful only for general distributional information on the language used in the documents.
2. LEWISSPLIT: The possible values are TRAINING, TEST, and NOT-USED. TRAINING indicates it was used in the training set in the experiments reported in [66, 67, 68, 71]. TEST indicates it was used in the test set for those experiments, and NOT-USED means it was not used in those experiments.
3. CGISPLIT: The possible values are TRAINING-SET and PUBLISHED-TESTSET indicating whether the document was in the training set or the test set for the experiments reported in [42, 43].
4. OLDID: The identification number (ID) the story had in the Reuters-22173 collection.

5. NEWID: The identification number (ID) the story has in the Reuters-21578, Distribution 1.0 collection. These IDs are assigned to the stories in chronological order.

In addition, some REUTERS tags have a sixth attribute, CSECS, which can be ignored.

The use of these attributes is critical to allowing comparability between different studies with the collection.

A.5 Document-Internal Tags

Just as the <REUTERS> and </REUTERS> tags serve to delimit documents within a file, other tags are used to delimit elements within a document. These are discussed in the order in which they typically appear, though the exact order should not be relied upon in processing. In some cases, additional tags occur within an element delimited by these top level document-internal tags. These are discussed in this section as well.

It is specified below whether each open/close tag pair is used exactly once (ONCE) per a story, or a variable (VARIABLE) number of times (possibly zero). In many cases the start tag of a pair appears only at the beginning of a line, with the corresponding end tag always appearing at the end of the same line. When this is the case, it is indicated it with the notation "SAMELINE" below, as an aid to those processing the files without SGML tools.

1. <DATE>, </DATE> [ONCE, SAMELINE]: Encloses the date and time of the document, possibly followed by some non-date noise material.
2. <MKNOTE>, </MKNOTE> [VARIABLE]: Notes on certain hand corrections that were done to the original Reuters corpus by Steve Finch.
3. <TOPICS>, </TOPICS> [ONCE, SAMELINE]: Encloses the list of TOPICS categories, if any, for the document. If TOPICS categories are present, each will be delimited by the tags <D> and </D>.
4. <PLACES>, </PLACES> [ONCE, SAMELINE]: Same as <TOPICS> but for PLACES categories.
5. <PEOPLE>, </PEOPLE> [ONCE, SAMELINE]: Same as <TOPICS> but for PEOPLE categories.
6. <ORGS>, </ORGS> [ONCE, SAMELINE]: Same as <TOPICS> but for ORGS categories.
7. <EXCHANGES>, </EXCHANGES> [ONCE, SAMELINE]: Same as <TOPICS> but for EXCHANGES categories.
8. <COMPANIES>, </COMPANIES> [ONCE, SAMELINE]: These tags always appear adjacent to each other, since there are no COMPANIES categories assigned in the collection.
9. <UNKNOWN>, </UNKNOWN> [VARIABLE]: These tags bracket control characters and other noisy and/or somewhat mysterious material in the Reuters stories.

10. <TEXT>, </TEXT> [ONCE]: There was an attempt to delimit all the textual material of each story between a pair of these tags. Some control characters and other "junk" material may also be included. The whitespace structure of the text has been preserved. The <TEXT> tag has the following attributes:

    a. TYPE: This has one of three values: NORM, BRIEF, and UNPROC. NORM is the default value and indicates that the text of the story had a normal structure. In this case the TEXT tag appears simply as <TEXT>. The tag appears as <TEXT TYPE="BRIEF"> when the story is a short one or two line note. The tags appears as <TEXT TYPE="UNPROC"> when the format of the story is unusual in some fashion that limited our ability to further structure it.

       The following tags optionally delimit elements inside the TEXT element. Not all stories will have these tags:

    b. <AUTHOR>, </AUTHOR>: Author of the story.
    c. <DATELINE>, </DATELINE>: Location the story originated from, and day of the year.
    d. <TITLE>, </TITLE>: Title of the story. An attempt to capture the text of stories with TYPE="BRIEF" within a <TITLE> element.
    e. <BODY>, </BODY>: The main text of the story.

A.6 Categories

A test collection for text categorization contains, at minimum, a set of texts and, for each text, a specification of what categories that text belongs to. For the Reuters-21578 collection the documents are Reuters newswire stories, and the categories are five different sets of content related categories, as shown in table A.1. For each document, a human indexer decided which categories from which sets that document belonged to.

The TOPICS categories are economic subject categories. Examples include "coconut", "gold", "inventories", and "money-supply". This set of categories is the one that has been used in almost all previous research with the Reuters data.

The EXCHANGES, ORGS, PEOPLE, and PLACES categories correspond to named entities of the specified type. Examples include "nasdaq" (EXCHANGES),

Table A.1 Category types in Reuters-21578.

Type	Number	+ 1 occurrence	+20 occurrences
EXCHANGES	39	32	7
ORGS	56	32	9
PEOPLE	267	114	15
PLACES	175	147	60
TOPICS	135	120	57

”gatt” (ORGS), ”perez-de-cuellar” (PEOPLE), and ”australia” (PLACES). Typically a document assigned to a category from one of these sets explicitly includes some form of the category name in the document’s text. (Something which is usually not true for TOPICS categories.)

Reuters-21578, Distribution 1.0 includes five files (all-exchanges-strings.lc.txt, all-orgs-strings.lc.txt, all-people-strings.lc.txt, all-places-strings.lc.txt, and all-topics-strings.lc.txt) which list the names of all legal categories in each set. A sixth file, cat-descriptions_120396.txt gives some additional information on the category sets.

Note that a sixth category field, COMPANIES, was present in the original Reuters materials distributed by Carnegie Group, but no company information was actually included in these fields. In the Reuters-21578 collection this field is always empty.

In the table above it can be seen how many categories appear in at least 1 of the 21,578 documents in the collection, and how many appear at least 20 of the documents. Many categories appear in no documents, but researchers are encouraged to include these categories when evaluating the effectiveness of their categorization system.

Additional details of the documents, categories, and corpus preparation process appear in [67], and at greater length in [66].

A.7 Using Reuters-21578 for Text Categorization Research

In testing a method for text categorization it is important that knowledge of the nature of the test data not unduly influence the development of the system, or the performance obtained will be unrealistically high. One way of dealing with this is to divide a set of data into two subsets: a training set and a test set. An experimenter then develops a categorization system by automated training on the training set only, and/or by human knowledge engineering based on examination of the training set only. The categorization system is then tested on the previously unexamined test set.

Effectiveness results can only be compared between studies that the same training and test set (or that use cross-validation procedures). One problem with the Reuters-22173 collection was that the ambiguity of formatting and annotation led different researchers to use different training/test divisions. This was particularly problematic when researchers attempted to remove documents that ”had no TOPICS”, as there were several definitions of what this meant.

To eliminate these ambiguities from the Reuters-21578 collection, it is defined exactly which articles are in each of the recommended training sets and test sets by specifying the values those articles will have on the TOPICS, LEWISSPLIT, and CGISPLIT attributes of the REUTERS tags. It is strongly encouraged that all studies on Reuters-21578 use one of the following training test divisions (or use multiple random splits, e.g. cross-validation).

A.7.1 The Modified Lewis ("ModLewis") Split

This split replaces the 14704/6746 split (723 unused) of the Reuters-22173 collection, which was used in [66, 67, 68, 71].

Table A.2 presents the splitting procedure. If `LEWISSPLIT` is `NOT-USED` or `TOPICS` is `BYPASS`, a document is not used.

The duplicate documents removed in forming Reuters-21578 are of course not present. The documents with TOPICS="BYPASS" are not used, since subsequent analysis strongly indicates that they were not categorized by the indexers. The 1,765 unused documents should not be tested on and should not be used for supervised learning. However, they may useful as additional information on the statistical distribution of words, phrases, and other features that might used to predict categories.

This split assigns documents from April 7, 1987 and before to the training set, and documents from April 8, 1987 and after to the test set. Table A.3 presents the number of documents used on ModLewis split.

Given the many changes in going from Reuters-22173 to Reuters-21578, including correction of many typographical errors in category labels, results on the ModLewis split cannot be compared with any published results on the Reuters-22173 collection.

Table A.2 ModLewis split.

Set	LEWISSPLIT	TOPICS
Train	TRAIN	YES ou NO
Test	TEST	Yes ou NO
Not Used	NOT-USED	-
Not Used	-	BYPASS

Table A.3 Used documents on ModLewis split.

Set	Documents
Train	13625
Test	6188
Nou used	1765

A.8 The Modified Apte ("ModApte") Split

This replaces the 10645/3672 split (7,856 not used) of the Reuters-22173 collection. It is an approximation to the training and test splits used in [6] and [7].

Table A.4 presents the split. As with the ModLewis, those documents removed in forming Reuters-21578 are not present, and BYPASS documents are not used.

The intent in [6] and [7] to use the Lewis split, but restrict it to documents with at least one `TOPICS` categories. However, but it was not clear exactly what

Table A.4 ModApte split.

Set	LEWISSPLIT	TOPICS
Train	TRAIN	YES
Test	TEST	YES
Not Used	NOT-USED	YES
Not Used	-	NO
Not Used	-	BYPASS

Apte, et al meant by having at least one TOPICS category (e.g. how was "bypass" treated, whether this was before or after any fixing of typographical errors, etc.). This interpretation is encoded in the TOPICS attribute.

As discussed above, some TOPICS="YES" stories have no TOPICS categories, and a few TOPICS="NO" stories have TOPICS categories. These facts are irrelevant to the definition of the split.

If you are using a learning algorithm that requires each training document to have at least TOPICS category, you can screen out the training documents with no TOPICS categories.

Please do NOT screen out any of the 3,299 documents - that will make your results incomparable with other studies.

As with ModLewis, it may be desirable to use the 8,676 unused documents for gathering statistical information about feature distribution.

As with ModLewis, this split assigns documents from April 7, 1987 and before to the training set, and documents from April 8, 1987 and after to the test set. The difference is that only documents with at least one TOPICS category are used. The rationale for this restriction is that while some documents lack TOPICS categories because no TOPICS apply (i.e. the document is a true negative example for all TOPICS categories), it appears that others simply were never assigned TOPICS categories by the indexers. (Unfortunately, the amount of time that has passed since the collection was created has made it difficult to establish exactly what went on during the indexing.)

Table A.5 presents the number of documents used on ModApte split.

Given the many changes in going from Reuters-22173 to Reuters-21578, including correction of many typographical errors in category labels, results on the ModApte split cannot be compared with any published results on the Reuters-22173 collection.

Table A.5 Used documents on ModApte split.

Set	Documents
Train	9603
Test	3299
Not used	8676

```
<REUTERS TOPICS="YES" LEWISSPLIT="TRAIN" CGISPLIT="TRAINING-SET"
OLDID="5552" NEWID="9">
<DATE>26-FEB-1987 15:17:11.20</DATE>
<TOPICS><D>earn</D></TOPICS>
<PLACES><D>usa</D></PLACES>
<PEOPLE></PEOPLE>
<ORGS></ORGS>
<EXCHANGES></EXCHANGES>
<COMPANIES></COMPANIES>
<UNKNOWN>
&#5;&#5;&#5;F
&#22;&#22;&#1;f0762&#31;reute
r f BC-CHAMPION-PRODUCTS-&lt;CH   02-26 0067</UNKNOWN>
<TEXT>&#2;
<TITLE>CHAMPION PRODUCTS &lt;CH> APPROVES STOCK SPLIT</TITLE>
<DATELINE>    ROCHESTER, N.Y., Feb 26 - </DATELINE><BODY>Champion
Products Inc said its
board of directors approved a two-for-one stock split of its
common shares for shareholders of record as of April 1, 1987.
    The company also said its board voted to recommend to
shareholders at the annual meeting April 23 an increase in the
authorized capital stock from five mln to 25 mln shares.
 Reuter
&#3;</BODY></TEXT>
</REUTERS>
```

Fig. A.1 Example of a Reuters-21578 document.

A.9 Stopwords

a; about; above; across; after; afterwards; again; against; all; almost; alone; along; already; also; although; always; am; among; amongst; amoungst; amount; an; and; another; any; anyhow; anyone; anything; anyway; anywhere; are; around; as; at; back; be; became; because; become; becomes; becoming; been; before; beforehand; behind; being; below; beside; besides; between; beyond; bill; both; bottom; but; by; call; can; cannot; cant; co; computer; con; could; couldnt; cry; de; describe; detail; do; done; down; due; during; each; eg; eight; either; eleven; else; elsewhere; empty; enough; etc; even; ever; every; everyone; everything; everywhere; except; few; fifteen; fify; fill; find; fire; first; five; for; former; formerly; forty; found; four; from; front; full; further; get; give; go; had; has; hasnt; have; he; hence; her; here; hereafter; hereby; herein; hereupon; hers; herself; him; himself; his; how; however; hundred; i; ie; if; in; inc; indeed; interest; into; is; it; its; itself; keep; last; latter; latterly; least; less; ltd; made; many; may; me; meanwhile; might; mill; mine; more; moreover; most; mostly; move; much; must; my; myself; name; namely; neither; never; nevertheless; next; nine; no; nobody; none; noone; nor; not; nothing; now; nowhere; of; off; often; on; once; one; only; onto; or; other; others; otherwise; our; ours; ourselves; out; over; own; part; per; perhaps; please; put; rather; re; same; see; seem; seemed; seeming; seems; serious; several; she; should; show; side; since; sincere; six; sixty; so; some; somehow; someone; something; sometime; sometimes; somewhere; still; such; system; take; ten; than; that; the; their; them; themselves;

then; thence; there; thereafter; thereby; therefore; therein; thereupon; these; they; thick; thin; third; this; those; though; three; through; throughout; thru; thus; to; together; too; top; toward; towards; twelve; twenty; two; un; under; until; up; upon; us; very; via; was; we; well; were; what; whatever; when; whence; whenever; where; whereafter; whereas; whereby; wherein; whereupon; wherever; whether; which; while; whither; who; whoever; whole; whom; whose; why; will; with; within; without; would; yet; you; your; yours; yourself; yourselves;

Appendix B
RCV1 - Reuters Corpus Volume I

B.1 Introduction

This chapter is a rather condensed version of the information available about the Reuters Corpus Volume I (RCV1) collection. RCV1 is an archive of over 800,000 manually categorized newswire stories using three category sets, that was recently made available by Reuters Ltd. for research purposes. Use of this data for research on text categorization requires a detailed understanding of the real world constraints under which the data was produced. RCV1 as distributed is simply a collection of newswire stories, not a test collection. It includes known errors in category assignment, provides lists of category descriptions that are not consistent with the categories assigned to articles, and lacks essential documentation on the intended semantics of category assignment [107]. Nevertheless it constitutes a valuable research tool for text, namely for categorization.

Reuters Ltd. has gone through significant restructuring since RCV1 was produced, and information that in a research setting would have been retained was therefore not recorded. In particular, no formal specification remains of the coding practices at the time the RCV1 data was produced. Fortunately, several researchers [107, 73] have examined the available information and, by combining related documentation and interviews with Reuters personnel, have largely reconstructed those aspects of coding relevant to text categorization research.

B.2 The Documents

Reuters Ltd. is the largest international text and television news agency. Its editorial division produces some 11,000 stories a day in 23 languages [73]. Stories are both distributed in real time and made available via online databases and other archival products.

The RCV1 dataset was created from one of those online databases. It consists of the English language stories produced by Reuters journalists between August 20, 1996, and August 19, 1997. A researcher can obtain the data on two CD-ROMs,

formatted in XML, by submiting a request to the National Institute of Science and Technology[1]. Figure B.1 shows an example story with some simplification of the markup for brevity, taken from [73].

RCV1 contains 35 times as many documents (806,791) as the popular Reuters-21578 collection (see Appendix A), making it one of the largest available text categorization test collection. Moreover, RCV1 is also more organized than previous collections. Each document is in a separate file and has a unique ID, ranging from 2286 to 810597 with some gaps. The ID order does not correspond to chronological order of the stories, but they have time stamps that give only the day, not the time, since the stories were taken from an archival database, not from the original stream sent out over the newswire.

Both text and metadata are formatted with XML, simplifying their use. As RCV1 was produced from an archival database, it has fewer alerts, corrections to previous stories, and other peculiarities.

RCV1 contains all or almost all stories of a particular type from an interval of one year. For temporal studies, this is a major advantage over Reuters-21578, which had uneven coverage of a fraction of a year.

The corpus has a limited number of duplicates, foreign language documents, and other similar issues, which can be problematic depending on the application at hand, but are comparable to to levels seen in operational settings.

B.3 The Categories

To aid retrieval from database products category codes from three sets (Topics, Industries, and Regions) were assigned to stories.

B.3.1 Topic Codes

In the experiments carried out in this thesis categories were taken from the topic codes, which were assigned to capture the major subjects of a story. They were organized in four hierarchical groups: CCAT (Corporate/Industrial), ECAT (Economics), GCAT (Government/Social), and MCAT(Markets).

There are 103 categories actually assigned to the data. In the second column of Table B.1 (Documents) the total number of positive documents for the ten most populated classes of RCV1 is given. For training, RCV1 defines 22,370 documents that should be used. For testing we selected the first 50,000 documents not used for training. Table B.1 also presents the effective number of positive documents used for training and testing.

One can see that this code set shows particular perspective on a data set. The RCV1 articles span a broad range of content, but the code set only emphasizes distinctions relevant to Reuters customers. For instance,there are three different Topic codes for corporate ownership changes, but all of science and technology is a single category (GSCI) [72].

[1] http://trec.nist.gov/data/reuters/reuters.html

```
<?xml version="1.0" encoding="iso-8859-1" ?>
<newsitem itemid="2330" id="root" date="1996-08-20" xml:lang="en">
<title>USA: Tylan stock jumps; weighs sale of company.</title>
<headline>Tylan stock jumps; weighs sale of company.</headline>
<dateline>SAN DIEGO</dateline>
<text>
<p>The stock of Tylan General Inc. jumped Tuesday after the maker of
process-management equipment said it is exploring the sale of the
company and added that it has already received some inquiries from
potential buyers.</p>
<p>Tylan was up $2.50 to $12.75 in early trading on the Nasdaq market.</p>
<p>The company said it has set up a committee of directors to oversee
the sale and that Goldman, Sachs &amp;amp; Co. has been retained as its
financial adviser.</p>
</text>
<copyright>(c) Reuters Limited 1996</copyright>
<metadata>
<codes class="bip:countries:1.0">
<code code="USA"> </code>
</codes>
<codes class="bip:industries:1.0">
<code code="I34420"> </code>
</codes>
<codes class="bip:topics:1.0">
<code code="C15"> </code>
<code code="C152"> </code>
<code code="C18"> </code>
<code code="C181"> </code>
<code code="CCAT"> </code>
</codes>
<dc element="dc.publisher" value="Reuters Holdings Plc"/>
<dc element="dc.date.published" value="1996-08-20"/>
<dc element="dc.source" value="Reuters"/>
<dc element="dc.creator.location" value="SAN DIEGO"/>
<dc element="dc.creator.location.country.name" value="USA"/>
<dc element="dc.source" value="Reuters"/>
</metadata>
</newsitem>
```

Fig. B.1 Example of a RCV1 document.

Table B.1 Positive documents for RCV1 categories.

Category	Documents	Training	Testing
CCAT	381327	10416	23077
GCAT	239267	2050	5180
MCAT	204820	5154	11110
C15	151785	3122	7454
ECAT	119920	3162	7539
M14	85440	1799	4887
C151	81890	515	698
C152	73092	1088	435
GPOL	56878	1627	3756
M13	53634	1095	2613

B.3.1.1 Industry Codes

Industry codes were assigned based on types of businesses discussed in the story. They were grouped in 10 subhierarchies,such as I2(METALSANDMINERALS) and I5(CONSTRUCTION). The Industry codes make up the largest of the three code sets, supporting many fine distinctions.

B.3.1.2 Region Codes

Region codes included both geographic locations and economic/political groupings. No hierarchical taxonomy was defined.

B.3.2 Coding Policy

Explicit policies on code assignment presumedly increase consistency and usefulness of coding, though coming up with precise policies is difficult. Reuters guidance for coding included two broad policies, among others. In [73], these policies are described as:

1. Minimum Code Policy: Each story was required to have at least one Topic code and one Region code;
2. Hierarchy Policy: Coding was to assign the most specific appropriate codes from the Topic and Industry sets, as well as (usually automatically) all ancestors of those codes. In contrast to some coding systems, there was no limit on the number of codes with the same parent that could be applied.

B.4 Stopwords

For comparison purposes, the same set of stopwords used for Reuters-21578 (see Appendix A) was filtered out from RCV1.

References

1. Aas, K., Eikvil, L.: Text categorisation: A survey. Technical report, Norwegian Computing Center (1999)
2. Abdalhaq, B., Cortés, A., Margalef, T., Bianchini, G., Luque, E.: Between classical and ideal: Enhancing wildland fire prediction using cluster computing. Cluster Computing 9(3), 329–343 (2006)
3. Abernethy, J., Bach, F., Evgeniou, T., Vert, J.: A new approach to collaborative filtering: Operator estimation with spectral regularization. Journal of Machine Learning Research 10, 803–826 (2009)
4. Al-Ani, A., Deriche, M.: Feature selection using a mutual information based measure. In: International Conference on Pattern Recognition - ICPR 2002, vol. 4, pp. 82–85 (2002)
5. Androutsopoulos, I., Koutsias, J., Chandrinos, K., Spyropoulos, C.: An experimental comparison of naïve bayesian and keyword-based anti-spam filtering with personal e-mail messages. In: International Conference on Research and Development in InformationRetrieval - ACM SIGIR 2000, pp. 160–167 (2000)
6. Apté, C., Damerau, F., Weiss, S.: Automated learning of decision rules for text categorization. ACM Transactions for Information Systems 12, 233–251 (1994)
7. Apté, C., Damerau, F., Weiss, S.: Toward language independent automated learning of text categorization models. In: International Conference on Research and Development in InformationRetrieval- ACM SIGIR 1994, pp. 23–30 (1994)
8. Armano, G., Manconi, A., Vargiu, E.: A multiagent system for retrieving bioinformatics publications from web sources. IEEE Transactions on NanoBioscience 6(2), 104–109 (2007)
9. Baeza-Yates, R., Ribeiro-Neto, B.: Modern Information Retrieval. ACM Press, New York (1999)
10. Baker, L., McCallum, A.: Distributional clustering of words for text classification. In: International Conference on Research and Development in Information Retrieval - ACM SIGIR 1998, pp. 96–103 (1998)
11. Bauer, E., Kohavi, R.: An empirical comparison of voting classification algorithms: Bagging, boosting, and variants. Machine Learning 36(1–2), 105–139 (1999)
12. Benali, F., Ubeda, S., Legrand, V.: Collaborative approach to automatic classification of heterogeneous information security. In: Second International Conference on Emerging Security Information, Systems and Technologies, pp. 294–299 (2008)
13. Bishop, C.: Pattern Recognition and Machine Learning. Springer, Heidelberg (2006)

14. Bishop, C., Tipping, M.: Bayesian Regression and Classification. In: Suykens, j., Horvath, G., Basu, S., Micchelli, C., Vandewalle, J. (eds.) Advances in Learning Theory: Methods, Models and Applications. NATO Science Series III: Computer and Systems Sciences, vol. 190, pp. 267–285. IOS Press, Amsterdam (2003)
15. Błażewicz, J., Ecker, K., Pesch, E., Schmidt, G., Weglarz, J.: Scheduling Computer and Manufacturing Processes. Springer, Heidelberg (2001)
16. Boser, B., Guyon, I., Vapnik, V.: A training algorithm for optimal margin classifiers. In: ACM Annual Workshop on Computational Learning Theory - COLT 1992, pp. 144–152 (1992)
17. Breiman, L.: Bagging predictors. Machine Learning 24(2), 123–140 (1996)
18. Brin, S., Page, L.: The anatomy of a large-scale hypertextual web search engine. In: Computer Networks and ISDN Systems, pp. 107–117 (1998)
19. Chin, O., Kulathuramaiyer, N., Yeo, A.: Automatic discovery of concepts from text. In: IEEE/WIC/ACM International Conference on Web Intelligence WI 2006, pp. 1046–1049 (December 2006)
20. Clack, C., Farringdon, J., Lidwell, P., Yu, T.: Autonomous document classification for business. In: International Conference on Autonomous Agents, pp. 201–208 (1997)
21. Cohen, W.: Learning to Classify English Text with ILP Methods. In: De Raedt, L. (ed.) Advances in Inductive Logic Programming, pp. 124–143. IOS Press, Amsterdam (1995)
22. Cohen, W., Hirsh, H.: Joins that generalize: Text classification using WHIRL. In: International ACM Conference on Knowledge Discovery and Data Mining - KDD 1998, pp. 169–173 (1998)
23. Cohen, W., Singer, Y.: Context-sensitive learning methods for text categorization. ACM Transactions on Information Systems 17(2), 141–173 (1999)
24. Cohn, D., Atlas, L., Ladner, R.: Improving generalization with active learning. Machine Learning 15(2), 201–221 (1994)
25. Cook, D., Holder, L. (eds.): Mining Graph Data. John Wiley & Sons, Chichester (2007)
26. Cooley, R.: Classification of news stories using support vector machines. In: International Joint Conference on Artificial Intelligence, Text Mining Workshop - IJCAI 1999 (1999)
27. Dagan, I., Engelson, S.: Committee-based sampling for training probabilistic classifiers. In: International Conference on Machine Learning - ICML 1995, pp. 150–157 (1995)
28. Dagan, I., Karov, Y., Roth, D.: Mistake driven learning in text categorization. In: Conference on Empirical Methods in Natural Language Processing - EMNLP 1997, pp. 55–63 (1997)
29. Deerwester, S., Dumais, S., Furnas, G., Landauer, T., Harshman, R.: Indexing by latent semantic indexing. Journal of the American Society of Information Science 41(6), 391–407 (1990)
30. Drucker, H., Vapnik, V., Wu, D.: Automatic text categorization and its applications to text retrieval. IEEE Transactions on Neural Networks 10(5), 1048–1054 (1999)
31. Duda, R., Hart, P.: Pattern Classification and Scene Analysis. John Wiley & Sons, Chichester (1993)
32. Dumais, S., Chen, H.: Hierarchical classification of Web content. In: Belkin, N.J., Ingwersen, P., Leong, M.-K. (eds.) International Conference on Research and Development in Information Retrieval - ACM SIGIR 2000, pp. 256–263 (2000)
33. Dumais, S., Platt, J., Heckerman, D., Sahami, M.: Inductive learning algorithms and representations for text categorization. In: International Conference on Information and Knowledge Management - ICIKM 1998, pp. 148–155. ACM Press, New York (1998)

34. Efron, B., Tibshirani, R.: An Introduction to the Bootstrap. Chapman & Hall, Boca Raton (1993)
35. European Commission. ICT - information and communication technologies - work programme 2007-2008 (2007)
36. Eyheramendy, S., Genkin, A., Ju, W., Lewis, D., Madigan, D.: Sparse bayesian classifiers for text categorization. Technical report, Department of Statistics, Rutgers University (2003)
37. Fawcett, T.: Roc graphs: Notes and practical considerations for data mining researchers. Technical Report HPL-2003-4, HP Laboratories (2004)
38. Freund, Y., Schapire, R.: Experiments with a new boosting algorithm. In: International Conference on Machine Learning - ICML 1996, pp. 148–156 (1996)
39. Freund, Y., Seung, H., Shamir, E., Tishby, N.: Information, Prediction, and Query by Committee. In: Advances in Neural Information Processing Systems - NIPS 1993, pp. 483–490. Morgan Kaufmann Publishers Inc, San Francisco (1993)
40. Gale, W., Church, W., Yarowski, D.: A method for disambiguating word senses in a large corpus. Computers and the Humanities 26(5), 415–439 (1992)
41. Godbole, S.: Inter-class Relationships in Text Classification. PhD thesis, Indian Institute of Technology (2006)
42. Hayes, P., Anderson, P., Nirenburg, I., Schmandt, L.: TCS: A shell for content-based text categorization. In: IEEE Conference on Artificial Intelligence Applications, pp. 320–326. IEEE Press, Los Alamitos (1990)
43. Hayes, P., Weinstein, S.: CONSTRUE/TIS: A system for content-based indexing of a database of news stories. In: Conference on Innovative Applications of Artificial Intelligence - IAAI 1990, pp. 49–64 (1990)
44. Haykin, S.: Neural Networks: A Comprehensive Foundation. McGraw Hill, New York (2000)
45. Hotho, A., Nürnberger, A., Paass, G.: A brief survey of text mining. LDV Forum 20(1), 19–62 (2005)
46. Hu, X., Zhang, X., Lu, C., Park, E., Zhou, X.: Exploiting wikipedia as external knowledge for document clustering. In: Knowledge Discovery and Data Mining (KDD 2009), pp. 389–396 (2009)
47. Huang, J., Wang, G., Wang, Z.: Cross-subject page ranking based on text categorization. In: International Conference on Information and Automation, pp. 363–368 (2008)
48. Hull, D.: Improving text retrieval for the routing problem using latent semantic indexing. In: International Conference on Research and Development in Information Retrieval - ACM SIGIR 1994, pp. 282–289 (1994)
49. Jia, Z., Hu, M., Song, H., Hong, L.: Web text categorization for enterprise decision support based on SVMs: An application of GBODSS. In: ISNN 2009. LNCS, vol. 5552, pp. 753–762. Springer, Heidelberg (2009)
50. Jo, T., Yeom, G.: List based matching algorithm for classifying news articles in newspage.com. In: IEEE International Conference on System of Systems Engineering, pp. 1–5 (2008)
51. Joachims, T.: A probabilistic analysis of the rocchio algorithm with tfidf for text categorization. In: International Conference on Machine Learning - ICML 1997, pp. 143–151 (1997)
52. Joachims, T.: Text categorization with support vector machines: Learning with many relevant features. In: Nédellec, C., Rouveirol, C. (eds.) ECML 1998. LNCS, vol. 1398, pp. 137–142. Springer, Heidelberg (1998)
53. Joachims, T.: Transductive inference for text classification using support vector machines. In: International Conference on Machine Learning - ICML 1999, pp. 200–209. Morgan Kaufmann Publishers, San Francisco (1999)

54. Joachims, T.: Learning to Classify Text Using Support Vector Machines. Kluwer Academic Publishers, Dordrecht (2001)
55. Johansson, R., Nugues, P.: Sparse bayesian classification of predicate arguments. In: Conference on Computational Natural Language Learning - CoNLL 2005, pp. 177–180 (2005)
56. John, G., Kohavi, R., Peleger, K.: Irrelevant features and the subset selection problem. In: International Conference on Machine Learning - ICML 1994, pp. 121–129 (1994)
57. Khy, S., Ishikawa, Y., Kitagawa, H.: A novelty-based clustering method for on-line documents. World Wide Web 11(1), 1–37 (2008)
58. Kim, H., Howland, P., Park, H.: Dimension reduction in text classification with support vector machines. Journal of Machine Learning Research (JLMR) 6, 37–53 (2005)
59. König, A., Brill, E.: Reducing the human overhead in text categorization. In: International ACM Conference on Knowledge Discovery and Data Mining - KDD 2006, pp. 598–603 (2006)
60. Kuncheva, L.: Combining Pattern Classifiers - Methods and Algorithms. Wiley, Chichester (2004)
61. Kuś, W.: Evolutionary optimization of forging anvils using grid based on alchemi framework. In: IEEE International Conference on e-Science and Grid Computing, pp. 121–125 (2006)
62. Kwok, J.: Automated text categorization using support vector machine. In: International Conference on Neural Information Processing - ICONI 1998, pp. 347–351 (1998)
63. Larkey, L.: A patent search and classification system. In: ACM Conference on Digital Libraries - DL 1999, pp. 179–187 (1999)
64. Larkey, L., Croft, W.: Combining classifiers in text categorization. In: International Conference on Research and Development in Information Retrieval - ACM SIGIR 1996, pp. 289–297 (1996)
65. Lee, C., Chiu, H., Yang, H.: A platform of biomedical literature mining for categorization of cancer related abstracts. In: Second International Conference on Innovative Computing, Information and Control, p. 174. IEEE Computer Society Press, Los Alamitos (2007)
66. Lewis, D.: Representation and Learning in Information Retrieval. PhD thesis, Computer Science Department; University of Massachusetts (1991)
67. Lewis, D.: An evaluation of phrasal and clustered representations on a text categorization. In: International Conference on Research and Development in Information Retrieval - ACM SIGIR 1992, pp. 37–50 (1992)
68. Lewis, D.: Feature selection and feature extraction for text categorization. In: Speech and Natural Language Workshop, pp. 212–217 (1992)
69. Lewis, D., Catlett, J.: Heterogeneous uncertainty sampling for supervised learning. In: International Conference on Machine Learning - ICML 1994, pp. 148–156 (1994)
70. Lewis, D., Gale, W.: A sequential algorithm for training text classifiers. In: International Conference on Research and Development in InformationRetrieval - ACM SIGIR 1994, pp. 3–12 (1994)
71. Lewis, D., Ringuette, M.: A comparison of two learning algorithms for text categorization. In: Symposium on Document Analysis and Information Retrieval, pp. 81–93. University of Nevada (1994)
72. Lewis, D., Schapire, R., Callan, J., Papka, R.: Training algorithms for linear text classifiers. In: International Conference on Research and Development in Information Retrieval - ACM SIGIR 1996, pp. 298–306. ACM Press, New York (1996)
73. Lewis, D., Yang, Y., Rose, T., Li, F.: Rcv1: A new benchmark collection for text categorization research. Journal of Machine Learning Research 5, 361–397 (2004)

74. Li, H., Yamanishi, K.: Text classification using esc-based stochastic decision lists. In: International ACM Conference on Information and Knowledge Management - CIKM 1999, pp. 122–130 (1999)
75. Li, Y., Jain, A.: Classification of text documents. Computing Journal 41(8), 537–546 (1998)
76. Liere, R., Tadepalli, P.: Active learning with committees for text categorization. In: Conference of the American Association for Artificial Intelligence - AAAI 1997, pp. 591–596 (1997)
77. Linstead, E., Rigor, P., Bajracharya, S., Lopes, C., Baldi, P.: Mining internet-scale software repositories. In: Platt, J.C., Koller, D., Singer, Y., Roweis, S. (eds.) Advances in Neural Information Processing Systems, vol. 20, pp. 929–936. MIT Press, Cambridge (2008)
78. Liu, B., Dai, Y., Li, X., Lee, W., Yu, P.: Building text classifiers using positive and unlabeled examples. In: IEEE International Conference on Data Mining - ICDM 2003, pp. 179–188. IEEE Computer Society, Los Alamitos (2003)
79. Liu, R.: Dynamic category profiling for text filtering and classification. Information Processing and Management: an International Journal 43(1), 154–168 (2007)
80. Lotrič, U., Silva, C., Ribeiro, B., Dobnikar, A.: Modeling execution times of data mining problems in grid environment. In: Trost, A., Zajc, B. (eds.) International ERK Conference, pp. 113–116. IEEE Press, Los Alamitos (2005)
81. MacKay, D.: The evidence framework applied to classification networks. Neural Computation 4(5), 720–736 (1992)
82. MacKay, D.: In: Domany, E., van Hemmen, J.L., Schulten, K. (eds.) Bayesian Methods for Backpropagation Networks, Models of Neural Networks III, ch. 6, pp. 211–254. Springer, Newyork (1994)
83. Makrehchi, M., Kamel, M.: A text classification framework with a local feature ranking for learning socialnetworks. In: Seventh IEEE Conference on Data Mining, pp. 589–594. IEEE Computer Society, Los Alamitos (2007)
84. Manning, C., Schütze, H.: Foundations of Statistical Natural Language Processing. MIT Press, Cambridge (1999)
85. Masand, B.: Optimising Confidence of Text Classification By Evolution of Symbolic Expressions. In: Advances in Genetic Programming, ch. 21, pp. 459–476. MIT Press, Cambridge (1994)
86. McCallum, A., Nigam, K.: Employing EM and pool-based active learning for text classification. In: International Conference on Machine Learning - ICML 1998, pp. 350–358. Morgan Kaufmann Publishers, San Francisco (1998)
87. Mei, J., Zhang, W., Wang, S.: Grid enabled problem solving environments for text categorization. In: IEEE International Conference on e-Science and Grid Computing, pp. 106–110 (2006)
88. Melab, N., Cahon, S., Talbi, E.: Grid computing for parallel bioinspired algorithms. Journal of Parallel and Distributed Computing 66(9), 1052–1061 (2006)
89. Minier, Z., Bodo, Z., Csato, L.: Wikipedia-based kernels for text categorization. In: International Symposium on Symbolic and Numeric Algorithms for Scientific Computing, pp. 157–164. SYNASC (2007)
90. Mitchell, T.: Machine Learning. McGraw Hill, New York (1996)
91. Mladenić, D.: Feature subset selection in text learning. In: Nédellec, C., Rouveirol, C. (eds.) ECML 1998. LNCS, vol. 1398, pp. 95–100. Springer, Heidelberg (1998)

92. Moulinier, I., Ganascia, J.: Applying an Existing Machine Learning Algorithm to Text Categorization. In: Wermter, S., Riloff, E., Schaler, G. (eds.) Connectionist, Statistical, and Symbolic Approaches to Learning for Natural Language Processing, pp. 343–354. Springer, Heidelberg (1996)
93. Moulinier, I., Raskinis, G., Ganascia, J.: Text categorization: a symbolic approach. In: Annual Symposium on Document Analysis and Information Retrieval - SDAIR 1996, pp. 87–99 (1996)
94. Nedjah, N., Mourelle, L., Kacprzyk, J., França, F., Souza, A. (eds.): Intelligent Text Categorization and Clustering. Studies in Computational Intelligence, vol. 164. Springer, Heidelberg (2009)
95. Ng, A., Jordan, M.: On Discriminative vs. Generative Classifiers: A Comparison of Logistic Regressionand Naïve Bayes. In: Dietterich, T., Becker, S., Ghahramani, Z. (eds.) Advances in Neural Information Processing Systems - NIPS 2002, vol. 14, pp. 609–616. MIT Press, Cambridge (2002)
96. Ng, H., Goh, W., Low, K.: Feature selection, perceptron learning, and a usability case study for text categorization. In: International Conference on Research and Development in InformationRetrieval - ACM SIGIR 1997, pp. 67–73 (1997)
97. Nigam, K., Mccallum, A., Thrun, S., Mitchell, T.: Text classification from labeled and unlabeled documents using EM. Machine Learning 39(2/3), 103–134 (2000)
98. Pal, S.K., Narayan, B.L.: A web surfer model incorporating topic continuity. IEEE Transactions on Knowledge and Data Engineering 17(5), 726–729 (2005)
99. Park, C.: Dimension reduction using least squares regression in multi-labeled text categorization. In: IEEE International Conference on Computer and Information Technology, pp. 71–76 (2008)
100. Polikar, R.: Ensemble based systems in decision making. IEEE Circuits and Systems Magazine 6(3), 21–45 (2006)
101. Porter, M.: An agorithm for suffix stripping. Program 14(3), 130–137 (1980)
102. Potamias, G., Koumakis, L., Moustakis, V.: Enhancing web based services by coupling document classification with user profile. In: The International Conference on Computer as a Tool - EUROCON 2005, vol. 1, pp. 205–208 (2005)
103. Quinn, M.: Parallel Programming in C with MPI and OpenMP. McGraw-Hill, New York (2003)
104. Ribeiro, B., Silva, C., Vieira, A., Neves, J.: Extracting discriminative features using non-negative matrix factorization in financial distress data. In: ICANNGA 2009. LNCS, Springer, Heidelberg (2009)
105. Rigutini, L., Di Iorio, E., Ernandes, M., Maggini, M.: Semantic labeling of data by using the web. In: IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology Workshops, December 2006, pp. 638–641 (2006)
106. Rocchio, J.: Relevance Feedback in Information Retrieval. In: Salton, G. (ed.) The SMART Retrieval System: Experiments in Automatic Document Processing. Prentice Hall, Englewood Cliffs (1971)
107. Rose, T., Stevenson, M., Whitehead, M.: The reuters corpus volume 1 - from yesterday's news to tomorrow's language resources. In: International Conference on Language Resources and Evaluation - LREC 2002, pp. 29–31 (2002)
108. Ruiz, M., Srinivasan, P.: Automatic text categorization and its application to text retrieval. IEEE Transactions on Knowledge and Data Engineering 11(6), 865–879 (1999)
109. Ruiz, M., Srinivasan, P.: Hierarchical text categorization using neural networks. Information Retrieval 5(1), 87–118 (2002)
110. Saldarriaga, S., Morin, E., Viard-Gaudin, C.: Categorization of on-line handwritten documents. In: International Workshop on Document Analysis Systems, pp. 95–102 (2008)

111. Sarnovský, M., Butka, P.: Grid-enabled support for classification and clustering of textual documents. In: Slovakian-Hungarian Joint Symposium on Applied Machine Intelligence and Informatics, pp. 265–275 (2007)
112. Schaffer, C.: A conservation law for generalization performance. In: International Conference on Machine Learning - ICML 1994, pp. 259–265 (1994)
113. Schapire, R., Singer, Y.: Boostexter: A boosting-based system for text categorization. Machine Learning 39(2/3), 135–168 (2000)
114. Schapire, R., Singer, Y., Singhal, A.: Boosting and rocchio applied to text filtering. In: International Conference on Research and Development in Information Retrieval - ACM SIGIR 1998, pp. 215–223 (1998)
115. Schölkopf, B., Burges, C., Smola, A.: Advances in Kernel methods, pp. 1–15. MIT Press, Cambridge (1999)
116. Schohn, G., Cohn, D.: Less is more: Active learning with support vector machines. In: International Conference on Machine Learning - ICML 2000, pp. 839–846. Morgan Kaufmann, San Francisco (2000)
117. Schölkopf, B.: Support Vector Learning. R. Oldenbourg Verlag (1997)
118. Schölkopf, B., Smola, A.: Learning with Kernels. MIT Press, Cambridge (2002)
119. Schölkopf, B., Smola, A., Müller, K.: Nonlinear component analysis as a kernel eigenvalue problem. Neural Computation 10(5), 1299–1319 (1998)
120. Schütze, H., Hull, D., Pendersen, J.: A comparison of classifiers and document representations for the routing problem. In: International Conference on Research and Development in InformationRetrieval - ACM SIGIR 1995, pp. 229–237 (1995)
121. Sebastiani, F.: A tutorial on automated text categorisation. In: Argentinian Symposium on Artificial Intelligence - ASAI 1999, pp. 7–35 (1999)
122. Sebastiani, F.: Machine learning in automated text categorization. ACM Computing Surveys 34(1), 1–47 (2002)
123. Sebastiani, F.: Classification of text, automatic. In: Brown, K. (ed.) The Encyclopedia of Language and Linguistics, 2nd edn., vol. 14, pp. 457–462. Elsevier Science Publishers, Amsterdam (2006)
124. Shawe-Taylor, J., Cristianini, N.: Kernel Methods for Pattern Analysis. Cambridge University Press, Cambridge (2004)
125. Silva, C., Ribeiro, B.: Rare class text categorization with SVM ensemble. Journal of Electrotechnical Review (Przeglad Elektrotechniczny) 1, 28–31 (2006)
126. Silva, C., Ribeiro, B.: On text-based mining with active learning and background knowledge using SVM. Journal of Soft Computing - A Fusion of Foundations, Methodologies and Applications 11(6), 519–530 (2007)
127. Silva, C., Ribeiro, B.: The importance of stop word removal on recall values in text categorization. In: IEEE International Joint Conference on Neural Networks - IJCNN 2003, pp. 1661–1666 (2003)
128. Silva, C., Ribeiro, B.: Labeled and unlabeled data in text categorization. In: IEEE International Joint Conference on Neural Networks - IJCNN 2004, pp. 2971–2976 (2004)
129. Silva, C., Ribeiro, B.: Margin-based active learning and background knowledge in text mining. In: International Conference on Hybrid Intelligent Systems - HIS 2004, pp. 8–13 (2004)
130. Silva, C., Ribeiro, B.: Text classification from partially labeled distributed data. In: International Conference on Adaptive and Natural Computing Algorithms - ICANNGA 2005. LNCS, pp. 445–448. Springer, Heidelberg (2005)
131. Silva, C., Ribeiro, B.: Automated learning of RVM for large scale text sets: Divide to conquer. In: Corchado, E., Yin, H., Botti, V., Fyfe, C. (eds.) IDEAL 2006. LNCS, vol. 4224, pp. 878–886. Springer, Heidelberg (2006)

132. Silva, C., Ribeiro, B.: Ensemble RVM for text classification. In: International Conference on Neural Information Processing - ICONIP2006 (2006)
133. Silva, C., Ribeiro, B.: Fast-decision SVM ensemble text classifier using cluster computing. In: International Conference on Neural, Parallel & Scientific Computations - ICNPSC 2006, pp. 253–259 (2006)
134. Silva, C., Ribeiro, B.: Two-level hierarchical hybrid SVM-RVM classification model. In: IEEE International Conference on Machine Learning and Applications - ICMLA 2006, pp. 89–94 (2006)
135. Silva, C., Ribeiro, B.: Combining active learning and relevance vector machines for text classification. In: IEEE International Conference on Machine Learning and Applications - ICMLA 2007, pp. 130–135 (2007)
136. Silva, C., Ribeiro, B.: RVM ensemble for text classification. International Journal of Computational Intelligence Research 3(1), 31–35 (2007)
137. Silva, C., Ribeiro, B.: Towards expanding relevance vector machines to large scale datasets. International Journal of Neural Systems 18(1), 45–58 (2008)
138. Silva, C., Ribeiro, B.: Improving text classification performance with incremental background knowledge. In: ICANN 2009, Part I. LNCS, vol. 5768. Springer, Heidelberg (2009)
139. Silva, C., Ribeiro, B.: Improving visualization, scalability and performance of multiclass problems with SVM manifold learning. In: ICANNGA 2009. LNCS. Springer, Heidelberg (2009)
140. Silva, C., Ribeiro, B., Lotrič, U.: Speeding-up text classification in a grid computing environment. In: IEEE International Conference on Machine Learning and Applications - ICMLA 2005, pp. 113–116 (2005)
141. Silva, C., Ribeiro, B., Lotrič, U., Dobnikar, A.: Distributed ensemble learning in text classification. In: International Conference on Enterprise Information Systems - ICEIS 2008, pp. 420–423 (2008)
142. Silva, C., Ribeiro, B., Sung, A.: Boosting RVM classifiers for large data sets. In: Beliczynski, B., Dzielinski, A., Iwanowski, M., Ribeiro, B. (eds.) ICANNGA 2007. LNCS, vol. 4432, pp. 228–237. Springer, Heidelberg (2007)
143. Song, Y., Zhang, L., Giles, C.: Automatic tag recommendation algorithms for social recommender systems. ACM Transactions on the Web, TWEB (2009)
144. Szummer, M.: Learning from Partially Labeled Data. PhD thesis, Massachusetts Institute of Technology (2002)
145. Tenenbaum, J., Silva, V., Langford, J.: A global geometric framework for nonlinear dimensionality reduction. Science 290(5500), 2319–2323 (2000)
146. Tian, Y., Yang, Q., Huang, T., Lin, C., Gao, W.: Learning contextual dependency network models for link-based classification. IEEE Transactions on Knowledge and Data Engineering 18(11), 1482–1496 (2006)
147. Tipping, M.: Sparse bayesian learning and the relevance vector machine. Journal of Machine Learning Research 1, 211–214 (2001)
148. Tong, S., Koller, D.: Support vector machine active learning with applications to text classification. Journal of Machine Learning Research 2, 45–66 (2001)
149. Tzeras, K., Hartmann, S.: Automatic indexing based on bayesian inference networks. In: International Conference on Research and Development in Information Retrieval - ACM SIGIR 1993, pp. 22–34 (1993)
150. van Rijsbergen, C.: Information Retrieval. Butterworths Ed. (1979)
151. Vapnik, V.: The Nature of Statistical Learning Theory. Springer, Heidelberg (1995)
152. Vapnik, V.: Statistical Learning Theory. Wiley, Chichester (1998)

153. Vert, J., Matsui, T., Satoh, S., Uchiyama, Y.: High-level feature extraction using SVM with walk-based graph kernel. In: IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2009 (2009)
154. Vert, J., Yamanishi, Y.: Supervised graph inference. Advances in Neural Information Processing Systems 17, 1433–1440 (2005)
155. Wagner, J., Foster, J., Genabith, J.: A comparative evaluation of deep and shallow approaches to the automatic detection of common grammatical errors. In: Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning - EMNLP-CoNLL 2007, pp. 112–121 (2007)
156. Wang, T., Desai, B.: An approach for text categorization in digital library. In: 11th International Database Engineering and Applications Symposium, pp. 21–27 (2007)
157. Wei, L., Yang, Y., Nishikawa, R., Wernick, M., Edwards, A.: Relevance vector machine for automatic detection of clustered microcalcifications. IEEE Transactions on Medical Imaging 24(10), 1278–1285 (2005)
158. Weiss, S., Apté, C., Damerau, F., Johnson, D., Oles, F., Goetz, T., Hampp, T.: Maximizing text-mining performance. IEEE Intelligent Systems 14(4), 63–69 (1999)
159. Wiener, E., Pedersen, J., Weigend, A.: A neural network approach to topic spotting. In: Annual Symposium on Document Analysis and Information Retrieval - SDAIR 1995, pp. 317–332 (1995)
160. Wolpert, D., Macready, W.: No free lunch theorems for search. Technical Report SFI-TR-95-02-010, Santa Fe Institute (1995)
161. Wolpert, D., Macready, W.: Coevolutionary free lunches. IEEE Transactions on Evolutionary Computation 9(6), 721–735 (2005)
162. Wu, Z., Hsu, L., Tan, C.: A survey of statistical approaches to natural language processing. Technical Report TRA4/92, Department of Information Systems and Computer Science, National University of Singapore (1992)
163. Yang, Y., Liu, X.: A re-examination of text categorization methods. In: International Conference on Research and Development in Information Retrieval - ACM SIGIR 1999, pp. 42–49 (1999)
164. Yang, Y., Pedersen, J.: A comparative study on feature selection in text categorization. In: International Conference on Machine Learning - ICML 1997, pp. 412–420 (1997)
165. Yu, L., Wang, S., Lai, K., Wu, Y.: A framework of web-based text mining on the grid. In: IEEE International Conference on Next Generation Web Services Practices, pp. 97–102 (2005)
166. Zhang, T., Oles, F.: A probability analysis on the value of unlabeled data for classification problems. In: International Conference on Machine Learning - ICML 2000, pp. 1191–1198 (2000)
167. Zhang, X., Mei, J., Wang, S., Zhang, W.: Web services enabled text categorization system: Service infrastructure building. International Journal of Computer Science and Network Security 7(2), 73–77 (2007)
168. Zheng, Y., Duan, L., Tian, Q., Jin, J.: Tv commercial classification by using multi-modal textual information. In: IEEE International Conference on Multimedia and Expo, pp. 497–500 (2006)
169. Zhou, D., Huang, J., Schölkopf, B.: Learning from labeled and unlabeled data on a directed graph. In: International Conference on Machine Learning - ICML -2005, pp. 1036–1043 (2005)
170. Zhou, G., Zhang, J., Su, J., Shen, D., Tan, C.: Recognizing names in biomedical texts: a machine learning approach. Bioinformatics 20(7), 1178–1190 (2004)

Index