Università della Calabria

Dipartimento di Matematica

Dottorato di Ricerca in Matematica ed Informatica

XIX Ciclo

Settore Disciplinare INF/01 INFORMATICA

Tesi di Dottorato

A Technique for Automatic Generation of Rule-based Text Classifiers exploiting Negative Information

Chiara Cumbo

Supervisore	**Coordinatore**
Prof. Pasquale Rullo	Prof.ssa Annamaria Canino

A.A. 2005 - 2006

A Technique for Automatic Generation of Rule-based Text Classifiers exploiting Negative Information

Chiara Cumbo

Dipartimento di Matematica,
Università della Calabria
87036 *Rende, Italy*
email : cumbo@mat.unical.it

Sommario

Negli ultimi decenni, con la rapida diffusione di Internet, la produzione di documenti testuali in formato elettronico è aumentata in maniera esponenziale. Di conseguenza, si è verificata una necessità crescente di soluzioni automatizzate per l'organizzazione di grandi quantità di testi in formato digitale e una particolare attenzione è rivolta al loro uso futuro. La progettazione di tali soluzioni è stata tradizionalmente oggetto di studio dell'*Information Retrieval* (IR), la disciplina che si occupa dell'accesso automatico a dati la cui semantica è scarsamente specificata. Esistono due approcci principali per fornire l'accesso efficace a vaste sorgenti di dati non strutturati di tipo testuale: fornire strumenti per la ricerca di documenti rilevanti all'interno delle basi documentali (obiettivo dell'area di ricerca denominata *Text Search*); fornire strumenti per convertire tali sorgenti non strutturate in sorgenti strutturate, in modo da facilitarne la memorizzazione, l'accesso e la navigazione. Questo è l'obiettivo della *Text Classification* (TC), disciplina che coniuga diverse aree di ricerca dell'IR, del *Machine Learning* (ML), e del *Natural Language Processing* (NLP), e che mira alla costruzione di sistemi per il partizionamento di una collezione di documenti non strutturati in gruppi semantici attraverso l'assegnamento dei testi a una o più categorie predefinite. Possibili applicazioni della TC spaziano dall'indicizzazione automatica di articoli scientifici, all'organizzazione delle email, allo spam filtering.

Negli ultimi anni sono stati proposti diversi metodi di generazione di classificatori dei testi basati su tecniche di machine learning (ossia mediante l'uso di dati di training etichettati), fra cui classificatori *k nearest neighbor* (k-NN) e metodi basati su modelli probabilistici, tra i quali classificatori Bayesiani, classificatori lineari, alberi di decisione, reti neurali, Support Vector Machines. Panoramiche su questi sistemi sono riportate in [36; 52]. In [47; 17; 7] sono stati proposti interessanti approcci per la generazione di classificatori basati regole. Questi presentano la desiderabile proprietà di essere leggibili e di agevole interpretazione, al contrario di altri approcci (come ad esempio SVM, reti neurali etc). Alcuni metodi (basati su regole e non) che tentano di sfruttare sia informazione positiva che negativa ai fini della classificazione dei testi possono essere trovati in letteratura [58; 25; 5; 64].

Tuttavia i sistemi di TC esistenti presentano ancora importanti limitazioni di varia natura. In particolare:

- Molti approcci (classificatori Bayesiani, SVM, reti neurali) sono caratterizzati da un modello di tipo black box, ovvero non possono essere interpretati

da un lettore umano.

- Sistemi tradizionali per l'apprendimento di regole di classificazione, quali RIPPER , generano classificatori che spesso constano di parecchie centinaia di regole, quindi di difficile interpretazione; inoltre, essi sono complessi e generalmente poco efficienti.

- I classificatori associativi applicati a problemi di TC (basati essenzialmente su estensioni del popolare algoritmo di rule mining "a priori"), risultano altresì "leggibili", ma la qualità delle regole generate, a differenza di quelle costruite per dati strutturati, risulta piuttosto bassa.

- Taluni metodi di induzione di classificatori associativi, proposti recentemente, consentono di definire regole con termini sia positivi che negativi, e sono quindi caratterizzati da un maggiore livello di accuratezza del classificatore generato, ma anche da un processo di selezione delle regole più macchinoso e meno efficiente in termini di tempo.

In questo lavoro di tesi, proponiamo un metodo originale per la generazione automatica di classificatori testuali rule-based. Il sistema software che implementa tale metodo, denominato OLEX, genera un classificatore per le categorie di interesse predefinite, attraverso un processo di apprendimento (*training*) da un insieme di documenti pre-classificati (*training set*). Le regole generate da OLEX hanno una struttura molto particolare, del tipo "un letterale positivo e più letterali negativi". Più specificatamente, un classificatore è un insieme di regole logiche proposizionali del tipo:

$$c \leftarrow t_1 \in d, t_{n+1} \notin d, \cdots, t_{n+m} \notin d,$$

$$\ldots$$

$$c \leftarrow t_n \in d, t_{n+1} \notin d, \cdots, t_{n+m} \notin d$$

ciascuna caratterizzata da un termine positivo e da zero o più termini negativi, che sembrano individuare in maniera molto accurata i documenti che devono

essere associati ad una categoria predefinita e quelli che devono invece essere scartati. I termini positivi garantiscono una copertura (recall) elevata, mentre i termini negativi consentono di tenere la precisione sotto controllo.

Il metodo proposto per la generazione di tali regole si basa su un'euristica per determinare il "miglior" insieme di termini positivi (indicativi dell'appartenenza di un documento ad una predefinita categoria) e quelli negativi (indicativi di non-appartenenza) in base al quale costruire le regole di classificazione.

La nostra tecnica consente di risolvere molte delle limitazioni su esposte. Essa possiede infatti le seguenti desiderabili proprietà:

- è basato su idee molto semplici e dirette e perciò fornisce una chiara intuizione del modello alla base del processo di apprendimento;
- i classificatori generati (ovvero l'insieme di regole logiche) sono compatti (poche unità, al massimo qualche decina di regole), leggibili e di agevole interpretazione;
- fornisce elevata accuratezza;
- opera in maniera efficiente;
- è robusto, ovvero mostra un comportamento simile su tutti i dataset più largamente utilizzati in letteratura su cui è stato sperimentato.

Il sistema proposto presenta un comportamento simile o migliore rispetto ai sistemi presenti in letteratura e risulta particolarmente adatto in contesti applicativi a supporto dei processi di IR in cui è necessario realizzare buone prestazioni. Inoltre, in virtù della elevata dichiaratività del linguaggio utilizzato per la specifica delle regole e della leggibilità del modello di classificazione generato, ben si presta ad essere incluso in sistemi knowledge-based ed esteso mediante tecniche di preprocessing semantico per il riconoscimento di concetti nei documenti, in modo da sfruttare tale riconoscimento ai fini della classificazione dei documenti stessi rispetto ad una data tassonomia di categorie.

In questa tesi ci concentriamo sulle questioni succitate: proponiamo una nuova tecnica per la costruzione automatica di un classificatore di testi che da un lato risulti leggibile e di facile comprensione, proprietà desiderabili e spesso assenti nei sistemi di classificazione più avanzati (SVMs, Neural networks), ma che dall'altro risulti anche efficace ed efficiente. Qui di seguito riportiamo brevemente i principali contributi di questa tesi.

1. Studio della letteratura su metodi e sistemi di classificazione dei testi, che ha consentito di ottenere una valutazione comparativa delle loro caratteristiche nell'ambito della tematica di interesse;

2. Sviluppo di un nuovo metodo di machine learning per la generazione di regole logiche per la classificazione testuale;

3. Sviluppo di un sistema prototipale che implementa il metodo di cui sopra;

4. Valutazione sperimentale e comparativa delle prestazioni del sistema realizzato rispetto ai sistemi di varia natura di TC, sulle collezioni di documenti maggiormente utilizzate in letteratura;

5. Integrazione e sperimentazione del sistema di OLEX all'interno di una piattaforma di content management sviluppata nell'ambito del progetto PIA-Exeura-03-06.

Contents

Introduction **11**
Text Categorization 11
Main Contribution 12
Organization of the Thesis 14

I Text Classification **15**

1 Text Classification Problem **17**
1.1 Problem Definition 17
1.2 Applications of text categorization 19
1.3 Different Approaches to Document Categorization 22

2 Performance Measures **25**
2.1 Measures of Categorization Effectiveness 25
2.1.1 Precision and Recall 26
2.1.2 Combined measures 28

3 Standard Test Collections **29**
3.1 Benchmark Dataset 29
3.2 Reuters Corpus 30
3.3 OHSUMED Corpus 32

II Related Work **34**

4 Rule-Based Classifiers **36**
4.1 Association Rule Mining 37
4.2 Decision Trees 40

4.3 Inductive Rule Learning Methods 41
4.3.1 RIPPER 42
4.3.2 TRIPPER 43

5 Use of Negative Information in TC 46
5.1 A variant of *k*-NN using Negative Evidence 46
5.1.1 Instance Based Classifier Induction 46
5.1.2 Negative Evidence in Instance-Based Classifiers 47
5.2 A use of Negative Information in Feature Selection 48

6 Associative Classifiers with Negation for Structured Data and Texts 50
6.1 Associative Classifiers with Negation 51
6.1.1 Negative Association Rules 51
6.1.2 Generation of Positive and Negative Rules 52
6.2 Associative Text Classifiers with Negation 55
6.2.1 Specializing Associative Rules by means of Negated Words 55

III OLEX: a System for Automatic Generation of Rule-based Text Classifiers exploiting Negative Information 59

7 Problem Definition and Terminology 61
7.1 Problem Statement 61

8 Learning Rules with Negation 63
8.1 Pre-processing 63
8.2 Learning 64
8.2.1 Vocabulary Reduction by Term Selection 64
8.2.2 Generation of Discriminating Terms 65
8.3 Classifier Construction 68

9 Experimental Results 70
9.1 Experiments Setup and Performance Metrics 70
9.2 Reuters 71
9.3 OHSUMED 75
9.4 FCSI 77
9.5 Final Remarks 79

10 Comparison and Discussion **80**
10.1 Rule Expressiveness 80
10.2 How Rules are Learned 81
10.3 Rule Effectiveness 82

11 Conclusions **83**

12 Future Work **85**

Abstract

Information Retrieval is concerned with locating information that will satisfy a user's information need. Traditionally, the emphasis has been on text retrieval: providing access to natural language texts where the set of documents to be searched is large and topically diverse. In a text categorization task, the system is responsible for assigning a document to one or more categories from among a given set of categories.

In this way, users are allowed to browse more easily the set of texts of their own interests, by navigating in category hierarchies. This paradigm is very effective for retrieval and for filtering of information but also in the development of user-driven on-line services. Given the large amounts of documents involved in the above applications, automated approaches to categorize data efficiently are needed.

The automated categorization (or classification) of texts into prespecified categories, although dating back to the early '60s, has witnessed a booming interest in the last ten years, due to the increased availability of documents in digital form and the ensuing need to organize them. In the research community the dominant approach to this problem is based on the application of supervised machine learning techniques: a general inductive process automatically builds a classifier by learning, from a set of previously classified documents, the characteristics of one or more categories. The advantages of this approach over the knowledge engineering approach (consisting in the manual definition of a classifier by domain experts) are a very good effectiveness, considerable savings in terms of expert manpower, and straightforward portability to different domains.

A good text classifier is a classifier that efficiently categorizes large sets of text documents in a reasonable time frame and with an acceptable accuracy, and that provides classification rules that are human readable for possible fine-tuning. If the training of the classifier is also quick, this could become in some application domains a good asset for the classifier. Many techniques and algorithms for automatic text categorization have been devised. According to published literature, some are more accurate than others, and some provide more interpretable classification models than others. However, none can combine all the beneficial properties enumerated above.

In this dissertation, we have defined and implemented a novel approach OLEX, for automatic text categorization. OLEX relies on an optimization algorithm whereby a set of both positive and negative information are generated from a set of training documents in order to learn profiles of predefined categories with respect to which we wish to construct text classifiers. The proposed method is simple and elegant. Despite this our text categorization method proves to be efficient and effective, and experiments on well-known collections show that the classifier performs well. In addition, training as well as classification are both fast and the generated rules are human readable.

Olex has been fully integrated into an industrial text classification system developed at Exeura s.r.l.

Briefly, the main contributions of the thesis are the following:

1. We study methods and systems for automatic text classification, analyze their complexity and their exploitation for a critical comparison.

2. We design a new machine learning method for generating logic rules for text categorization.

3. We implement our approach in a prototype, the OLEX system.

4. We perform a systematic experimentation and report experimental results on a number of well-known benchmark text collections to assess the impact of our approach and to compare it with respect to other systems.

5. We integrate the support for learning process in OLEX Content Management Suite, within project PIA-Exeura-03-06.

Introduction

Text Categorization

Automatic Text Categorization (ATC) has a long history, dating back at least to 1960, and has always been an important application and research topic since the first usage of electronic documents. Until the late 80s, the dominant approach to the problem involved knowledge-engineering automatic categorizers, i.e. manually building a set of rules encoding expert knowledge on how to classify documents. In the 90s, with the booming production and availability of on-line documents, automated text categorization has witnessed an increased and renewed interest. A newer paradigm based on *machine learning* has superseded the previous approach.

The text classification task can be defined as assigning category labels to new documents based on the knowledge gained in a classification system at the training stage. In the training phase we are given a set of documents with class labels attached, and a classification system is built using a learning method.

Nowadays, text categorization is a necessity due to the very large amount of text documents that we have to deal with daily. A text categorization system can be used in indexing documents to assist information retrieval tasks as well as in classifying e-mails, memos or web pages in a yahoo-like manner. Needless to say, automatic text categorization is essential.

A *text classifier* (or simply "classifier") is a program capable of assigning natural language texts to one or more thematic categories on the basis of their contents. A number of machine learning methods to automatically construct classifiers using labelled training data have been proposed in the last few years, including *k*-nearest neighbors (*k*-NN), probabilistic Bayesian, neural networks and SVMs. Overviews of these techniques can be found in [36; 52]. In a different vein, rule learning algorithms, such as [47; 17; 7], have become a successful strategy for text categorization. Rule-based classifiers provide the desirable prop-

erty of being readable, easy for people to understand, contrary to most of the other approaches that lack interpretability (e.g., SVMs, Neural networks, etc.). Several approaches (either rule-based or not) exploiting negative information for text classification can be found in the literature, such as those in [58; 25; 6; 64].

Main Contribution

This thesis is concerned with the study of Text Categorization and the definition of a novel approach exploiting both positive and negative information generated from a set of training documents in order to learn profiles of predefined categories with respect to which we wish to construct text classifiers.

Some evidence raised during our studies:

- Although many approaches have been proposed, automated text categorization is still a major area of research primarily because the effectiveness of current automated text classifiers is not faultless and still needs improvement.
- Some methods are more accurate than others.
- Some methods provide more interpretable classification models than others.
- There is not a system which can combine all the beneficial properties mentioned above.

Our work faces the above issues, aiming at overcoming the limitations of existent text categorization systems by defining a novel method able to:

(i) perform as well as other methods in the literature.

(ii) be fast during both training and categorization phases.

(iii) build classifiers which can be read, understood and modified by humans.

In this dissertation we describe OLEX, a novel method for the automatic construction of rule-based classifiers. Here, a classifier is a set of propositional rules of the form

$$c \leftarrow t_1 \in d, t_{n+1} \notin d, \cdots, t_{n+m} \notin d,$$

$\cdots$

$c \leftarrow t_n \in d, t_{n+1} \notin d, \cdots, t_{n+m} \notin d$

stating the condition "if any of the terms (n-grams) $t_1, \cdots, t_n$ occurs in d and *none* of the terms $t_{n+1}, \cdots, t_{n+m}$ occurs in d then classify d under category c". Thus, the appearance of a (positive) term t_i, $1 \leq i \leq n$, in a document d requires the contextual *absence* of a (possibly empty) set of other (negative) terms $t_{n+1}, \cdots, t_{n+m}$ in order for d be classified under c[1].

The process of constructing a classifier for a category c from a training set TS relies on the following steps. First, the reduced vocabulary V is constructed by selecting a set of "meaningful" terms from TS. Second, starting from V, an optimization algorithm is applied to determine a "best" set X_c of *discriminating* terms for c; a discriminating term is either positive (i.e., indicative of membership in c) or negative (i.e., indicative of non-membership). Third, based on X_c, the classifier of c is constructed.

Experimental results, obtained on two standard benchmark data collections, namely, REUTERS-21578 and OHSUMED, confirm the expectations on our model. In summary they show that:

1. OLEX achieves good performance on both data collections, performing similar or outperforming most of the existing classifiers; intuitively, the paradigm "one positive term, zero or more negative terms" seems to be effective as positive terms allow us to catch most of the right documents, while negative ones help us not to make "too many" mistakes;

2. OLEX produces compact and readable classifiers – they usually range from just some to a few dozens rules;

3. OLEX operates efficiently.

Briefly, the main contributions of the thesis are the following:

1. We study methods and systems for automatic text classification, analyze their complexity and their exploitation for a critical comparison.

2. We design a new machine learning method for generating logic rules for text categorization.

3. We implement our approach in a prototype, the OLEX system.

[1] in general, d may "satisfy" more classifiers, so that it may be assigned to multiple categories

4. We perform a systematic experimentation and report experimental results on a number of well-known benchmark text collections to assess the impact of our approach and to compare it with respect to other systems.

5. We integrate the support for automatic generation of text classifiers in OLEX Content Management Suite, within project PIA-Exeura-03-06.

Organization of the Thesis

This thesis consists of three parts. First, (i) we introduce the Text Categorization problem, then (ii) we discuss some interesting related works and finally (iii) we introduce our machine learning approach aimed at generating rule-based text classifiers. More in details, the organization of the thesis is described as follows.

[I] The first part introduces the text categorization task, focuses on the ML approach to ATC and addresses some issues about performance measures and standard test collections used to evaluate the effectiveness of a text classifier.

[II] The second part describes some related works concerning rule-based approaches that can be found in the literature as well as methods exploiting negative information.

[III] Finally, the third part describes OLEX, a system for the automatic construction of rule-based text classifiers relying on a novel method exploiting both positive and negative information.

Part I

Text Classification

In this part we present Automated Text Categorization (ATC).

Automatic text classification (ATC) is a discipline at the crossroads of information retrieval (IR), machine learning (ML), and computational linguistics (CL), and consists in the realization of text classifiers, i.e. software systems capable of assigning texts to one or more categories, or classes, from a predefined set. Applications range from the automated indexing of scientific articles, to e-mail routing, spam filtering, authorship attribution, and automated survey coding.

This part of the thesis will focus on the ML approach to ATC, whereby a software system (called the learner) automatically builds a classifier for the categories of interest by generalizing from a training set of pre-classified texts.

The part is organized as follows:

- Chapter 1 provides a formal definition of the text classification problem.
- In Chapter 2 we give a detailed analysis of the performance measures defined in Information Retrieval and their application to TC.
- Finally, in Chapter 3, we illustrate the benchmark corpora widely used to evaluate text classifiers.

Chapter 1

Text Classification Problem

Automatic Text Categorization (ATC) is the problem of automatically assigning one or more predefined categories to free text documents. While more and more textual information is available online, effective retrieval is difficult without good indexing and summarization of document content. Document categorization is one solution to this problem.

In this chapter, we provide a formal definition of the text classification task and we briefly review the most important applications in which ATC has been used [51].

1.1 Problem Definition

Document (or *text*) *categorization* (or *classification*) may be seen as the task of determining an *assignment* of a value from $0, 1$ to each entry of the *decision matrix* AS represented in Table 1.1: where $C = c_1, ..., c_m$ is a set of pre-defined *categories*, and $D = d_1, ..., d_n$ is a set of documents to be categorized. A value of 1 for a_{ij} is interpreted as a decision to file d_j under c_i, while a value of 0 is

	d_1	...	...	d_k	...	...	d_n
c_1	a_{11}	...	...	a_{1k}	...	...	d_{1n}
...	...	...	...	...	...	...	...
c_i	a_{i1}	...	...	a_{ik}	...	...	d_{in}
...	...	...	...	...	...	...	...
c_m	a_{m1}	...	...	a_{mk}	...	...	d_{mn}

Table 1.1: Decision Matrix A

interpreted as a decision not to file d_j under c_i.

More formally, the task is to approximate the unknown total function

$$f : D \times C \rightarrow \{0, 1\}$$

that describes how documents ought to be classified, by means of a total function

$$f' : D \times C \rightarrow \{0, 1\}$$

(called the *classifier*, or *model*) such that f and f' coincide as much as possible (how to define precisely and measure this degree of coincidence, called *effectiveness*, will be discussed in Chapter 2).
Fundamental to the understanding of this task are two observations:

- the categories are just symbolic labels. No additional knowledge of their "meaning" is available to help in the process of building the categorizer; in particular, this means that the "text" constituting the label (e.g. `Art` in a news categorization task) cannot be used;
- the attribution of documents to categories should, in general, be attributed on the basis of the *semantics* of the documents, and not on the basis of *metadata* (e.g. publication date, document type, etc.) that may be available from an external source. This means that the notion of relevance of a document to a category is inherently subjective.

Single-label and Multi-label Categorization

Most of the research in text categorization has been devoted to *binary* problems where a document is classified as either *relevant* or not *relevant* with respect to a predefined topic. However there are many sources of textual data such as Internet News, electronic mail and digital libraries,which are composed of different topics and which therefore pose a *multi-class* categorization problem. We might want that for a given integer k, exactly k (or $\leq k$, or $\geq k$) elements of C must be assigned to each element of D. The case $k = 1$ Is often called the *single-label* case (or the *non-overlapping categories* case). Moreover, in multi-class problems it is often the case that documents are relevant to more than one category. For example a news article may well be relevant to several topics. The general case in which any number of categories from 0 to m may be assigned to the same document is called the *multi-label* case. The common approach for multi-class, multi-label

text categorization is to break the task into disjoint binary categorization problems one for each class. To classify a new document one needs to apply all the binary classifiers and combine their predictions into a single decision. The end result is a ranking of possible topics. The main drawback with this approach is that it ignores any correlation between the different classes.

Category-pivoted categorization or document-pivoted categorization

An important distinction is whether we want to fill the matrix one row at a time (*category-pivoted categorization*-CPC), or fill it one column at a time (*document-pivoted categorization*. DPC). This distinction is mostly pragmatic rather than conceptual, but is important in the sense that the sets C of categories and D of documents are not always available in their entirety right from the start. DPC is thus suitable when documents might become available one at a time over a long span of time, e.g. in the case a user submits one document at a time for categorization, rather than submitting a whole batch of them all at once. In this case, sometimes the categorization task takes the form of ranking the categories in decreasing order of their estimated appropriateness for document d; because of this, CPC is sometimes called *category-ranking* classification or *on-line* classification [63]. CPC is instead suitable if we consider the possibility that a new category c_{m+1} is inserted into a previously existing set of categories $C = c_1, ..., c_m$ after a number of documents have already been categorized under C, which means that these documents need to be categorized under c_{m+1} too. In this case, sometimes the categorization task takes the form of ranking the documents in decreasing order of their estimated appropriateness for category c_{m+1}; symmetrically to the previous case, DPC may also be called *document-ranking* classification. CPC is more commonly used than DPC, as the case in which documents are submitted one at a time is somehow more common than the case in which newer categories dynamically crop up.

1.2 Applications of text categorization

The automatic categorization of texts dates at least from the early '60s ([41]) and it has been used in a number of various actual domains. In the following, we briefly provide the most important applications of document categorization which have become the focus of many recent research efforts.

Automatic indexing for Boolean information retrieval systems

The first use to which automatic categorizers were put at, and the application that spawned most of the early research in the field, is that of automatic document indexing for use in information retrieval (IR) systems relying on a controlled dictionary. The most prominent example of such IR systems is, of course, that of Boolean systems. In these systems, each document is assigned one or more keywords or keyphrases describing its content, where these keywords and keyphrases belong to a finite set of words, called controlled dictionary) and often consisting of a hierarchical thesaurus (e.g. the MESH thesaurus covering the medical field [2]). Usually, this assignment is performed by trained human indexers, and is thus an extremely costly activity. If the entries in the thesaurus are viewed as categories, document indexing becomes an instance of the document categorization task, and may thus be addressed by the automatic techniques. In this case a typical constraint may be that $k_1 \leq x \leq k_2$ keywords are assigned to each document, for given k_1, k_2. Document-pivoted categorization might typically be the best option, so that documents are categorized on a first come, first served basis. Various automatic document categorizers explicitly addressed at the document indexing application have been described in the literature [20; 23; 11].

Document organization

In general, all issues pertaining to document organization and filing , be it for purposes of personal organization or document repository structuring, may be addressed by automatic categorization techniques. For instance, at the offices of a newspaper, incoming classified ads should be, prior to publication, categorized under the categories used in the categorization scheme adopted by the newspaper. While most newspapers would handle this application manually, those dealing with a high daily number of classified ads might prefer an automatic categorization system to choose the most suitable category for a given ad. Note that in this case a typical constraint might be that exactly one category is assigned to each document. Again, a first-come, first-served policy might look the aptest here, which would make one lean for a document-pivoted categorization style.

Document filtering

Document filtering (also known as document routing) refers to the activity of categorizing a dynamic, rather than static, collection of documents, in the form of a stream of incoming documents dispatched in an asynchronous way by an information producer to an information consumer [10]. A typical case of this is a newsfeed, whereby the information producer is a news agency (e.g. Reuters or Associated Press) and the information consumer is a newspaper. In this case, the filtering system should discard (i.e. block the delivery to the consumer of) the documents the consumer is not likely to be interested in (e.g. all news not concerning sports, in the case of a sports newspaper). Filtering can be seen as a special case of categorization with non-overlapping categories, i.e. the categorization of incoming documents in two categories, the relevant and the irrelevant. Additionally, a filtering system may also perform a further categorization into topical categories of the documents deemed relevant to the consumer; in the example above, all articles about sports are deemed relevant, and should be further subcategorized according e.g. to which sport they deal with, so as to allow individual journalists specialized in individual sports to access only documents of high prospective interest for them. The construction of information filtering systems by means of machine learning techniques is widely discussed in the literature [50].

Web pages categorization

Automatic document categorization has recently arisen a lot of interest also for its possible Internet applications. One of these is automatically categorizing Web pages, or sites, into one or several of the categories that make up commercial hierarchical catalogues (such as those embodied in `Yahoo!` and other Internet portals). When Web documents are catalogued in this way, rather than addressing a generic query to a general-purpose Web search engine, a searcher may find it easier to first navigate in the hierarchy of categories and then issue his search from (i.e. restrict his search to) a particular category of interest. Automatically categorizing Web pages has obvious advantages, since the manual categorization of a large enough subset of the Web is problematic to say the least.

1.3 Different Approaches to Document Categorization

Knowledge-Engineering Approach

In the '80s, the main approach used to the construction of automatic document categorizers involved *knowledge-engineering* them, i.e. manually building an expert system capable of taking categorization decisions. Such an expert system might have typically consisted of a set of manually defined rules (one per category) of type:

$$if < DNFBooleanformula > then < category >$$

to the effect that if the document satisfied $< DNFBooleanformula >$ (DNF standing for *disjunctive normal form*), then it was categorized under $< category >$. The typical example of this approach is the Construe system [28], built by Carnegie Group for use at the Reuters news agency. The main drawback of this manual approach to the construction of automatic classifiers is the existence of a *knowledge acquisition bottleneck*, similarly to what happens in expert systems. That is, rules must be manually defined by a knowledge engineer with the aid of a domain expert (in this case, an expert in document relevance to the chosen set of categories). If the set of categories is updated, then these two professional figures must intervene again, and if the classifier is ported to a completely different domain (i.e. set of categories), the work has to be repeated anew. On the other hand, it was suggested that this approach can give very good effectiveness results: Hayes et al. [28] report a 0.90 breakeven result (that we will define in Chapter 2) on a subset of the REUTERS-21578 (a well known test collection that we review in Chapter 3) test collection, a figure that outperforms most of the best classifiers built in the late '90s by machine learning techniques. However, no other classifier has been tested on the same dataset as Construe, and it is not clear how this dataset was selected from the REUTERS-21578 collection (i.e. whether it was a random or a favourable subset of the whole collection). As convincingly argued in [62] and in [51], the results above do not allow us to confidently say that these effectiveness results may be obtained in the general case.

Machine Learning Approach

Since the early '90s, a new approach to the construction of automatic document classifiers, the *machine learning approach* [42], has gained prominence and even-

tually become the dominant one. In the machine learning approach a general *inductive* process automatically builds a classifier for a category c_i by observing the characteristics of a set of documents that have previously been classified manually under c_i by a domain expert; from these characteristics, the inductive process gleans the characteristics that a novel document should have in order to be categorized under c_i . Note that this allows us to view the construction of a classifier for the set of categories $C = c_1, \ldots, c_m$ as m independent tasks of building a classifier for a single category $c_i \in C$, each of these classifiers being a rule that allows to decide whether document d_j should be categorized under category c_i. The advantages of this approach over the previous one are evident: the engineering effort goes towards the construction not of a classifier, but of an automatic builder of classifiers. This means that if the original set of categories is updated, or if the system is ported to a completely different domain, all that is needed is the inductive, automatic construction of a new classifier from a different set of manually categorized documents, with no required intervention of either the domain expert or the knowledge engineer. In terms of effectiveness, categorizers build by means of machine learning techniques nowadays achieve impressive levels of performance, making automatic classification a qualitatively viable alternative to manual classification.

In this dissertation, owing to its wide applicability, we propose a new system OLEX adopting the machine learning approach.

Training Set and Test Set

As previously mentioned, the machine learning approach relies on the existence of a an initial *corpus* $Co = d_1, ..., d_s$ of documents previously categorized under the same set of categories $C = c_1, \ldots, c_m$ with which the categorizer must operate.

This means that the corpus comes with a *correct decision matrix* CA_{ij}.

A value of 1 for ca_{ij} is interpreted as an indication from the expert to file d_j under c_i, while a value of 0 for is interpreted as an indication from the expert not to file d_j under c_i. A document d_j is often referred to as a positive example of c_i if ca_{ij} = 1, a negative example of c_i if ca_{ij} = 0.

For evaluation purposes, in the first stage of classifier construction the initial corpus is typically divided into two sets, not necessarily of equal size:

- a *Training Set* $Tr = \overline{d_1}, \ldots, \overline{d_g}$. This is the set of example documents observing the characteristics of which the classifiers for the various categories are induced;

- a ***Test Set*** $Te = \overline{d_{g+1}}, ..., \overline{d_s}$. This set will be used for the purpose of testing the effectiveness of the induced classifiers. Each document in Te will be fed to the classifiers, and the classifier decisions compared with the expert decisions; a measure of classification effectiveness will be based on how often the values for the a_{ij}'s obtained by the classifiers match the values for the ca_{ij}'s provided by the experts.

Note that in order to give a scientific character to the experiment the documents in Te cannot participate in any way in the inductive construction of the classifiers; if this condition were not to be satisfied, the experimental results obtained would typically be unrealistically good.

Chapter 2

Performance Measures

As in the case of information retrieval systems, the evaluation of document classifiers is typically conducted experimentally, rather than analytically. The reason for this tendency is that, in order to evaluate a system analytically (e.g. proving that the system is correct and complete) we always need a formal specification of the problem that the system is trying to solve (e.g. with respect to what correctness and completeness are defined), and the central notion of document classification (namely, that of relevance of a document to a category) is unfit for formalization, due to its subjective character.

In this chapter, we report the classic IR notions of *precision* (Pr) and *recall* (Re), adapted to the case of document categorization classifiers, and some combined measures, F-measure and break-even point, used throughout the remainder of this dissertation in order to express evaluation of different classifiers previously devised in literature compared with our system OLEX.

2.1 Measures of Categorization Effectiveness

The experimental evaluation of classifiers, rather than concentrating on issues of efficiency, usually tries to evaluate the *effectiveness* of a classifier, i.e. its capability of taking the right categorization decisions. The main reasons for this bias are that efficiency is a notion dependent on the hw/sw technology used. Once this technology evolves, the results of experiments aimed at establishing efficiency are no longer valid. This does not happen for effectiveness, as any experiment aimed at measuring effectiveness can be replicated, with identical results, on any different or future hw/sw platform; effectiveness is really a measure of how the system is good at tackling the central notion of classification, that of relevance of

a document to a category.

In the following subsections we provide a definition of the effectiveness measures of precision, recall and F-measure and break-even point.

2.1.1 Precision and Recall

Classification effectiveness is measured in terms of the classic IR notions of precision (Pr) and recall(Re), adapted to the case of document categorization.

- *Precision* wrt c_i (Pr_i) is defined as the conditional probability

  $$P(ca_{ix} = 1 | a_{ix} = 1)$$

  i.e. as the probability that if a random document d_x is categorized under c_i, this decision is correct.

- Analogously, *recall* c_i $(Re_i)$is defined as the conditional probability

  $$P(a_{ix} = 1 | ca_{ix} = 1)$$

  i.e. as the probability that, if a random document d_x should be categorised under c_i, this decision is taken.

These category-relative values may be averaged, in a way to be discussed shortly, to obtain Pr and Re, i.e. values global to the whole category set. Borrowing terminology from logic, Pr may be viewed as the "degree of soundness" of the classifier wrt the given category set C, while Re may be viewed as its "degree of completeness" wrt C. As they are defined here, (Pr_i) and (Re_i) (and consequently Pr and Re) are to be understood, as subjective probabilities, i.e. values measuring the expectation of the user that the system will behave correctly when classifying a random document under c_i. If we define

- FP_i *false positives* wrt c_i the number of documents of the test set that have been incorrectly classified under c_i;

- TN_i *true negatives* wrt c_i the number of documents of the test set that have been correctly not classified under c_i;

- TP_i *true positives* wrt c_i the number of documents of the test set that have been correctly classified under c_i;

- FN_i *false negatives* wrt c_i the number of documents of the test set that have been incorrectly not classified under c_i.

precision wrt c_i and recall wrt c_i may thus be estimated as:

$$(Pr_i) = \frac{TP_i}{TP_i + FP_i} \tag{2.1}$$

$$(Re_i) = \frac{TP_i}{TP_i + FN_i} \tag{2.2}$$

For obtaining estimates of precision and recall relative to the whole category set, two different methods may be adopted:

- *microaveraging*: precision and recall are obtained by globally summing over all individual decisions, i.e.:

$$(Pr_i)_\mu = \frac{TP}{TP + FP}(Re_i)_\mu = \frac{TP}{TP + FN} \tag{2.3}$$

- precision and recall are first evaluated locally for each category, and then globally by averaging over the results of the different categories, i.e.

$$(Pr_i)^M = \frac{\sum_{i=1}^{m} Pr_i}{m} \tag{2.4}$$

$$(Re_i)^M = \frac{\sum_{i=1}^{m} Re_i}{m} \tag{2.5}$$

[51] underlies how these two methods may give quite different results, especially if the different categories are unevenly populated: for instance, if the classifier performs well on categories with a small number of positive test instances, its effectiveness will probably be better according to macroaveraging than according to microaveraging. There is no agreement among authors on which is better. Some believe that microaveraged performance is somewhat misleading because more frequent topics are weighted heavier in the average[57] and thus favour macroaveraging, while others believe that topics should indeed count proportionally to their frequency, and thus lean towards microaveraging.

2.1.2 Combined measures

Neither precision nor recall make sense in isolation of the other. In fact, in order to obtain a classifier with 100% recall, one would only need to set every threshold to 0, thereby obtaining the trivial acceptor (i.e. the classifier that classifies all documents under all categories). Quite obviously, in this case precision would usually be very low (more precisely, equal to *apc*, the average percentage of categories per test document). Conversely, it is well-known from everyday information retrieval practice that higher levels of precision may be obtained at the price of a low recall. In practice, by tuning thresholds, a classification algorithm is tuned so as to improve *Pr* to the detriment of *Re* or viceversa. A classifier should thus be measured by means of a combined effectiveness measure which both Pr and Re concur to determine. Various such measures have been proposed, among which the following are most frequent:

- effectiveness is computed as the *breakeven point*, i.e. the value at which *Pr* equals *Re*
- effectiveness is computed as the value of the F_α function ([34]), for some $0 \leq \alpha \leq 1$:

$$F_\alpha = \frac{1}{\alpha \frac{1}{Pr} + (1-\alpha)\frac{1}{Re}} \tag{2.6}$$

In this formula α may be seen as the relative degree of importance attributed to *Pr* and *Re*: if $\alpha = 1$, then F_α coincides with *Pr*, if $\alpha = 0$ then F_α coincides with *Re*. Usually, a value of α= 0.5 is used, which attributes equal importance to *Pr* and *Re*. Rather than $F_{0.5}$ this is usually called F_1 (see [34] for details). As shown in [62], for a given classifier, its breakeven value is always less or equal than its F_1 value.

Once an effectiveness measure is chosen, a classifier can be tuned (e.g. thresholds and other internal parameters can be set) so that the resulting effectiveness is the best achievable by that classifier. The tuning of a parameter *p* (be it a threshold or other) is normally done experimentally.

Chapter 3

Standard Test Collections

The existence, public availability, and widespread acceptance of a standard corpora (that is, set of documents) for a given information retrieval task are beneficial to research on this task, because they allow different researchers to experimentally compare their own systems by comparing the results they have obtained on this benchmark.

In this chapter, we introduce two well-know collections, REUTERS-21578 corpus and OHSUMED corpus, commonly used in text categorization experimentation. In the following chapters of this dissertation, we will present, where possible, a systematic, comparative study of experiments on these corpora conducted in previous works and by our system OLEX.

3.1 Benchmark Dataset

To establish the retrieval performance of an IR system, it is necessary to use benchmark dataset. Among the test collections available in the public domain, especially two play a dominant role and are widespread accepted:

- the REUTERS-21578 corpus
- the OHSUMED corpus

These two text collections (especially REUTERS-21578 together with its older version REUTERS-22173) are the most used for the experimentation performed in text categorization.

Unfortunately, to use the same test collection it is necessary but not sufficient for obtaining definitive results concerning whether a given classifier outperforms

another, because many of these experiments have been carried out in sometimes different experimental conditions. In fact, at least six different versions (including REUTERS-21578) of the REUTERS-22173 collection have been carved out of the original and used for experimentation.

In order to directly compare the experimental results on two different classifiers, the experiments should be performed under the following conditions [51]:

- the same collection (i.e. same documents and same categories) is used for both classifiers;
- the same choice (split) of training set and test set is made for both classifiers;
- the same effectiveness measure is used for both classifiers.

Unfortunately, much of earlier experimentation (at least until 1997) was not performed with this rules in mind, therefore results reported by these researches are seldom directly comparable. By experimenting three different classifiers on five versions of ReutersOld [60] has experimentally shown that a lack of compliance with these three conditions (and with the first condition in particular) may significantly influence the experimental results. In particular, [60] shows that experiments carried out on Version 2 are not directly comparable with those using later versions, since the former includes a significant percentage (58%) of unlabelled test documents which, being negative examples of all categories, tend to depress effectiveness.

3.2 Reuters Corpus

The REUTERS-21578 test collection [1], together with its earlier variants, has been a standard benchmark for the text categorization task throughout the last 10 years. It consists of a revised version of an older corpus known as ReutersOld [33].

The documents are newswire stories covering the period between 1987 and 1991. The documents were originally labelled by Carnegie Group, Inc. and Reuters, Ltd. in the course of developing the CONSTRUE text categorization system [27], and were subsequently collected and formatted by David Lewis with the help of several other people. The previous version of the collection REUTERS-22173 , was used in a number of published studies up until 1996, when a revision

[1]Distribution 1.0 test collection is available for research purposes only from http://www.daviddlewis.com/resources/testcollections/reuters21578

of the collection resulted in the removal of 595 duplicates from the original set of 22,173 documents, thus leaving the 21,578 documents that now make REUTERS-21578 , and in the correction of several other errors. The REUTERS-21578 documents actually used in TC experiments are only 12,902, because the creators of the collection found ample evidence that the other 8,676 documents had not been considered for labeling by the people who manually assigned categories to documents (indexers). To make different experimental results comparable, standard splits (i.e., partitions into a training and a test set) have been defined by the creators of the collection on the 12,902 documents. Apart from very few exceptions, TC researchers have used the *ModApté split*, in which 9,603 documents are selected for training and the other 3,299 form the test set.

As we will see in Chapter 9 and in Chapter 10, the experiments performed on this dataset on our system OLEX refer to the ModApté split. An interesting and complete analysis of this split is reported in [18].

Group of Categories

There are five groups of categories that label REUTERS-21578 documents:

- EXCHANGES
- ORGS
- PEOPLE
- PLACES
- TOPICS.

Only the TOPICS group has actually been used in TC experimental research, because the other four groups do not constitute a very challenging benchmark for TC. The TOPICS group contains 135 categories. Some of the 12,902 legitimate documents have no categories attached to them, but unlike the 8,676 documents removed from consideration they are unlabelled because the indexers deemed that none of the TOPICS categories applied to them. Among the 135 categories, 20 have (in the ModApté split) no positive training documents; as a consequence, these categories have never been considered in any TC experiment, because the TC methodology requires deriving a classifier either by automatically training an inductive method on the training set only, and/or by human knowledge engineering based on the analysis of the training set only.

Reuters Subsets

Because the 115 remaining categories have at least one positive training example each, in principle they can all be used in experiments. However, several researchers have preferred to carry out their experiments on different subsets of categories. Globally, the three subsets that have been most popular [18] are :

- R(10) : the set of the 10 most populated categories, that is the categories with the highest number of positive training examples
- R(90): the set of 90 categories with at least one positive training example and one test example ([18] asserts R(90) to be the most frequently chosen subset)
- R(115): the set of 115 categories with at least one positive training example

[18] show how this collection has several characteristics that make it interesting for TC experimentation: similar to many other applicative contexts, it is multilabel, i.e., each document d_i may belong to more than one category; the set of categories is not exhaustive, i.e., some documents belong to no category at all; the distribution of the documents across the categories is highly skewed, in the sense that some categories have very few documents classified under them (positive examples) while others have thousands; there are several semantic relations among the categories (e.g., there is a category WHEAT and a category GRAIN, which are obviously related), but these relations are hidden (i.e., there is no explicit hierarchy defined on the categories). This collection is also fairly challenging for TC systems based on machine learning (ML) techniques, because several categories have (under any possible split between training and test documents) very few positive training examples, making the inductive construction of a classifier a hard task. All of these properties have made REUTERS-21578 the benchmark of choice for TC research in the past years.

3.3 OHSUMED Corpus

The OHSUMED documents[2] are titles or title-plus-abstract from medical journals (Ohsumed actually consists of a subset of the Medline document base); the categories are the terms of the MeSH thesaurus [2]. This test collection was created to assist information retrieval research. It is a clinically-oriented Medline subset,

[2]publicly available from ftp://medir.ohsu.edu/pub/ohsumed

consisting of 348,566 references (out of a total of over 7 million), covering all references from 270 medical journals over a five-year period (1987-1991). The test database is about 400 megabytes in size. A number of fields normally present in the Medline record but not pertinent to content-based information retrieval have been deleted. The only fields present include the title, abstract, MeSH indexing terms, author, source, and publication type. The original test collection was subsequently used in experiments with the SMART retrieval system [29].

From the 50,216 documents in 1991 which have abstracts, a commonly used split [31] consists in using the first 10,000 for training and the second 10000 for testing. The classification task considered here is to assign the documents to one or multiple categories of the 23 MeSH *diseases* categories. In chapter9 we will refer to this subsets of documents and categories, thus, it will be possible to make a direct comparison with the results obtained in [31]. Other used the OHSUMED collection for TC experiments, e.g., [63], but the employed document set and categories vary. How observe [44], literature results can give an indication of the magnitude order of the OHSUMED performance. For instance, from the fact that accuracy does not overcome 70% in all results obtained in different portion of OHSUMED , it possible to argue that this corpus is more difficult than REUTERS-21578 , for which classifiers reaches 86% of accuracy. This is a confirm that here, differently from REUTERS-21578 that is characterized by a rather direct correspondence between words and categories (for the category “wheat” for example, the occurrence of the word “wheat” in a document is an very good predictor) the connection between words and categories is less direct [31].

Part II

Related Work

As mentioned in the introduction, this dissertation concerns with OLEX, a technique for the automatic learning of rule-based classifiers which exploits negative information (i.e., absence of terms). In this part we present some related works concerning rule-based approaches that can be found in the literature as well as methods exploiting negative information.
The part is organized as follows.

- In Chapter 4 we analyze a new classification method for automatic text categorization introduced in [5] that borrows from market basket analysis techniques using association rule mining in the data-mining field. Then we describe decision trees (e.g. in C4.5 [48]), we provide a description of a good example of rule-based method, that is the popular rule-induction algorithm RIPPER [16; 14], and finally we briefly discuss about TRIPPER (Taxonomical Ripper) [55], a RIPPER extension exploiting knowledge in the form of taxonomies.

- In Chapter 5 we investigate the use of negative information in previous approach: we first discuss about a proposal of a variant of the *k*-NN approach, taking into account evidence provided by negative training instances [25], and then we illustrate a recent feature selection approach [64] for text categorization that *explicitly* considers negative features.

- In Chapter 6 we describe a new framework that uses different types of association rules, positive and negative, for classifying *structured data* and then we analyze an associative text classifier exploiting both presence and absence of words to perform classification recently devised by [9].

Chapter 4

Rule-Based Classifiers

Many text classifiers that have been proposed in the literature using machine learning techniques, as probabilistic models or SVM, are essentially quantitative (i.e. numeric), and as such have sometimes been criticized since, effective as they may be, are not readily interpretable by humans. A class of algorithms that do not suffer from this problem are symbolic (i.e. non-numeric) algorithms, among which inductive rule learners and decision tree inducers are the most important examples.

Rule-based classifiers learn by inferring a set of rules (a disjunction of conjunctions of atomic tests) from pre-classified documents. Rule-based systems enjoy a number of desirable properties, the former of which the fact that people can read classifiers, understand and modify them. Certain types of prior knowledge can also be easily communicated to rule-based systems. Recent studies in the data mining community proposed new methods for classification employing association rule mining [37; 39]. These classification systems discover the strongest association rules in the database and use them to build a categorizer.These associative classifiers have proven to be powerful and achieve high accuracy. However, they were initially implemented and tested only on small numerical datasets from the UCI archives[1].

In this chapter we analyze a new classification method for automatic text categorization introduced in [5] that borrows from market basket analysis techniques using association rule mining in the data-mining field. Then we describe decision trees (e.g. in C4.5 [48]), we provide a description of a good example of rule-based method, that is the popular rule-induction algorithm RIPPER [16; 14], and finally we briefly discuss about TRIPPER (Taxonomical Ripper) [55], a

[1] University of California Irvine knowledge discovery in databases archive. http://kdd.ics.uci.edu/.

RIPPER extension exploiting knowledge in the form of taxonomies.

4.1 Association Rule Mining

Association Rules Generation

Association rule mining is a data mining task that discovers relationships among items in a transactional database. Association rules have been extensively studied in the literature. The efficient discovery of such rules has been a major focus in the data mining research community. From the original *apriori* algorithm [3] there has been a remarkable number of variants and improvements culminated by the publication the FP-Tree growth algorithm [26]. However, most popular algorithms designed for the discovery of all types of association rules, are apriori-based.
Formally, association rules are defined as follows.

Definition 4.1 Let $I = \{i_1, i_2, \ldots, i_m\}$ be a set of items; let D be a set of transactions, where each transaction T is a set of items such that $T \subseteq I$. A transaction T is said to contain X, a set of items in I, if $X \subseteq T$. An *association rule* is an implication of the form:

$$X \Rightarrow Y$$

where $X \subseteq I$, $Y \subseteq I$ and $X \cap Y = \emptyset$.

Definition 4.2 The rule $X \Rightarrow Y$ has a *support* s in the transaction set D if $s\%$, of the transactions in D contain $X \cup Y$.

In other words, the support of the rule is the probability that X and Y hold together among all the possible presented cases.

Definition 4.3 It is said that the rule $X \Rightarrow Y$ holds in the transaction set D with *confidence* c if $c\%$, of transactions in D that contain X also contain Y.

In other words, the confidence of the rule is the conditional probability that the consequent Y is true under the condition of the antecedent X. The problem of discovering all association rules from a set of transactions D consists of generating the rules that have a support and confidence greater than given thresholds. These rules are called *strong rules*.

The main idea behind apriori algorithm is to scan the transactional database searching for k-itemsets (k items belonging to the set of items I). As the name of the algorithm suggests, it uses prior knowledge for discovering frequent itemsets in the database. The algorithm employs an iterative search and uses k-itemsets discovered to find (k+1) itemsets. The frequent itemsets are those that have the support higher than a minimum threshold.

Associative Classifiers

Besides the classification methods described above, recently a new method that builds associative general classifiers has been proposed. In this case the learning method is represented by the association rule mining. The main idea behind this approach is to discover strong patterns that are associated with the class labels. The next step is to take advantage of these patterns such that a classifier is built and new objects are categorized in the proper classes. Two such models were presented in the literature: CMAR [37] and CBA [39]. Although both of them proved to be effective and achieve high accuracy on relatively small UCI datasets [19], they have some limitations. Both models perform only single-class classification and were not implemented for text categorization. In many applications, however, and in text categorization in particular, multiple class classification is required.

Building an Associative Text Classifier

A method to build a categorization system that merges association rule mining task with the text classification problem is devised in [5]. Once the entire set of rules has been generated, some pruning techniques are applied for reducing the set of association rules found in the text corpora. Then the association rules set is used in the prediction of classes for new documents. details on the process are given in the subsections below. If a document D_i is assigned to a set of categories $C = \{c_1, c_2, \ldots, c_m\}$ and after word pruning the set of terms $T = \{t_1, t_2, \ldots, t_n\}$ is retained, the following transaction is used to model the document $D_i : \{c_1, c_2, \ldots, t_1, t_2, \ldots, t_n\}$ and the association rules are discovered from such transactions representing all documents in the collection. The association rules are, however, constrained in that the antecedent has to be a conjunction of terms from T, while the consequent of the rule has to a member of C.

Association Rule Generation

In [5], the algorithm introduced, takes advantage of the apriori algorithm to discover frequent term-sets in documents. Eventually, these frequent itemsets associated with text categories represent the discriminate features among the documents in the collection. The association rules discovered in this stage of the process are further processed to build the associative classifier. Using the apriori algorithm on transactions representing the documents would generate a very large number of association rules, most of them irrelevant for classification. Apriori-based algorithm used in [5] is guided by the constraints on the rules to discover. While building a classifier, the interesting rules are those ones that indicate a category label, rules with a consequent being a category label.
In other words, given the document model described above, rules of the form:

$$T \Rightarrow c_i$$

where $T_j \subseteq T$ and $c_i \subseteq C$. are discovered.

To discover these interesting rules efficiently the rule shape constraint is pushed in the candidate generation phase of the apriori algorithm in order to retain only the suitable candidate itemsets. Moreover, at the phase for rule generation from all the frequent k-itemsets, the rule shape constraint is used again to prune those rules that are of no use in the classification.

[5] present two approaches considered in building an associative text classifier. The first one ARC-AC (Association Rule-based Classifier with All Categories) is to extract association rules from the entire training set following the constraints discussed above. As a result of discrepancies among the categories in a text collection of a real-world application, the authors discovered that it is difficult to handle some categories that have different characteristics (small categories, overlapping categories or some categories having documents that are more correlated than others). Therefore a second solution ARC-BC (Associative Rule-based Classifier By Category) is proposed to solve such problems. In this latter approach each set of documents belonging to one category is considered as a separate text collection to generate association rules from. If a document belongs to more than one category this document will be present in each set associated with the categories that the document falls into.

A serious drawback of ARC-BC algorithm is the high number of rules composing the classifier that are generated. Although the rules are human readable and understandable if the amount of rules generated is too large it is time consuming to read the set of rules for further tuning of the system. This problem leads

to the introduction of pruning methods. Although the rules are similar to those produced using a rule-based induced system, the approach is different. In addition, the number of words belonging to the antecedent could be large (in some experiments carried out by [5] up to 10 words), while in some studies with rule-based induced systems, the rules generated have only one or a pair of words as antecedent.

Pruning the Set of Association Rules

The number of rules that can be generated in the association rule mining phase could be very large. Because such a huge amount of rules could contain noisy information which would mislead the classification process and make the classification time longer [5] present some pruning methods based on the definition of more general rule and higher ranked rule: eliminate the specific rules and keep only those that are more general and with high confidence, and prune unnecessary rules by database coverage.

Prediction of Classes Associated with New Documents

The set of rules selected after the pruning phase represent the actual classifier. This categorizer is used to predict with which classes new documents are labeled. Given a new document, the classification process searches in this set of rules for finding those categories that are the closest to be assigned to the document presented for categorization by employing a dominance factor (proportion of rules of the most dominant category in the applicable rules for a document to classify).

Experimental results reported in [5] show that the association rule-based classifier performs well and its effectiveness is comparable to most well-known text classifiers. One major advantage of the association rule-based classifier is its relatively fast training time. The drawback lies in the huge set of rules generated that have to be submitted to a time-consuming phase of pruning. Notwithstanding this, the use of associative rules to text classification introduced in [5] is interesting as rules generated are understandable and can easily be manually updated or adjusted if necessary.

4.2 Decision Trees

A decision tree text classifier consists of a tree in which internal nodes are labelled by terms, branches departing from them are labelled by tests on the weight that the

term has in the representation of the test document, and leaf nodes are labelled by (not necessarily different) categories. Such a classifier categorizes a test document d_j by recursively testing for the weights that the terms labeling the internal nodes have in the representation of d_j , until a leaf node is reached; the label of this leaf node is then assigned to d_j. Most such text classifiers assume a binary document representation, and thus consist of binary trees.

There are a number of standard packages around for the induction of a decision tree from a training set, and most decision tree approaches to TC have made use of one such package. Among the most popular packages are ID3 (used in [22]), C4.5 (used in [15; 17; 31; 35]) and C5 (used in [38]).

A possible procedure for the induction of a decision tree for category c_i from a set of training examples consists in a *divide and conquer* strategy of recursively:

(i) checking whether all the training examples have the same label (either c_i or $\overline{c_i}$);

(ii) if not, selecting a term t_k, partitioning the training examples into classes of documents that have the same value for t_k, and placing each such class in a separate subtree.

The process is recursively repeated on the subtrees until each leaf node of the tree so generated contain training examples assigned to the same category c_i, which is then chosen as the label for the leaf node. The key step of this process is the choice of the term t_k on which to operate the partition, a choice which is generally made according to an information gain or entropy criterion. However, such a "fully grown" tree may be prone to overfitting, as some branches may be excessively specific to the training data. Any decision tree induction method thus includes a method for growing the tree and one for pruning it, i.e. for removing the overly specific branches so as to minimize the probability of misclassifying test documents. Variations on this basic schema for tree induction abound.

4.3 Inductive Rule Learning Methods

A classifier for category c_i built by an inductive rule learning method consists of a disjunctive normal form (DNF) rule, i.e. of a conjunction of conditional formulae ("clauses"). Clause premises denote the presence or absence of terms in the test document, while the clause head denotes the decision whether to classify it or not under c_i. DNF induction methods are known to be of equal power to decision tree

methods from machine learning theory. However, one of their advantages is that they tend to generate more compact classifiers than decision tree inducers.

Rule induction methods usually attempt to select from all the possible covering rules (i.e. those rules that correctly classify all the training examples) the "best" one according to some minimality criterion. While decision trees are typically induced through a top-down, "divide-and-conquer" strategy, DNF rules are often induced in a bottom-up fashion.

At the start of the induction of the classifier for c_i, every training example is viewed as a clause $\tau_1, \ldots, \tau_n \rightarrow \gamma_i$, where $\tau_1, \ldots, \tau_n$ are the terms contained in the document and γ_i equals c_i or $\overline{c_i}$ according to whether the document is a positive or negative example of c_i. This set of clauses is already a DNF classifier for c_i, but obviously scores tremendously high in terms of overfitting. The induction algorithm employs then a process of generalization whereby the rule is simplified through a series of modifications (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. 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. Individual DNF rule learners vary widely in terms of the methods, heuristics and criteria employed for generalization and for pruning.

In the following sections we illustrate a popular inductive rule learner RIPPER and its extension with taxonomic knowledge TRIPPER .

4.3.1 RIPPER

RIPPER (***R**epeated **I**ncremental **P**runing to **P**roduce **E**rror **R**eduction*), was proposed by [14]. It consists of two main stages:

- the first stage constructs an initial ruleset using a rule induction algorithm called IREP*(*Incremental Reduced Error Pruning*) [24];
- the second stage further optimizes the ruleset initially obtained.

These stages are repeated for k times. IREP*[14] is called inside RIPPER -k for k times, and at each iteration, the current dataset is randomly partitioned in two subsets: a growing set, that usually consists of 2/3 of the examples, and a pruning set, consisting in the remaining 1/3. These subsets are used for two different purposes: the growing set is used for the initial rule construction (the rule growth

phase) and the pruning set is used for the pruning (the rule pruning phase). IREP* uses MDL[49] as a criterion for stopping the process.

The rule growth phase

The initial form of a rule is just a head (the class value) and an empty antecedent. At each step, the best condition based on its information gain is added to the antecedent. The stopping criterion for adding conditions is either obtaining an empty set of positive instances that are not covered or not being able to improve the information gain score.

The rule pruning phase

Pruning is an attempt to prevent the rules from being too specific. Pruning is done accordingly to a scoring metric denoted by $v*$. IREP* chooses the candidate literals for pruning based on a score which is applied to all the prefixes of the antecedent of the rule on the pruning data. The score is defined as follows:

$$v*(rule, prunepos, pruneneg) = \frac{p-n}{p+n} \tag{4.1}$$

where *p* (*n*) denotes the total number of positive (negative) instances covered by the rule.

4.3.2 TRIPPER

TRIPPER [55] is a taxonomy-based extension of RIPPER . The key ingredients of TRIPPER are:

1. the use of an augmented set of features based on taxonomies defined over values of the original features (WordNet in the case of text classification) in the growth phase

2. the replacement of pruning, as an overfitting avoidance method, with the more general method of abstraction guided by a taxonomy over the features.

Before discussing these issues, we need to provide a formally definition of taxonomy.

Definition 4.4 Let $S = v_1, v_2, ...v_n$ be a set of feature values. Let T be a directed tree where *children(i)* denotes the set of nodes that have incoming arrows to the node *i*. A node *i* is called *leaf* if it has no children. A *taxonomy Tax(T,S)* is a mapping which assigns to a node *i* of the tree *T* a subset *S* of *S* with the following properties:

$$Tax(T,S)(i) = \bigcup_{j \in children(i)} Tax(T,S)(j) \tag{4.2}$$

$$Leaves(T) = S \tag{4.3}$$

Improvement at Rule Growth Phase

Introducing the taxonomical knowledge at the rule-growth phase is a straightforward process we call feature space augmentation. The augmentation process takes all the interior nodes of the attribute value taxonomy and adds them to the set of candidate literals used for the growth phase.

Improvement at Rule Pruning Phase

A more general version of feature selection than pruning is abstraction: in the case of abstraction, instead of casting the problem as a matter of preserving or discarding a feature, we are able to choose from a whole range of levels of specificity for the feature under consideration.
The algorithm devised in [55] uses exactly this idea to incrementally search for useful abstractions for the literals in the suffix to be pruned according to the *v** score of the rule prefixes.

Comparison with RIPPER

Experiments reported in [55], were performed on the benchmark dataset REUTERS-21578 using the *ModApte split* [8] of training and testing data. Only the ten biggest classes in the dataset were used (i.e. REUTERS-21578 subset called R(10) in [18], as shown in Chapter 3) and only the 300 best features were used as inputs to the classifier. The experiments compare RIPPER with TRIPPER . The text-specific taxonomies used for experiments on the REUTERS-21578 dataset comes from WordNet[1], using only the hypernymy relation that stands for "isa" relation between concepts.[55] experiments show that TRIPPER outperforms, or performs as well as RIPPER in terms of break-even point on the REUTERS-21578 dataset in a majority (8 out of 10) of classes; TRIPPER generates more

abstract (and often more comprehensible) rules than RIPPER : some of the abstract literals discovered to be important for a third of the 10 classes. Furthermore, the rules generated by TRIPPER are often more concise than those generated by RIPPER .
The usefulness of abstraction is confirmed by the prevalence of abstract literals in almost all the rules of every rule set. Both of the phases (growth and pruning) generated improvements, lending empirical support for the idea that both of the extensions are useful.

The experiments reported in [55] show that TRIPPER generally outperforms RIPPER on the REUTERS-21578 text classification task in terms of break-even points, while generating potentially more comprehensible rule sets than RIPPER . The additional computation cost of TRIPPER is small when compared with RIPPER , consisting in an additional multiplicative factor that represents the height of the largest taxonomy, which in the average case scales logarithmically with the number of feature values.

Chapter 5

Use of Negative Information in TC

Several approaches (either rule-based or not) exploiting negative information for text classification can be found in the literature, such as those in [58; 25; 6; 64]. In this chapter we focus our attention on some of them. In Section 5.1 we analyze *k*-NN$_{neg}$, a proposal of a variant of the *k*-NN approach, taking into account evidence provided by negative training instances [25], while in Section 5.2 we illustrate a recent feature selection approach [64] for text categorization that *explicitly* considers negative features.

5.1 A variant of *k*-NN using Negative Evidence

In [25], the authors observe that the basic philosophy underlying *k*-NN and all the instance-based algorithms in the TC literature, is that only positive instances of a category are used as evidence towards the fact that document d_j belongs to category c_i.

Therefore they first propose a variant of the *k*-NN approach, called *k*-NN$_{neg}$, with the purpose of taking into account also evidence provided by negative training instances, and then devise a variant, called *k*-NN$^{p}_{neg}$, with the intention of appropriately limiting negative contribution of very dissimilar negative training instances.

5.1.1 Instance Based Classifier Induction

One of the most popular paradigms for the inductive construction is the *instance-based* approach, which is well exemplified by the *k*-NN algorithm (used e.g. by [59]).

For deciding whether a document d_j should be classified under a category c_i, *k*-NN select the k training documents most similar to d_j. Those documents d'_z that belong to c_i are seen as carrying evidence towards the fact that d_j also belongs to c_i, and the amount of this evidence is proportional to the similarity between d'_z and d_j. Classifying a document with *k*-NN this means computing

$$CSV_i(d_j) = \sum_{d'_z \in Tr_k(d_j)} RSV(d_j, d'_z) \cdot v_{iz} \tag{5.1}$$

where

- $CSV_i(d_j)$ refers to *Categorization Status Value* and measures the computed evidence that d_j belongs to c;
- $RSV(d_j, d'_z)$ refers to *Retrieval Status Value* and measures the semantic relatedness between d_j and d'_z;
- $Tr_k(d_j)$ represents the set of the k training documents d'_z with the highest $RSV(d_j, d'_z)$;
- the value of v_iz is defined as follows $v_iz = \begin{cases} 1 & \text{if } d'_z \text{ is a positive instance of } c_i \\ 0 & \text{if } d'_z \text{ is a negative instance of } c_i \end{cases}$

The threshold k is usually determined experimentally on a validation set. [61; 62] has found a range value of $30 \leq k \leq 45$ to provide the best effectiveness.

5.1.2 Negative Evidence in Instance-Based Classifiers

In the *k*-NN$_{neg}$ method proposed in [25], the equation 5.2 is modified considering new values for v_iz

- $v_iz = \begin{cases} 1 & \text{if } d'_z \text{ is a positive instance of } c_i \\ -1 & \text{if } d'_z \text{ is a negative instance of } c_i \end{cases}$

In the *k*-NN$_{neg}$ method, if a document d'_z is a negative instance of c_i, this fact is not ignored anymore like in original *k*-NN, and it is instead used as evidence of the fact that d_j does not belongs to c_i.

[25] carry out a systematic experimentation on REUTERS-21578 benchmark, by comparing their classifier induction method with the standard *k*-NN at different values of threshold k. They observe that the devised variant *k*-NN$_{neg}$ performs as

well as the original method *k*-NN but no better than it, and has moreover the drawback of being less robust than the original version because of greater sensitivity to the choice of threshold k. In fact *k*-NN$_{neg}$ reaches its peak of effectiveness for $k = 10$ and therefore a limited number of training documents similar to the test document is needed (while *k*-NN reaches a slight higher effectiveness for $k = 50$), but it degrades somehow for values of k greater than 10 (while *k*-NN performances rise with higher values of k). A valuable interpretation of these results given by the authors is that negative evidence brought by training documents very dissimilar from the test document may be self-defeating. Even in other works [31] the "least negative instances" (i.e. the negative instances most similar to the positive ones) are considered the most interesting ones, because they are the the most hard to separate from the positive instances. Moving from this observations, authors then propose a variant of *k*-NN$_{neg}$ with the intention of downplaying the influence of the "most negative instances"(i.e. the negative instances most dissimilar to the positive ones).

This variant of *k*-NN$_{neg}$, called *k*-NN$^{p}_{neg}$, is defined in [25] by modifying equation as follows:

$$CSV_i(d_j) = \sum_{d'_z \in Tr_k(d_j)} RSV(d_j, d'_z)^p \cdot v_{iz} \tag{5.2}$$

By raising component RSV to power of parameter p, it is possible to appropriately de-emphasizing the importance of very dissimilar training instances. Preliminary performed experiments confirmed authors'intuition: *k*-NN$^{p}_{neg}$ method (provided with a careful choice of level of de-emphasizing represented by the parameter p) consistently outperforms both standard *k*-NN and *k*-NN$^{1}_{neg}$.

5.2 A use of Negative Information in Feature Selection

In [64] is proposed a feature selection approach for text categorization that *explicitly* considers negative features. It constructs a feature set for each category by first selecting a set of terms highly indicative of *membership* as well as another set of terms highly indicative of *non-membership*, then unifying the two sets. The size ratio of the two sets is empirically chosen to obtain optimal performance. This is in contrast with the standard local feature selection approaches that either

- only select the terms most indicative of membership or

- implicitly but not optimally combine the terms most indicative of membership with non-membership.

[64] conduct systematic experiments on several feature selection metrics: chi-square, correlation coefficient, odds ratio, and a coefficient called GSS, proposed in [25], that is a simplified variant of the chi-square. Their results, compared to those ones of standard approaches, show that the proposed approach, considering improves text categorization performance.

Chapter 6

Associative Classifiers with Negation for Structured Data and Texts

Associative classifiers for structured data has been recently introduced in literature. An associative classifier is a small set of classification rules which form a model of the data. The classifier is used to assign a class label to new data for which the class label is unknown. A notable feature of associative classifiers is the understandability of the generated model, which can be read and interpreted also by a human being. The first proposals of associative classifiers differ only in the way rules are ranked and selected in the model.

Traditional associative classification techniques, when applied to text categorization, yield a classifier which is usually characterized by low precision and high recall.

Recently have been devised new techniques, for classifying structured data and texts, aimed at improving traditional associative classification models taking advantage of *negative information*.

In this chapter, we analyze two interesting approaches proposed in the last years in this direction:

- the first, called ARC-PAN, consists of a new framework that uses different types of association rules, positive and negative, for classifying *structured data*. Negative association rules of interest are rules that either associate negations of attribute values to classes or negatively associate attribute values to classes.

- the second, called NeW, considers in the text classification task both presence and absence of words to perform classification to increase the quality of the rules included in the text classifier.

6.1 Associative Classifiers with Negation

Recent studies in the data mining community proposed new methods for classification employing association rule mining [9, 10, 2, 3]. These associative classifiers have proven to be powerful and achieve high accuracy. However, they were only discovering and using positive association rules in the classification process. In this section we analyze and discuss the potential of negative association rules in the categorization task introduced in [6].

6.1.1 Negative Association Rules

This section extends terminology introduced in the previous chapter for association rules with the definition of *negative* association rules.

[12] mentioned for the first time the notion of negative relationships in the literature. Their model is chisquare based. They use the statistical test to verify the independence between two variables. To determine the nature (positive or negative) of the relationship, a correlation metric was used.

In [39] the authors present a new idea to mine strong negative rules. They combine positive frequent itemsets with domain knowledge in the form of a taxonomy to mine negative associations. However, their algorithm is hard to generalize since it is domain dependant and requires a predefined taxonomy. A similar approach is described in [19].

[58] derived a new algorithm for generating both positive and negative association rules. They add on top of the support-confidence framework another measure called mininterest (the argument is that a rule $A \Rightarrow B$ is of interest only if $supp(A \bigcup B) - supp(A)supp(B) \geq mininterest$). Although they introduce the "mininterest" parameter, the authors do not discuss how to set it and what would be the impact on the results when changing this parameter.

In [6] is defined as *generalized negative association rule*, a rule that contains a negation of an item (i.e a rule for which its antecedent or its consequent can be formed by a conjunction of presence or absence of terms).

An example for such association would be as follows:

$$A \wedge \neg B \wedge \neg C \wedge D \rightarrow E \wedge \neg F.$$

Deriving such an algorithm that can determine such type of associations is not an easy problem, since it is well known that the itemset generation in the association rule mining process is an expensive one.

It would be necessary not only to consider all items in a transaction, but also all possible items absent from the transaction. There could be a considerable exponential growth in the candidate generation phase. This is especially true in datasets with highly correlated attributes. That is why it is not feasible to extend the attribute space by adding the negated attributes and use the existing association rule algorithms.

To avoid this problem, in [6] authors consider the generation and use in the classification process of a subset of the generalized negative association rules and refer to them as *confined negative association rules*.

A confined negative association rule is one of the follows:

$\neg X \rightarrow Y$ or
$X \rightarrow \neg Y$

where the entire antecedent or consequent must be a conjunction of negated attributes or a conjunction of non-negated attributes.

As we will see in the third part of this dissertation, this restriction is not introduced in our approach, that allows both negative and positive items (i.e. in our terminology negative and positive discriminating terms) in the antecedent (i.e., according to our notation, in datalog rule body).

6.1.2 Generation of Positive and Negative Rules

The most common framework in the association rules generation is the support-confidence one. Although these two parameters allow the pruning of many associations that are discovered in data, there are cases when many uninteresting rules may be produced. In this section we analyze another framework considered in [6] that adds to the support-confidence some measures based on correlation analysis.

The algorithm proposed in [6] generates a set of rules which is the union of so called PCR (Positive Classification Rules) and NCR (Negative Classification Rules). This set of rules is later used in the classification stage. The algorithm creates only rules of the form:

set_of_features $\rightarrow$ *class_label*.

The algorithm is an apriori-like process. It generates first the set of frequent 1-itemsets. Once the 1-frequent itemsets is generated the candidate sets C_2 to C_n, belonging to a set C keeping all class labels existing in the data set, are found as a join between F_{k-1} and F_1. Those candidates that exceed minimum support threshold are added to the corresponding frequent set. For each candidate the PONERG

	Y	$\neg Y$	$\sum_{row}$
X	f_{11}	f_{10}	f_{1+}
$\neg X$	f_{01}	f_{00}	f_{0+}
$\sum_{col}$	f_{+1}	f_{+0}	N

Figure 6.1: 2x2 Contingency table for binary variables

function (**PO**sitive and **NE**gative **R**ule **G**eneration) is called to generate the positive and negative association rules. The PONERG function generates the positive and negative rules based on the item correlation with a class label. This function takes as input an itemset and the set of class labels. Correlation coefficient measures the strength of the linear relationship between a pair of two variables. It is discussed in the context of association patterns in [53].

Definition 6.1 *For two variables X and Y, the correlation coefficient is given by the following formula:*

$$\rho = \frac{Cov(X,Y)}{\sigma X \sigma Y} \tag{6.1}$$

In this equation, $Cov(X,Y)$ represents the covariance of the two variables and σXstands for the standard deviation.

Let X and Y be two binary variables. Table 6.1 summarizes the information about X and Y variables in a dataset in a 2x2 contingency table. The cells of this table represent the possible combinations of X and Y and give the frequency associated with each combination. N is the size of the dataset considered. The correlation coefficient between the item and the class label is computed in [6] according the following definition (that is a variant of ϕ correlation coefficient introduced by Pearson):

$$\phi = \frac{Nf11 - f_{1+} * f_{f+1}}{\sqrt{f_1 + (N - f_{1+}) f_{+1} (N - f_{+1})}} \tag{6.2}$$

If the correlation in absolute value is greater than the correlation threshold given, than the classification rule is of interest. If the correlation is positive, a positive association rule is discovered. When the correlation is negative, negative rules are generated. Given two items X and Y,

- a *positive* association rule is a rule of the form

$$X \rightarrow Y$$

- a *negative* association rule is one of the follows:

$$\neg X \rightarrow Y$$
or
$$X \rightarrow \neg Y.$$

Once the rules are generated, they are added to PCR or NCR if their confidence exceeds the minimum confidence threshold.

The values for the correlation coefficient are chosen by the authors considering first as correlation threshold the value 0.5, to discover strong correlations. However, there were two datasets where no strong correlations were discovered between attribute values and class labels. For these cases, the threshold was lowered to the 0.3 value to discover moderate correlations.

The set of rules that were generated by algorithm PONERG of [6] represent the actual classifier. This categorizer is used to predict to which classes new objects are attached. Given a new object, the classification process searches in this set of rules for those classes that are relevant to the object presented for classification. The set of positive and negative rules discovered are ordered by confidence and support. This sorted set of rules represents the associative classifier called by [6] ARC-PAN(Association Rule Classification with Positive And Negative).

In [6] the authors observe that the association rules of the type $X \rightarrow C$ and $\neg X \rightarrow C$can be treated in the same way. Both of them have a confidence attached and they have an association with the class label. These types of rules can be considered together and their confidence can be added to the C class total. However, the rules of the type $X \rightarrow \neg C$ have to be treated differently. [6] chose to subtract their confidences from the total confidence of their corresponding class but investigating other methods to score these kind of rules remains an open issue.

[6] tested their proposed algorithm on some datasets from UCI ML Repository [19] and compared with C4.5 [48] and CBA [39]. The results for the classification with classifier based on the positive and negative rules (ARC-PAN) seem encouraging. When all types of rules are used the classification accuracy increases on three datasets when compared with classifier C4.5 and with the CBA. The classification accuracy can be improved as well with only the generation of positive association rules that are strongly correlated. [6] show also as a drastic reduction in rule number takes place when the correlation measure is used to derive inter-

esting rules. Moreover, as observed from the error results presented, the error rate remains in the same range, or even decreases in some cases.

A small set of classification rules is very desirable. When a small set of classification rules is presented, the classification phase is faster, which can be important for some applications. Another advantage is that a small set becomes human readable. It is realistically feasible to read, edit and augment hundreds of rules, but thousands of rules is impractical. Because of the transparency of the associative classifier, manually updating some rules is favorable and practical in many applications.

6.2 Associative Text Classifiers with Negation

In this section we analyze a very recent technique, proposed in [9]: the devised system, called NeW (**Ne**gated **W**ords classifiers), in order to increase the precision of previous approaches defining associative text classifiers, proposes the use of classification rules including *negated words*, i.e. words that the considered document should not contain. Differently from the ARC-PAN technique for structured data, examined in the previous section, in the body of negative rules generated by NeW "positive" and "negative" items can occur together. Classification rules with negated words are in the form "if a document includes words A and B, but not word Z, then it belongs to class C".

As mining classification rules with negated words becomes quickly intractable when decreasing the support threshold, [9] tackle this problem by generating negated words only to specialize rules that may wrongly classify training documents. Hence precision is increased, without losing recall.

6.2.1 Specializing Associative Rules by means of Negated Words

Association rules [4] are rules in the form $X \rightarrow Y$. For classification purposes X is a set of items (words), and Y a class label. [9] says that a set of items X matches a document d if $X \subseteq d$. As usual, the quality of an association rule is measured by its support, given by the number of documents matching $X \cup Y$ over the number of documents in the database, and its confidence, given by the number of documents matching $X \cup Y$over the number of documents matching X.

Absence of items, initially proposed in [12], are introduced in [9] to improve the quality of associative classifiers. Classification rules with negated items are in the form:

$$X\overline{Z} \rightarrow c$$

where X and Z are sets of items (that is words), and c is a class label. Set X contains "positive" items and cannot be empty. Set Z contains "negated" (i.e., absent) items and may be empty. When introducing item negation, the definition of document matching is refined as follows.

Definition 6.2 [9] A set of items $X \cup \overline{Z}$ *matches* a document d if $X \subseteq d$ and $Z \cap d = \oslash$.

The support of a classification rule with negated items $X\overline{Z} \rightarrow c$, analogously to traditional (i.e. positive) association rules, is given by the number of documents matching $X \cup \overline{Z} \rightarrow c$ over the number of documents in the data set. Confidence is given by the number of documents matching $X \cup \overline{Z} \cup c$ over the number of documents matching $X \cup \overline{Z}$.

In order to extract rules with negated words, the inclusion of a negated word w in a transaction *t* when $w \notin t$, albeit a very straightforward solution, causes itemset extraction to become unfeasible also for very high support thresholds.[9] address this issue by first extracting "traditional" classification rules (without negated items) and then by specializing only a subset of the classification rules by selectively adding negated words. The NeW extraction algorithm proposed in [9] is based on the FP-growth [26] association rule mining algorithm, adapted to extract only classification rules. Classification rules are extracted with a different support threshold for each class, in order to appropriately represent also unfrequent classes [40]. We adopted the (simpler) approach of CBA [40]. For each class ci the minimum support threshold minsupi is computed as $minsup_i = freq(c_i) * minsup$, where $freq(c_i)$ is the frequency of c_i in the training data and $minsup$ is the selected minimum support threshold. During rule extraction the χ^2 test is applied to discard negatively correlated and uncorrelated rules.

Only words positively correlated with at least one class label are considered, while all other words are pruned from the documents. Since rules including negatively correlated words are usually not accurate, the quality of generated rules is increased. Furthermore, the complexity of the extraction task is reduced.

The number of classification rules generated by rule extraction can be huge. Hence, this set is usually pruned [37; 39] to select a small subset of high quality rules. During the pruning phase, performed by means of the commonly used database coverage technique [37; 39],[9] apply a specialization technique in order to increase the precision of selected rules. Differently from previous approaches,

before including a rule *r* in the classifier, [9] specialize it with negated words. Specialization occurs only if *r* assigns at least a training document to the wrong class. In this case, *r* is specialized by adding one at a time the most frequent words included in the misclassified documents, but in none of the correctly classified documents. On training data, rules with negated words increase precision (previously misclassified documents might now be covered by a more appropriate rule) without losing recall (the class assignment of all correctly classified documents is not altered). On test data, while negated words still yield an increase in precision, they may cause a negligible recall loss (some correctly assigned documents may not be covered any more because of negated words). Before performing database coverage, a global order is imposed on the rule set. Rules are sorted by decreasing confidence, support, and length (number of items in the rule body). In NeW a training data is removed from the training set (i.e., it is no more considered) when it has been covered by rules passing a database threshold coverage, analogously to [37]. Other approaches (e.g., [39] remove each training data as soon as it is covered by one rule.

Including more than one rule for each training data increases the number of rules in the model. Hence, when classifying a new document, NeW has a wider choice of rules and it may have better chance to correctly classify new documents.

Before including a rule r in the classifier,NeW considers it for specialization if it misclassifies at least one training document. The initial set of candidate negated words contains all words included in at least 2 "wrong documents" $r.wDocs$ (i.e. true negative training documents covered by r) but not included in any of $r.rDocs$ (i.e true positive training documents covered by r). Requiring the absence of these words from a document prevents misclassification of documents by means of r. The most frequent word w, that is the candidate negated word contained in the largest number of documents in the true negative training documents) is added to r as a negated word. If several words have the same frequency, they are considered in lexicographical order. All documents including word w are deleted from $r.wDocs$, because r does not cover them any more. Since w can only be present in wrong documents $r.wDocs$, the classification of documents in $r.rDocs$ is not affected. The set of candidate negated words is computed again on the remaining documents in $r.wDocs$. The specialization process continues until r misclassifies at least one document and at least one negated word is available.

[9] report a set of experimental results performed by NeW on the commonly used ModApte split of the REUTERS-21578 Corpus. The experiments show that

NeW's results are higher or closer to most previous approaches (e.g., SVM [31], K-NN [61], ARC-BC [6], C4.5[48]). The experiments also highlight the importance of negated words on the quality of the classifier. NeW achieves better results than previous associative classifier ARC-BC[6].

The NeW algorithm in [9] has been evaluated by creating a binary classifier for each class c_i and by computing the recall/precision breakeven point (BEP) measure (the point at which precision and recall are equal).

NeW outperforms some previous approaches (Bayes, Rocchio, C4.5 and ARC-BC), while it has performance lower than K-NN and SVM.

[9] remarks how, without negated words, NeW yields a "medium" quality classifier, in which both macro-average BEP, and micro-average BEP are about 5-7% lower than that of NeW with negated words. This quality difference is due to a significant increase in the precision of specialized rules. Rules with negated words allow NeW to retain the high recall characterizing associative classifiers (the number of true positive is rather stable), while precision is increased (the number of false positive decreases significantly).

Part III

OLEX: a System for Automatic Generation of Rule-based Text Classifiers exploiting Negative Information

This part of dissertation describes OLEX, a novel method for the automatic construction of rule-based text classifiers. OLEX relies on an optimization algorithm whereby a set of (both positive and negative) discriminating terms is generated for the category being learned. Such terms are then used to construct a classifier of the form "if term t_1 or ... term t_n occurs in document d, and none of terms $t_{n+1}, \cdots t_{n+m}$ occurs in d, then d belongs to category c". The proposed method is simple and elegant. Despite this, the results of a systematic experimentation performed on both the REUTERS-21578 and the OHSUMED data collections show that OLEX is both effective and efficient.

The part is organized as follows.

- In Chapter 7 we introduce the problem and our terminology.
- In Chapter 8 we present OLEX learning process based on a novel optimization algorithm.
- In Chapter 9 we show the results of a systematic experimentation performed on both the most widely used text categorization test collection.
- In Chapter 10 we will concentrate on the following aspects: the expressiveness of the rules learned by Olex, the way they are learned and their effectiveness.

Chapter 7

Problem Definition and Terminology

In this chapter we introduce terminology used in the remainder of this dissertation and the statement of problem that our devised method is aimed at solving.

7.1 Problem Statement

Throughout the following chapters we assume the existence of

- a finite set $\mathcal{C}$ of categories,
- a finite set $\mathcal{D}$ of documents (called *corpus*)
- the relation $\mathcal{I} \subseteq \mathcal{C} \times \mathcal{D}$ (the *ideal* classification) which says, for each document $d \in D$, the categories of $\mathcal{C}$ to which d belongs.

The corpus $\mathcal{D}$ is partitioned into a

- *training set* TS and a
- *validation set*.

We denote by TS_c the subset of TS whose documents belong to category c according to $\mathcal{I}$ (the *training set of* c).

Now, the problem is that of automatically inducing, for each $c \in \mathcal{C}$, a set of classification rules (the *classifier*, or *categorizer*, of c), by learning the properties of c from TS_c.

Once a classifier has been constructed, its capability to take the right categorization decision is tested by applying it to the documents of the validation set and then comparing the predicted classification to the ideal one. The effectiveness of

Table 7.1: List of the main symbols

Symbol	**Description**
$\mathcal{D}$	corpus of documents
$\mathcal{C}$	set of categories
TS	training set
TS_c	training set of category c
f, k, n	tuning parameters: f is a term scoring function, k the number of terms per category and n the n-gram length
$V(f,k,n)$	reduced vocabulary
t	term (n-gram)
t^+, t^-	positive and negative discriminating terms
X_c	set of discriminating terms for category c
$F(X_c)$	F-measure as a function of X_c

a classifier is measured in terms of the classical notions of *precision*, *recall* and *F-measure* [52].

In table 7.1 it is reported a list of the main symbols used throughout the following chapters.

Chapter 8

Learning Rules with Negation

In this chapter we present OLEX optimization algorithm whereby a set of (both positive and negative) discriminating terms is generated from a set of training documents in order to learn profiles of predefined categories with respect to which we wish to construct text classifiers. Such terms are then used to construct a classifier of the form "if term t_1 or ... term t_n occurs in document d, and none of terms $t_{n+1}, \cdots t_{n+m}$ occurs in d, then d belongs to category c". The proposed method is simple and elegant. Despite this, as we will see in the next chapter, OLEX is both effective and efficient.

OLEX classifier induction process consists of three main steps:

- *Pre-processing*
- *Learning*
- *Classifier Construction.*

The following sections illustrate exhaustively each step.

8.1 Pre-processing

To process documents, we transform them according to the following steps: *Stop word removal*, *Stemming* and *N-gram Extraction*.
Stopword removal. Stopwords are words, such as prepositions, pronouns, etc., that are used to structure phrases rather than provide meaning. We remove them by using a list of common stopwords.
Stemming Stemming is used to reduce words to their root. Porter [46] has defined

Table 8.1: Feature scoring functions used for vocabulary reduction (N is the total number of documents).

Function	Notation	Mathematical form	references
Information gain	$IG(t,c)$	$\frac{A}{N} \times log\frac{N \cdot A}{(A+C)(A+B)} + \frac{C}{N} \times log\frac{N \cdot C}{(A+C)(C+D)}$	[13; 63]
Chi-square	$CHI(t,c)$	$\frac{N(AD-CB)^2}{(A+C)(B+D)(A+B)(C+D)}$	[13; 63]
Odds Ratio	$OR(t,c)$	$\frac{A(N-B)}{B(N-A)}$	[13]

a widely used algorithm for word stemming, and we have incorporated it in our system.

N-gram extraction. According to [13], we define a *n*-gram t (in the following often called "term") as a set $\{s_1, \cdots, s_n\}$ of n word stems. We say that t $occurs$ in a document d if, after stop word removal, d contains a sequence of (consecutive) words whose stems are exactly those in t. Notice that the order of appearance in d of the words giving rise to t is immaterial, so that phrases like "information retrieval" and "retrieving information" map to the same 2-gram $\{inform, retriev\}$. We consider *n*-grams for $n \leq 3$.

8.2 Learning

The learning task is based on two steps, namely, *Vocabulary Reduction* and *Selection of Discriminating Terms*. The input consists of the pre-processed documents of the training set TS, along with the ideal classification $\mathcal{I}$. The output consists, for each category $c \in C$, of a set of discriminating terms.

8.2.1 Vocabulary Reduction by Term Selection

This task is aimed at controlling the dimension of the training data. The advantage is two-fold: first, the search space of our optimization algorithm (see Subsection 8.2.2) is considerably reduced; second, irrelevant or redundant features in the training data, that may degrade the accuracy of the learning phase, are removed. A comparative study of a number of feature scoring functions can be found in [63; 21].

Given a category c, let $\mathcal{T}_c$ be the set of terms occurring in the pre-processed documents of TS_c. With each element $t \in \mathcal{T}_c$ we associate the quantities A, B, C and D defined, according to the two-way contingency table of a term t and a category c, as follows: A is the number of documents in the training set TS_c where t occurs; B the number of documents not in TS_c where t occurs; C the number of documents in TS_c where t does not occur, and D the number of documents not in TS_c where t does not occur. Now, selecting the set $V_c(f, k, n)$ of the k best terms for c, of length $\leq n$, consists in: (1) scoring the terms in $\mathcal{T}_c$, of length $\leq n$, by using the feature scoring function f chosen among those shown in Table 8.1, namely, information gain IG, chi-square CHI and odd ratios OR, and (2) keeping the k terms that score highest.

Definition 8.1 (Reduced Vocabulary) Given a scoring function f and integers k and n ($n \leq 3$), the *reduced vocabulary* of the corpus $\mathcal{D}$ is $V(f, k, n) = \cup_{c \in \mathcal{C}} V_c(f, k, n)$.

As we will see in Section9.1, f, k and n represent the tuning parameters of our approach.
The intuition behind the above definition is that, by selecting the highest scored terms for each category c and putting them all together in $V(f, k, n)$, the reduced vocabulary will contain the terms that are highly indicative of membership in c (i.e., those that have been chosen as representative of c) as well as the terms that are highly indicative of non-membership in c (i.e., those that have been chosen as representative of other categories).

8.2.2 Generation of Discriminating Terms

This task is the heart of our method. Here, the aim is that of devising, for each category $c \in \mathcal{C}$, a set of terms taken from the reduced vocabulary $V(f, k, n)$, some of which are indicative of membership in c, some others of non-membership, capable of "best" characterizing the documents in the training set TS_c of c. Since the whole optimization task consists of $|\mathcal{C}|$ independent sub-tasks (one for each category), in the following we will concentrate on a single category $c \in \mathcal{C}$.

Definition 8.2 (**Discriminating Terms**) We are given the reduced vocabulary $V(f, k, n)$, for fixed values of f, k and n, and a category $c \in \mathcal{C}$. A *discriminating term* (for c) is a pair $< t, s >$, where t is a term in $V(f, k, n)$ and s, the *sign* of t, an element of $\{+, -\}$. We will represent $< t, s >$ as t^s. A discriminating term (often "d-term", for short) with sign "+" (resp. "-") is called $positive$ (resp. $negative$) d-term. We say that t^s *occurs* in a document d if t occurs in

d. Intuitively, a positive d-term for c occurring in d is interpreted as indicative of membership of d in c, while a negative d-term is taken as evidence against membership.

Example 8.3 (Discriminating Terms) Assume that the set of d-terms associated with category c is $X_c = \{t_1^+, t_2^+, t_3^-, t_4^-\}$, where $t_1,\ t_2,\ t_3,\ t_4$ are terms from the reduced vocabulary. The intuitive meaning of X_c is the following: a document d containing either t_1 or t_2, but neither t_3 nor t_4, is classifiable under c; otherwise it is not.

Let $\Delta(t)$ denote the set of documents of the training set TS where term t occurs. Thus, given the set of discriminating terms

$$X_c = \{t_1^+, \cdots, t_n^+, t_{n+1}^-, \cdots, t_{n+m}^-\}$$

the set

$$\Theta(X_c) = \bigcup_{i=1}^{n} \Delta(t_i) \setminus \bigcup_{j=n+1}^{n+m} \Delta(t_j)$$

of the documents containing some positive d-term of X_c and none of the negative d-terms in X_c, represents the set of all documents *classifiable* under c according to X_c. Precision, recall and F-measure w.r.t. c resulting from classifying $\Theta(X_c)$ under c are the following:

$$Pr(X_c) = \frac{|\Theta(X_c) \cap TS_c|}{|\Theta(X_c)|} \tag{8.1}$$

$$Re(X_c) = \frac{|\Theta(X_c) \cap TS_c|}{|TS_c|} \tag{8.2}$$

$$F(X_c) = \frac{2Pr(X_c) \cdot Re(X_c)}{Pr(X_c) + Re(X_c)} = \frac{2 \cdot |\Theta(X_c) \cap TS_c|}{|\Theta(X_c)| + |TS_c|}. \tag{8.3}$$

PROBLEM *DTERM-GENERATION*. We are given: (1) the reduced vocabulary $V(f, k, n)$, for fixed values of f, k and n, (2) the set $\Delta(t)$, for each $t \in V(f, k, n)$, of the documents where t occurs and (3) the training set TS_c of c. Then the problem is finding a set X_c of discriminating terms for c such that $F(X_c)$ is maximum.

Algorithm DTERM-Generation

Input: the reduced vocabulary $V(f, k, n)$, for given values of the tuning parameters f, k and n; the set of documents $\Delta(t)$, for each $t \in V(f, k, n)$, and the training set T_c of c;
Output: a suboptimal set X_c of discriminating terms;
Method: perform the following steps:

1. let $f\_opt = 0$ and $X_c = X = \emptyset$;
2. let $\mathcal{DT} = \{t_i^+, t_i^- \mid t_i \in V(f, k, n)\}$ be the set of all discriminating terms constructible from $V(f, k, n)$;
3. **while** $\mathcal{DT} \neq \emptyset$
4. let $t^s \in \mathcal{DT}$ be the most relevant d-term, i.e., the one which added to X,
5. maximizes the objective function F evaluated by equation 8.3;
6. $X = X \cup \{t^s\}$;
7. $\mathcal{DT} = \mathcal{DT} \setminus \{t^+, t^-\}$;
8. **if** $F(X) > f\_opt$ **then**
9. $f\_opt = F(X)$;
10. $X_c = X$;
11. **end-if**;
12. **end-while**;
13. **return** X_c;

Figure 8.1: Algorithm DTERM-GENERATION

DTERM-GENERATION is a NP-complete problem. Thus, to deal with it, we have defined the greedy heuristics sketched in Figure 8.1. Essentially, the algorithm computes the sequence

$$F(X_0) = 0\ , \cdots ,\ F(X_i) = F(X_{i-1} \cup \{t_i^s\}),\ \cdots ,$$

$$F(X_p) = F(X_{p-1} \cup \{t_p^s\})$$

where $X_0 = \emptyset$; the (approximated) solution provided by the algorithm is a set X_i of d-terms such that $F(X_i) \geq F(X_j)$, for $j = 1, p$. The while-loop (see line 3 of Figure 8.1), whereby the above sequence is generated, is controlled by $\mathcal{DT} = \{t_1^+, t_1^-, \cdots, t_p^+, t_p^-\}$, i.e., the set of all d-terms obtainable from the reduced vocabulary $V(f, k, n)$; at each round, the algorithm picks the d-term $t^s \in \mathcal{DT}$ with the maximum relevance, i.e., the one which maximizes $F(X \cup \{t^s\})$; the evaluation of

F is based on equation 8.3 above. Then, t^s is removed from $\mathcal{DT}$. We notice that the set X_c of d-terms computed by Algorithm DTERM-GENERATION contains at least one positive d-term.

Fact 1 Algorithm DTERM-GENERATION takes time $O(m^2)$, where $m = |V(f, k, n)|$ is the number of terms in the reduced vocabulary (for given values of f, k and n).

Proof. (sketch) We first notice that $|\mathcal{DT}| = 2 \cdot |V(f, k, n)|$, $\mathcal{DT}$ being by definition the set $\{t_i^+, t_i^- \mid t_i \in V(f, k, n)\}$ of all discriminating terms constructible from $V(f, k, n)$ (see Figure 8.1). Thus, the while loop of DTERM-GENERATION is ran in $O(m)$ time, where $m = |V(f, k, n)|$. Now, at each step of this loop, the most "relevant" discriminating term in $\mathcal{DT}$ is chosen by computing, for each element $t^s \in \mathcal{DT}$, the value $F(X \cup \{t^s\})$ of the objective function F. This is clearly done in $O(m)$ time. It turns out that DTERM-GENERATION takes time $O(m^2)$. □

One point that is worth noticing is that, since Algorithm DTERM-GENERATION relies on the reduced vocabulary, the computation of negative d-terms requires the same computational effort as the computation of positive ones.

8.3 Classifier Construction

As we have seen in the previous section, given the set

$$X_c = \{t_1^+, \cdots t_n^+, t_{n+1}^-, \cdots, t_{n+m}^-\}$$

of d-terms for category c, a document d is *classifiable* under c according to X_c if any of $t_1, \cdots, t_n$ occurs in d and none of $t_{n+1}, \cdots, t_{n+m}$ occurs in d. Based on that, we are now in a position to define the *classifier* of c.

Definition 8.4 (**Classifier**) Let

$$X_c = \{t_1^+, \cdots t_n^+, t_{n+1}^-, \cdots, t_{n+m}^-\}$$

be the set of d-terms for category c. The *classifier* of c is the set of (classification) rules $\chi_c = \{r_1, \cdots, r_n\}$, where r_i, $1 \leq i \leq n$, is:

$$c \leftarrow t_i \in d, t_{n+1} \notin d, \cdots, t_{n+m} \notin d.$$

Informally, a rule r_i, $1 \leq i \leq n$, succeeds w.r.t a document d if t_i occurs in d and none of $t_{n+1}, \cdots, t_{n+m}$ occurs in d. Thus, the informal meaning of χ_c is: classify d under c if (and only if) any rule in χ_c succeeds w.r.t. d, i.e., if d is classifiable according to X_c.

Example 8.5 (Classifier) Let $X_c = \{t_1^+, t_2^+, t_3^-, t_4^-\}$ be the set of d-terms of c. Then χ_c consists of the following rules:

$$c \leftarrow t_1 \in d, t_3 \notin d, t_4 \notin d,$$

$$c \leftarrow t_2 \in d, t_3 \notin d, t_4 \notin d.$$

Chapter 9

Experimental Results

Il this chapter we show the results of a systematic experimentation performed on both the most widely used text categorization test collection: the REUTERS-21578 corpus and the OHSUMED corpus. They show that OLEX is both effective and efficient. Another set of experiments was conducted on another texts collection, that is not a benchmark corpus(FCSI), belonging to an American insurance company interested in automatically categorizing documents at present manually labelled by insurance experts.

9.1 Experiments Setup and Performance Metrics

We have experimentally evaluated our algorithm using two well-known benchmark corpora, the REUTERS-21578 and the OHSUMED collections, and a further texts collection, the so called FCSI.
According to our method, we performed on these data sets the following steps:

1. computation of the reduced vocabulary $V(f, k, n)$, for given values of f (scoring function), k (number of terms taken from each category) and n (maximum term length);

2. for each category, execution of the optimizer for the computation of the associated set of d-terms;

3. for each category, generation of the classifier.

We repeated the above three steps for different values of f, k and n, each time testing the classifiers over the validation set. To measure the classification effectiveness, we used the standard definition of micro-averaged precision and recall

(see Chapter 2):

$$\mu Pr = \frac{\Sigma_{c\in\mathcal{C}} TP_c}{\Sigma_{c\in\mathcal{C}}(TP_c + FP_c)};\ \ \mu Re = \frac{\Sigma_{c\in\mathcal{C}} TP_c}{\Sigma_{c\in\mathcal{C}}(TP_c + FN_c)}$$

where TP_c is the number of *true positive* documents w.r.t. c (i.e., the number of documents of the validation set that have correctly been classified under c), FP_c the number of *false positive* documents w.r.t. c and FN_c the number of *false negative* documents w.r.t. c. Based on the above formulas, both the micro-averaged F-measure (with $\beta = 1$) and the micro-averaged break-even point (BEP) have been calculated (the latter as the arithmetic average of micro-averaged precision and recall).

9.2 Reuters

The first data set we considered is the REUTERS-21578 (see Section 3.2. To this corpus we have applied the ModApté split in which 9,603 documents (those having at least one category assigned and are dated earlier than April 7th, 1987) are used to form the training set (TS) and 3,299 (those having at least on category assigned and are dated April 7th, 1987 or later) to form the validation set.
We have conducted a first group of experiments on the ten most populated categories (we will refer to them as R10) listed in table 9.3. To this end, we have:

1. performed the vocabulary reduction by selecting the best terms from the *entire* training set of each category in R10;

2. run the optimization algorithm for each category c in R10 by considering as positive examples *all* documents in its training set TS_c and as negative examples those in $TS \setminus TS_c$;

3. classified w.r.t. the categories in R10 *all* the 3,299 documents in the validation set.

Since the tuning of the algorithm parameters has been carried out over the validation set, the performance results we are presenting must be interpreted as an upper bound to the performance of our approach.
Table 9.1 shows the micro-averaged values for Precision, Recall and F-measure obtained by using the different selection functions shown in Table 8.1 to select vocabularies with sizes ranging from 30 to 6,000, each consisting of n-grams of length ranging between 1 and 3. The following points are worth noticing. First,

Table 9.1: Micro-averaged Precision, Recall and F-measure over R10. The n-gram length is variable, ranging from 1 to 3. Highlighted with bold font, the best performance of each feature scoring function.

Reduced Vocabulary (N. of terms)	**Reduction Function**								
	IG			CHI			OR		
	μPr	μRe	μF	μPr	μRe	μF	μPr	μRe	μF
30	79.62	77.79	78.69	83.22	75.13	78.97	98.85	36.88	53.72
200	86.32	80.41	83.26	87.40	81.13	84.14	96.36	47.3	60.13
400	**88.27**	**84.00**	**86.08**	88.76	83.35	85.97	95.22	50.70	66.17
600	89.21	82.81	85.89	**89.12**	**84.03**	**86.50**	93.04	53.25	67.73
1000	88.70	82.92	85.70	89.72	82.70	86.07	90.63	57.30	70.21
1500	87.61	82.74	85.11	89.09	83.28	86.09	88.79	59.96	71.58
2000	88.68	81.56	84.97	88.50	82.60	85.45	87.19	64.26	73.99
6000	86.76	80.16	83.33	89.04	80.48	80.43	**84.31**	**69.97**	**76.47**

on R10 OLEX achieves a maximum micro-averaged F-measure of 86.50, obtained by using the CHI function and a vocabulary of 600 terms (60 terms per category). Second, functions IG and CHI perform likewise, with a slight predominance of CHI, while OR performs worst. The latter is indeed characterized by high precisions but very poor recalls (contrary to IG and CHI, both characterized by a good balancing of precision and recall values); this stems from the tendency of OR to select very specific terms. Third, reducing the dimension of the vocabulary provides a benefit in terms of categorization performance. For instance, using the CHI function, classifiers perform highest at a vocabulary size of 600 features; likewise, for the IG function only 400 features are needed. We notice that a further reduction of the vocabulary to 3 terms per category entails just a slight loss in categorization performance. Much larger vocabularies are on the contrary needed for the OR function (it performs highest at 6,000 terms), because of its aptitude to be very selective (high precisions but poor recalls). In general, the observed effect of vocabulary reduction is similar to that noted in [63].

In Table 9.2, we show the effect of the n-gram length on the classifier performance. In particular, we compare the results shown in Table 9.1, obtained by using variable-length n-grams (n=1-3), with those obtained by restricting to only one-grams (that is, the classical *bag-of-word* representation). Interestingly, as far as CHI and IG is concerned, the difference between the two cases is quite neg-

ligible, while variable-length n-grams perform significantly better in the case of OR. This is likely due to the following: both IG and CHI naturally tend to select short-length n-grams, so that the selected terms are substantially the same in the case of fixed length n=1 and variable length n=1-3; contrary to this, as already mentioned, OR has a strong tendency to select n-grams of length 3, so that the selected terms are significantly different in the case of fixed length n=1 and variable length n=1-3. Previous results with using multi-word terms show different conclusions. In some cases, e.g. [43], using such terms improve the classification accuracy, while in others, e.g., [45] no positive effects are detected.

Table 9.2: Micro-averaged F-measure over R10 for n-gram length equal to 1 (L=1) and n-gram length ranging between 1 and 3 (L=1-3)

Reduced Vocabulary (N. of terms)	**Reduction Function**					
	IG		CHI		OR	
	L=1	L=1-3	L=1	L=1-3	L=1	L=1-3
200	83.80	83.26	84.42	84.14	54.11	60.13
600	85.43	85.89	86.43	86.50	61.05	67.73
1500	85.18	85.11	86.50	86.09	67.17	71.58

Table 9.3 shows the performance of each of the categories in R10 when the selection function is CHI and the vocabulary size is 600 terms.

Here, for each of the ten categories the size of the respective training set is reported. The columns labelled “Optimization” show the value of the F-measure predicted by the optimizer, along with the number of both positive and negative learned d-terms. We point out that the number of positive d-terms for a classifier coincides with the number of its rules. The columns labelled “Validation” show the Precision, Recall, F-measure and Break Even Point (BPE) values obtained by the learned classifiers on the validation set. As far as optimization is concerned, we emphasize the predominance of negative terms over positive ones (282 against 89), so stressing on the importance of negative information in our category prediction technique. For an instance, for category *earn* the optimization algorithm has learned 11 positive terms (such as “vs”, “loss”, “earn”, “oper year”, etc.) and 31 negative ones (such as “money”, “market open”, “tonn”, “opec”, etc.). So, the classifier learned for *earn* consists of 11 rules of the following type

$$\text{earn} \leftarrow \text{“vs”} \in d, \text{“money”} \notin d, \cdots, \text{“market open”} \notin d$$

Table 9.3: Optimization and validation results for the categories in R10; the feature scoring function is CHI and the vocabulary dimension is 600. The sign $''+''$ means positive d-term, $''-''$ negative d-term. The number of positive d-terms for a classifier coincides with the number of its classification rules.

Category		**Optimization**			**Validation**			
Name	**Size**	F	+	-	Pr	Re	F	BEP
earn	2877	93.90	11	31	97.38	95.86	96.61	96.62
acq	1650	84.88	18	41	92.08	82.47	87.01	87.28
money-fx	538	76.53	14	39	69.44	69.83	69.63	69.64
grain	433	92.82	7	14	92.36	89.26	90.78	90.81
crude	389	81.74	15	17	82.58	77.78	80.11	80.18
trade	369	72.02	1	39	59.78	47.01	52.63	53.40
interest	347	65.58	8	22	62.10	45.04	52.21	53.57
wheat	212	92.27	1	7	86.25	97.18	91.39	91.72
ship	197	85.56	12	15	80.00	76.40	78.16	78.20
corn	181	91.30	2	7	85.00	91.07	87.93	88.04
Total			89	232				
micro-avg					89.12	84.03	86.50	86.58

earn $\leftarrow$ "loss" $\in d$, "money" $\notin d, \cdots$, "market open" $\notin d$

...

expressing that if either term "vs" or term "loss" or ... occurs in d, and none of terms "money" or "market open" or ... occurs in d then classify d under category *earn* (here, each term is represented as an alphabetically ordered sequence of stems). This classifier provides a F-measure for category *earn* equal to 96.61, slightly better than the value predicted by the optimizer (93.90). On this regard, we notice that the predicted F-measure values are highly reliable - the difference between predicted and test values being on the average around $\pm 5\%$.

To conclude this section, we report on the experiments conducted on R90, i.e., the set of the 90 categories with at least one positive training example and one test example. Here, the "best" micro-averaged F-measure is 80.01, obtained by using the CHI function and a reduced vocabulary consisting of 60 terms/category. In table 9.4 we show the F-measure values for all categories of R90 with at least one hundred positive examples. Here, there not exist any correlation between the size of the training set and the respective F-measure value (notice that categories in table 9.4 are listed in decreasing order of size, ranging from 2877 to 101). This

Table 9.4: F-measure for the categories of R90 with at least 100 positive examples

Category	**Size**	F
earn	2877	96.62
acq	1650	87.28
money-fx	538	69.64
grain	433	90.81
crude	389	80.18
trade	369	53.40
interest	347	53.57
wheat	212	91.72
ship	197	78.20
corn	181	88.04
money-supply	138	60.32
dlr	131	50.30
oilseed	124	76.57
sugar	126	86.88
coffee	111	89.66
gnp	101	60.00
micro-avg (all)		80.01

means that our method can learn accurate classifiers even for relatively small categories (this is not the case of other machine learning techniques, e.g., decision tree induction classifiers [30]).

Time Efficiency

The optimization algorithm run times on R10, for two values of the vocabulary size, are reported in Table 9.5.

Those times were obtained on a 2 GHz Pentium 1 Gb RAM.

9.3 OHSUMED

The second data set we considered is OHSUMED (see Section 3.3). In particular, we took into account the collection consisting of the first 20,000 documents from the 50,216 medical abstracts of the year 1991. The first 10,000 were used for training and the second 10,000 for testing. The task considered was to assign

Table 9.5: Optimization run times (in seconds).

Voc Size	**earn**	**acq**	**money**	**grain**	**crude**	**trade**	**interest**	**wheat**	**ship**	**corn**	**TOT**
30	1.20	1.30	2.70	0.20	0.10	3.90	0.50	0.04	2.00	0.10	12.40
600	57.00	82.01	71.86	10.00	13.02	61.87	29.20	28.10	18.90	40.04	412.00

Table 9.6: Experimental results on the 5 most frequent MeSH "diseases" categories of OHSUMED and micro-averaged performance over all 23 categories. The feature scoring function is CHI and the vocabulary dimension is 690 (30 terms/category)

Category		Optimization			Validation			
Name	**Size**	F	+	-	Pr	Re	F	BEP
Pathology	1799	58.27	77	74	41.47	55.92	47.62	48.70
Cardiovascular	1249	80.43	20	51	72.46	72.40	72.43	72.43
Immunologic	525	75.55	13	17	73.57	66.47	69.84	70.02
Neoplasms	1163	80.47	22	25	77.55	79.14	78.34	78.35
Digestive System	588	72.56	23	39	70.45	60.76	65.25	65.61
Total (all)			388	696				
micro-avg (top 5)					60,60	65,99	63,18	63,30
micro-avg (all)					59.62	59.36	59.49	59.49

those documents to one or more categories of the 23 MeSH *diseases*. For lack of space, we report here only a brief summary of the experimental results.

The behavior of our system on OHSUMED confirms the one observed on the REUTERS-21578 and can be summarized as follows: first, functions IG and CHI perform in a similar way (the latter a little better), while OR performs worst; second, reducing the vocabulary size has a positive effect on the performance; indeed, the system achieves (see Table 9.6), with 30 terms per category selected through the CHI function, the best micro-averaged F-measure equal to 59.49 (it is 63.18 over the five most populated categories); third, IG and CHI are substantially invariant w.r.t. the n-gram length, while OR performs better with variable-length n-grams; finally, there is no correlation between the size of the training set and the accuracy of the induced classifier.

9.4 FCSI

The last corpus we considered is the one we called FCSI. Unlike the previous ones, it is not a benchmark corpus, but it is a collection of documents (car accidents reports realized by police officers) belonging to an American insurance company, MCM. The real necessity of the insurance company is that of distinguishing cases in which a dossier represents a case of subrogation (that is, cases in which the company requests to a third party the compensation for damages caused to one of its own insured party) from those which does not. Therefore, in this particular domain we individuated 2 disjoint categories of interest:

- SUBROGATION_YES (or subro_YES)
- SUBROGATION_NO (or subro_NO).

At present, MCM uses a classification system based on rules manually written by insurance experts. The company is interested in our automatic learning system, and therefore we can envisage an interesting proposal of an industrial application for our research results.

For learning the classifiers, we selected, from the whole corpus, which contains about 35,000 documents, the subset of documents already manually classified. The so created training corpus consisted of 2,962 documents. It was randomly split into two parts with a percentage of 60% to form the training set (which thus contains 1,778 documents), and the validation set (which contains 1,184 documents).

In the first phase of experimentation, we followed the flow already outlined for the experiments on the other two corpora, for the purpose of generating the best classifiers for both categories. Like the other two cases, we generated classifiers for different values of the tuning parameters of our model (i.e., scoring function, number of terms and n-gram length). The detailed results are summarized in Table 9.7, Table 9.8 and Table 9.9. It can be noted that the best classifiers for the two categories are obtained using different scoring functions and vocabulary size. In particular, for category subro_YES, the best classifier (F= 93.46) was generated using a vocabulary reduced at 100 terms per category, evaluated through the function OR (Table 9.9). This is likely due to the specificity of the underlying domain, rich of terms particularly faithful to the subject of interest and to the composition of the training set of this category for which a more selective scoring function (which is OR) turns out to be more effective than others. Contrary to this, category subro_NO has a very different behavior; indeed, its best classifier

Table 9.7: Performance values for the classifiers obtained considering ngrams with length variable from 1 to 3 and the selection function CHI

Reduced Vocabulary (N. of terms)	**SUBROG_YES**			**SUBROG_NO**		
	Pr	*Re*	*F*	*Pr*	*Re*	*F*
20	76.02	94.76	84.36	80.82	37.03	50.79
40	75.48	95.47	84.31	84.68	63.60	72.64
60	75.33	95.61	84.27	87.27	78.87	82.86
80	75.76	95.61	84.54	88.94	80.75	84.65
100	75.76	95.61	84.54	88.94	80.75	84.65
120	75.76	95.61	84.54	88.94	80.75	84.65
160	77.26	95.75	**85.52**	86.42	83.89	85.14
300	77.12	95.47	85.32	89.09	82.01	**85.40**

Table 9.8: Performance values for the classifiers obtained considering ngrams with length variable from 1 to 3 and the selection function IG

Reduced Vocabulary (N. of terms)	**SUBROG_YES**			**SUBROG_NO**		
	Pr	*Re*	*F*	*Pr*	*Re*	*F*
20	76.02	94.76	84.36	85.79	68.20	75.99
40	75.48	95.47	84.31	86.28	72.38	78.72
60	75.99	95.47	84.62	85.94	80.54	83.15
80	75.98	95.89	84.78	86.27	84.10	85.17
100	75.81	95.89	84.68	86.85	84.31	85.56
120	77.13	96.03	85.55	90.02	81.17	85.37
160	77.28	95.89	**85.59**	89.20	81.17	85.00
300	78.77	92.49	85.08	90.37	82.43	**86.22**

(F= 86.22) is generated from a vocabulary consisting of 300 terms per category, selected through the function IG (Table 9.8). By using both the above classifiers we obtain a micro-average F-measure equal to 91,00%.

To conclude this section, we remark that, also on the FCSI training corpus, score functions IG and CHI show a similar behavior and that the reduction of the vocabulary brings advantages to the effectiveness of the produced classifier.

Table 9.9: Performance values for the classifiers obtained considering ngrams with length variable from 1 to 3 and the selection function OR

Reduced Vocabulary (N. of terms)	**SUBROG_YES**			**SUBROG_NO**		
	Pr	*Re*	*F*	*Pr*	*Re*	*F*
40	98.79	81.02	89.03	87.88	12.13	21.32
100	96.27	87.82	91.85	86.41	18.62	30.64
200	93.73	93.20	**93.46**	92.31	27.62	42.52
400	89.74	94.19	91.91	86.73	35.56	50.44
600	88.44	93.20	90.76	82.11	37.45	**51.44**

9.5 Final Remarks

The reported experimental results show a number of enjoyable properties of OLEX:

1. its capability to learn accurate classifiers even from small vocabulary sizes; (see table 9.1);

2. the substantial independence of the accuracy of the induced classifiers from the size of the training sets (see tables 9.4 and 9.6);

3. simplicity and readability of classifiers; it is worth noticing that, in the case of R10, OLEX classifiers range from 1 rule (for categories “trade” and “wheat”) to 18 (for “acq”) – see the number of positive terms in Table 9.3; needless to say, small classifiers are highly desirable, as they are both faster and more comprehensible (thus allowing possible human refinement of categories);

4. high accuracy in practical applications (see FCSI experimentation) and

5. high time efficiency.

Further, OLEX shows a uniform behavior on all data sets we considered, which is a noticeable proof of robustness.

Chapter 10

Comparison and Discussion

In this chapter, we will concentrate on the following aspects: the expressiveness of the rules learned by Olex, the way they are learned and their effectiveness.

10.1 Rule Expressiveness

What is original in our approach is the paradigm "one positive term, more negative terms". Intuitively, it seems that positive terms allow us to catch most of the right documents (i.e., they guarantee high recall values), while negative ones help us not to make "too many" mistakes (i.e., they improve the precision of positive terms), thus providing a good discriminatory ability. There is a strict relation between the expressiveness of such rules and the compactness of the induced classifiers.
To our knowledge, no other rule induction methods provide the same rule language. For instance, RIPPER [17] uses the notion of "context" of a word w, that is, a set of other words that must *co-occur* with w within a document d in order for d be classified under a given category. Thus, RIPPER's classification rules are Horn clauses (i.e., rules with only positive literals in their bodies) of the form $c \leftarrow w_1 \in d, \cdots, w_n \in d$, where w_i is a word, $1 \leq i \leq n$. [1]
Associative classifiers, such as CBA [39], CMAR [37] and MCAR [54], have recently been proposed and represent a promising approach. Here, a classification rule is an association rule $X \rightarrow C$, where X is a set of attributes and C a class label. Associative rule induction methods exploiting negative information are de-

[1] Notice that RIPPER can learn rules having (positive) literals stating that a word occurs 0 times in a document. However, on sparse data these literals are not chosen very often by their general purpose literal selection heuristics.

scribed in [6] and [9]. In [6], a negative rule is either $X \rightarrow \neg C$ or $\neg X \rightarrow C$, where $\neg X$ denotes a conjunction of negated attribute; it turns out that the antecedent of a rule consists of either *all* negated or *all* positive attributes (i.e., positive and negative terms cannot occur at the same time). Contrary to this, the approach described in [9] allows associative rules of the form $X, \neg Y \rightarrow C$, where $\neg Y$ is a (possibly empty) conjunction of negated terms.

10.2 How Rules are Learned

Associative Classifier induction methods are essentially based on extensions of the popular association rule mining algorithm *apriori*. For instance, in [6], a rule is generated when the correlation coefficient between the itemset X and the class label C is in absolute greater than the given correlation threshold. In particular, if the correlation is positive (resp. negative) a positive (rep. negative) association rule is generated.
Concerning "traditional" rule-based classifiers, two dominant techniques for rule learning can be found in the literature: decision trees and "divide-and-conquer" methods. Both rely on a two-stage process: first they induce an initial set of rules (the rule growth phase) and then refine them by some optimization techniques (the rule pruning phase). A practical implementation of decision-tree-based approach is C4.5, the archetype of this class of systems; after the decision tree has been transformed into a rule set, C4.5 implements a pruning stage which requires four steps to produce the final rule set - a rather complex and time consuming task (in fact, C4.5 showed to be extremely inefficient in many example sets). A more efficient decision-tree-based rule induction algorithm has recently been reported in [32]. The "divide-and-conquer" algorithms represent a more direct way to generate rules. Practical implementations of this approach are RIPPER [17] and Swap-1 [56]. TRIPPER (Taxonomical RIPPER) [55] extends RIPPER using Wordnet to guide rule induction.
The method proposed in this thesis does not fall in any of the aforementioned schemes. Rather, it relies on a simple, yet effective, optimization algorithm for the computation of discriminating terms. Unlike C4.5 and RIPPER, which requires lengthy optimization processes (the former to discard, the latter to adjust the induced rules), it is a single-step process which does not need any post-induction optimization. The implementation, although at a prototypical level, showed to be very efficient on large data sets.

10.3 Rule Effectiveness

We preliminarily observe that, although many results are available in the literature (e.g. [7; 32]), a direct comparison is often difficult, as different training sets and/or experimental settings have been used. For this reason, we restrict our attention to the results shown in [55; 9; 31], which refer to settings similar to ours. In [55], RIPPER and TRIPPER are assessed on R10. By comparing those results with the ones of Table 9.3 above, we observe that Olex generally outperforms RIPPER, and is competitive with TRIPPER (even though this system makes use of prior knowledge). In [9], the associative classifier NeW is tested on R90, where it achieves a micro-averaged BEP of 81.8 (slightly higher than the 80.01 of Olex). In [31], Joachims compares the performance on R90 of SVMs with four classifiers, obtaining the following break-evens: 86.4 SVM, 82.3 k-NN, 79,9 Rocchio, 79,4 C4.5 (decision tree), 72.2 Bayes Classifiers. As we can see, our approach (BEP=80.01) outperforms Rocchio, C4.5 and Bayes classifiers, but performs worse than SVMs and kNN. We note that K-NN and SVM, albeit yielding better classification results, do not yield an interpretable model that allows a human being to understand the classification result.
Concerning the OHSUMED test set, in [31] Joachims categorizes the same test collection we used in our experimentation to compare the performance of SVMs with four classifiers. According to the reported results, OLEX (BEP = 59.5) performs better than Naive Bayes (57.0), Rocchio (56.6), k-NN (59.1) and C4.5 (50.0), while it is less effective than SVMs (66.0).

Chapter 11

Conclusions

In this dissertation we proposed a novel approach for the automatic construction of rule-based text classifiers. Here, a classifier is a set of classification rules whose antecedent contains one positive literal and zero or more negative literals. Experimental results are very encouraging. In summary, the proposed method enjoys a number of desirable properties:

1. it is based on very simple and straightforward ideas and thus provides a clear intuition of what learning is about;
2. classifiers are readable, easy for people to understand, contrary to most of the other approaches that lack interpretability;
3. despite its simplicity, it can provide high performances in practical applications;
4. it operates efficiently;
5. it is accurate even for relatively small categories (i.e., it is not biased towards majority classes);
6. it is robust, i.e., shows a similar behavior on both data sets we have experimented.

In summary, the main contributions of the work are the following:

1. We have studied methods and systems for automatic text classification, analyze their complexity and their exploitation for a critical comparison.

2. We have designed a new machine learning method for generating logic rules for text categorization.

3. We have implemented our approach in a prototype, the OLEX system.

4. We have performed a systematic experimentation and report experimental results on a number of well-known benchmark text collections to assess the impact of our approach and to compare it with respect to other systems.

5. We have integrated the support for learning process in OLEX Content Management Suite, within project PIA-Exeura-03-06.

Chapter 12

Future Work

In this dissertation we show how OLEX seems to be a very interesting learning technique. Despite the method is simple and elegant, the system is both effective and efficient, as reported in experimental results. However, there is still room for improvements. In particular, by allowing exactly one positive literal, the proposed method suffers from being unable to express co-occurrence based feature dependencies. This might represent a severe limitation to the rule language expressiveness. To overcome this restriction, we are currently involved at devising an efficient and effective algorithm for generating *generalized rules*, i.e., rules supporting conjunctions of positive terms in their bodies.

Bibliography

[1] *WordNet: An Electronic Lexical Database (Language, Speech, and Communication)*. The MIT Press, May 1998.

[2] *MeSH. Medical Subject Headings*. MD: National Library of Medicine, Bethesda, US, 2004.

[3] Rakesh Agrawal, Tomasz Imielinski, and Arun N. Swami. Mining association rules between sets of items in large databases. In Peter Buneman and Sushil Jajodia, editors, *Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data*, pages 207–216, Washington, D.C., 26–28 1993.

[4] Rakesh Agrawal and Ramakrishnan Srikant. Fast algorithms for mining association rules. In Jorge B. Bocca, Matthias Jarke, and Carlo Zaniolo, editors, *Proc. 20th Int. Conf. Very Large Data Bases, VLDB*, pages 487–499. Morgan Kaufmann, 12–15 1994.

[5] Maria-Luiza Antonie and Osmar R. Zaïane. Text document categorization by term association. In *ICDM '02: Proceedings of the 2002 IEEE International Conference on Data Mining (ICDM'02)*, page 19, Washington, DC, USA, 2002. IEEE Computer Society.

[6] Maria-Luiza Antonie and Osmar R. Zaïane. An associative classifier based on positive and negative rules. In *DMKD '04: Proceedings of the 9th ACM SIGMOD workshop on Research issues in data mining and knowledge discovery*, pages 64–69, New York, NY, USA, 2004. ACM Press.

[7] Chidanand Apté, Fred J. Damerau, and Sholom M. Weiss. Automated learning of decision rules for text categorization. *ACM Transactions on Information Systems*, 12(3):233–251, 1994.

[8] Chidanand Apté, Fred J. Damerau, and Sholom M. Weiss. Towards language-independent automated learning of text categorization models. In W. Bruce Croft and Cornelis J. van Rijsbergen, editors, *Proceedings of SIGIR-94, 17th ACM International Conference on Research and Development in Information Retrieval*, pages 23–30, Dublin, IE, 1994. Springer Verlag, Heidelberg, DE.

[9] E. Baralis and P. Garza. Associative text categorization exploiting negated words. In *Proceedings of the 2006 ACM symposium on Applied computing*, pages 530 – 535, 2006.

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

[11] Peter Biebricher, Norbert Fuhr, Gerhard Knorz, Gerhard Lustig, and Michael Schwantner. The automatic indexing system AIR/PHYS. From research to application. In Yves Chiaramella, editor, *Proceedings of SIGIR-88, 11th ACM International Conference on Research and Development in Information Retrieval*, pages 333–342, Grenoble, FR, 1988. ACM Press, New York, US. Reprinted in Karen Sparck Jones and Peter Willett (eds.), "Readings in Information Retrieval", Morgan Kaufmann, San Francisco, US, 1997, pp. 513–517.

[12] Sergey Brin, Rajeev Motwani, and Craig Silverstein. Beyond market baskets: Generalizing association rules to correlations. In Joan Peckham, editor, *SIGMOD 1997, Proceedings ACM SIGMOD International Conference on Management of Data, May 13-15, 1997, Tucson, Arizona, USA*, pages 265–276. ACM Press, 1997.

[13] Maria Fernanda Caropreso, Stan Matwin, and Fabrizio Sebastiani. A learner-independent evaluation of the usefulness of statistical phrases for automated text categorization. In Amita G. Chin, editor, *Text Databases and Document Management: Theory and Practice*, pages 78–102. Idea Group Publishing, Hershey, US, 2001.

[14] William W. Cohen. Fast effective rule induction. In Armand Prieditis and Stuart Russell, editors, *Proc. of the 12th International Conference on Machine Learning*, pages 115–123, Tahoe City, CA, July 9–12, 1995. Morgan Kaufmann.

[15] William W. Cohen and Haym Hirsh. Joins that generalize: text classification using WHIRL. In Rakesh Agrawal, Paul E. Stolorz, and Gregory Piatetsky-Shapiro, editors, *Proceedings of KDD-98, 4th International Conference on Knowledge Discovery and Data Mining*, pages 169–173, New York, US, 1998. AAAI Press, Menlo Park, US.

[16] William W. Cohen and Yoram Singer. Context-sensitive learning methods for text categorization. In Hans-Peter Frei, Donna Harman, Peter Schäuble, and Ross Wilkinson, editors, *Proceedings of SIGIR-96, 19th ACM International Conference on Research and Development in Information Retrieval*, pages 307–315, Zürich, CH, 1996. ACM Press, New York, US. An extended version appears as [17].

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

[18] Franca Debole and Fabrizio Sebastiani. An analysis of the relative hardness of Reuters-21578 subsets. *Journal of the American Society for Information Science and Technology*, 2004. Forthcoming.

[19] C.L. Blake D.J. Newman, S. Hettich and C.J. Merz. UCI repository of machine learning databases, 1998.

[20] B.J. Field. Towards automatic indexing: automatic assignment of controlled-language indexing and classification from free indexing. *Journal of Documentation*, 31(4):246–265, 1975.

[21] George Forman. An extensive empirical study of feature selection metrics for text classification. *Journal of Machine Learning Research*, 3:1289–1305, 2003.

[22] Norbert Fuhr and Chris Buckley. A probabilistic learning approach for document indexing. *ACM Trans. Inf. Syst.*, 9(3):223–248, 1991.

[23] Norbert Fuhr, Stephan Hartmann, Gerhard Knorz, Gerhard Lustig, Michael Schwantner, and Konstadinos Tzeras. AIR/X – a rule-based multistage indexing system for large subject fields. In André Lichnerowicz, editor, *Proceedings of RIAO-91, 3rd International Conference "Recherche d'Information Assistee par Ordinateur"*, pages 606–623, Barcelona, ES, 1991. Elsevier Science Publishers, Amsterdam, NL.

[24] Johannes Furnkranz and Gerhard Widmer. Incremental reduced error pruning. In *Proceedings the Eleventh International Conference on Machine Learning*, New Brunswick, NJ.

[25] Luigi Galavotti, Fabrizio Sebastiani, and Maria Simi. Experiments on the use of feature selection and negative evidence in automated text categorization. In José L. Borbinha and Thomas Baker, editors, *Proceedings of ECDL-00, 4th European Conference on Research and Advanced Technology for Digital Libraries*, pages 59–68, Lisbon, PT, 2000. Springer Verlag, Heidelberg, DE. Published in the "Lecture Notes in Computer Science" series, number 1923.

[26] Jiawei Han, Jian Pei, and Yiwen Yin. Mining frequent patterns without candidate generation. In *SIGMOD '00: Proceedings of the 2000 ACM SIGMOD international conference on Management of data*, pages 1–12, New York, NY, USA, 2000. ACM Press.

[27] P. J. Hayes and S. P. Weinstein. CONSTRUE/TIS: a system for content-based indexing of a database of news stories. In *Proceedings of IAAI90, 2nd Conference on Innovative Applications of Artificial Intelligence*, pages 1–5, 1990.

[28] Philip J. Hayes and Steven P. Weinstein. CONSTRUE/TIS: a system for content-based indexing of a database of news stories. In Alain Rappaport and Reid Smith, editors, *Proceedings of IAAI-90, 2nd Conference on Innovative Applications of Artificial Intelligence*, pages 49–66. AAAI Press, Menlo Park, US, 1990.

[29] William Hersh, Christopher Buckley, T.J. Leone, and David Hickman. OHSUMED: an interactive retrieval evaluation and new large text collection for research. In W. Bruce Croft and Cornelis J. van Rijsbergen, editors, *Proceedings of SIGIR-94, 17th ACM International Conference on Research and Development in Information Retrieval*, pages 192–201, Dublin, IE, 1994. Springer Verlag, Heidelberg, DE.

[30] N. Japkowicz and S. Stephen. The class imbalance problem: a systematic study. *Intelligent Data Analysis Journal*, 6(5):429–449, 2002.

[31] Thorsten Joachims. Text categorization with support vector machines: learning with many relevant features. In Claire Nédellec and Céline Rouveirol, editors, *Proceedings of ECML-98, 10th European Conference on Machine*

Learning, pages 137–142, Chemnitz, DE, 1998. Springer Verlag. Published in the "Lecture Notes in Computer Science" series, number 1398.

[32] David E. Johnson, Frank J. Oles, Tong Zhang, and Thilo Goetz. A decision-tree-based symbolic rule induction system for text categorization. *IBM Systems Journal*, 41(3):428–437, 2002.

[33] David D. Lewis. *Representation and learning in information retrieval*. PhD thesis, Department of Computer Science, University of Massachusetts, Amherst, US, 1992.

[34] David D. Lewis. Evaluating and optmizing autonomous text classification systems. In Edward A. Fox, Peter Ingwersen, and Raya Fidel, editors, *Proceedings of SIGIR-95, 18th ACM International Conference on Research and Development in Information Retrieval*, pages 246–254, Seattle, US, 1995. ACM Press, New York, US.

[35] David D. Lewis and Jason Catlett. Heterogeneous uncertainty sampling for supervised learning. In William W. Cohen and Haym Hirsh, editors, *Proceedings of ICML-94, 11th International Conference on Machine Learning*, pages 148–156, New Brunswick, US, 1994. Morgan Kaufmann Publishers, San Francisco, US.

[36] David D. Lewis and Philip J. Hayes. Guest editors' introduction to the special issue on text categorization. *ACM Transactions on Information Systems*, 12(3):231, 1994.

[37] W. Li, J. Han, and J. Pei. Cmar: Accurate and efficient classification based on multiple-class association rule. In *ICDM'01*, San Jose, CA.

[38] Yong H. Li and Anil K. Jain. Classification of text documents. *The Computer Journal*, 41(8):537–546, 1998.

[39] B. Liu, W. Hsu, and Y. Ma. Integrating classification and association rule mining. In *Proc. of the KDD*, New York, NY.

[40] Bing Liu, Yiming Ma, and Ching Kian Wong. Improving an association rule based classifier. In *Principles of Data Mining and Knowledge Discovery*, pages 504–509, 2000.

[41] M.E. Maron. Automatic indexing: an experimental inquiry. *Journal of the Association for Computing Machinery*, 8(3):404–417, 1961.

[42] T. Mitchell. *Machine learning*. New York, US, 1996.

[43] Dunja Mladenić and Marko Grobelnik. Word sequences as features in text-learning. In *Proceedings of ERK-98, the Seventh Electrotechnical and Computer Science Conference*, pages 145–148, Ljubljana, SL, 1998.

[44] Alessandro Moschitti. *Natural Language Processing and Text Categorization: a study on the reciprocal beneficial interactions*. Phd thesis, University of Rome Tor Vergata, Rome, Italy, 2003.

[45] Alessandro Moschitti and Roberto Basili. Complex linguistic features for text classification: A comprehensive study. In *ECIR*, pages 181–196, 2004.

[46] M.F. Porter. An algorithm for suffix stripping. *Program*, 3(14):130–137, 1980.

[47] J. R. Quinlan. Generating production rules from decision trees. In *Proc. of IJCAI-87*, 304–307, 1987.

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

[49] J. Ross Quinlan. MDL and categorial theories. In *International Conference on Machine Learning*, pages 464–470, 1995.

[50] Hinrich Schütze, David A. Hull, and Jan O. Pedersen. A comparison of classifiers and document representations for the routing problem. In Edward A. Fox, Peter Ingwersen, and Raya Fidel, editors, *Proceedings of SIGIR-95, 18th ACM International Conference on Research and Development in Information Retrieval*, pages 229–237, Seattle, US, 1995. ACM Press, New York, US.

[51] Fabrizio Sebastiani. A tutorial on automated text categorisation. In Analia Amandi and Ricardo Zunino, editors, *Proceedings of ASAI-99, 1st Argentinian Symposium on Artificial Intelligence*, pages 7–35, Buenos Aires, AR, 1999. An extended version appears as [52].

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

[53] P. Tan, V. Kumar, and J. Srivastava. Selecting the right interestingness measure for association patterns, 2002.

[54] F. Thabtah, P. Cowling, and Y. Peng. Mcar:multi-class classification based on association rules. In *Proc. of the 3rd ACS/IEEE Int. Conf. on Computer Systems and Applications*, Cairo, Egypt.

[55] Flavian Vasile, Adrian Silvescu, Dae-Ki Kang, and Vasant Honavar. Tripper: Rule learning using taxonomies. In *10th Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD 2006)*, Lecture Notes in Artificial Intelligence, Singapore, April 2006. Springer Verlag. to appear.

[56] S. Weiss and N. Indurkhya. Optimized rule induction. *IEEE EXPERT*, 8(6):61–69, 1993.

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

[58] X. Wu, C. Zhang, and S. Zhang. Mining both positive and negative association rules. In *Proceedings of 19th International Conference on Machine Learning*, pages 658–665, Sydney, Australia, 2002.

[59] Yang. A comparative study on feature selection in text categorization. In *International Conference on Machine Learning*, pages 412–420. ACL, 1997.

[60] Y. Yang and X. Liu. A re-examination of text categorization methods. In *Proceedings of SIGIR-99, 22nd ACM International Conference on Research and Development In Information Retrieval*, pages 122–130, 1999.

[61] Yiming Yang. Expert network: effective and efficient learning from human decisions in text categorisation and retrieval. In W. Bruce Croft and Cornelis J. van Rijsbergen, editors, *Proceedings of SIGIR-94, 17th ACM International Conference on Research and Development in Information Retrieval*, pages 13–22, Dublin, IE, 1994. Springer Verlag, Heidelberg, DE.

[62] Yiming Yang. An evaluation of statistical approaches to text categorization. *Information Retrieval*, 1(1/2):69–90, 1999.

[63] Yiming Yang and Jan O. Pedersen. A comparative study on feature selection in text categorization. In Douglas H. Fisher, editor, *Proceedings of ICML-97, 14th International Conference on Machine Learning*, pages 412–420, Nashville, US, 1997. Morgan Kaufmann Publishers, San Francisco, US.

[64] Z. Zheng and R. Srihari. Optimally combining positive and negative features for text categorization. In *Proceedings of the ICML, Workshop on Learning from Imbalanced Datasets II*, Washington DC, 2003.

List of Publication

1. Cumbo C., Policicchio V. L., Rullo, P,. Learning Rules with Negation for Text Categorization. *In Proceedings of the 2007 ACM Symposium on Applied Computing* (Seoul, Korea, March 11 - 15, 2007). SAC '07. ACM Press (accepted, to appear).

2. C. Cumbo, S. Iiritano, P. Rullo. Reasoning-based knowledge extraction for text classification. *7th International Conference on Discovery Science*, University of Padova, Padova, Italy, October 2-5, 2004

3. C. Cumbo, W. Faber, G. Greco, N. Leone, Enhancing the Magic-Set Method for Disjunctive Datalog Programs- *ICLP'04, 20th International Conference on Logic Programming*, September 6-10, 2004, Saint-Malo, France.

4. C. Cumbo, S. Iiritano, P. Rullo. OLEX-A reasoning-based text classifier. *Logics in Artificial Intelligence, Ninth European Conference, JELIA'04*, September 27-30, 2004, Lisbon, Portugal.

5. F. Calimeri, M. Citrigno, C. Cumbo, W. Faber, N. Leone, S. Perri, G. Pfeifer. New DLV Features for Data Integration. *Logics in Artificial Intelligence, Ninth European Conference, JELIA'04*, September 27-30, 2004, Lisbon, Portugal.

6. C. Cumbo, S. Iiritano, P. Rullo. Combining logic programming and domain ontologies for text classification. *CILC (Convegno Italiano di Logica Computazionale)* 2004, Parma, 16-17 June, 2004 .

7. C. Cumbo, M. Ettorre, S. Iiritano, M. A. Mastratisi, M. Ruffolo, P. Rullo, A logic-based system for textual document classification - *Workshop OntoLex 2004: Ontologies and Lexical Resources in Distributed Environments*, Lisbon, Portugal, 29th May 2004.

Acknowledgements

I am sincerely grateful to my supervisor prof. Pasquale Rullo who advised me during the years of my PhD course. He guided me throughout the work and generously supported me giving precious suggestions. Most part of the content of this thesis is based on several scientific papers that we wrote together.

Particular thanks are devoted to prof. Nicola Leone for giving me many valuable comments and advices.

Many thanks to my colleagues at the Department of Mathematics of University of Calabria and at Exeura s.r.l. which provided me with useful comments for my research activities.

Among them I would like to thank Veronica L. Policicchio with whom I shared long experimental sessions and constructive discussions during the development of this work.

Moreover I would like to thank Prof. Annamaria Canino for having continuously encouraged me and the other PhD students during these years.