Università della Calabria

Dipartimento di Matematica

Dottorato di Ricerca in Matematica ed Informatica

XX Ciclo

Settore Disciplinare INF/01 INFORMATICA

Tesi di Dottorato

Olex
Effective Rule Learning for Text Categorization

Veronica Lucia Policicchio

Supervisore
Prof. Pasquale Rullo

Coordinatore
Prof.ssa Annamaria Canino

A.A. 2006 - 2007

Olex
Effective Rule Learning for Text Categorization

VeronicaLucia Policicchio

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

Sommario

Le prime ricerche nell'ambito del *Text Categorization*, una sotto-area dell' Information Retrieval il cui obiettivo è la classificazione automatica di documenti rispetto a un insieme di categorie predefinite, risalgono ai primi anni '60. Tuttavia è nell'ultimo decennio che tale problema ha ricevuto interesse crescente sia nel settore della ricerca scientifica che in contesti applicativi. Infatti, la disponibilità di grandi quantità di dati, resa possibile dallo sviluppo delle moderne tecnologie informatiche e dei servizi web affermatisi di recente, ha posto il problema della loro memorizzazione e organizzazione. Nell'ambito della comunità scientifica, l'approccio dominante è basato sull'applicazione di tecniche di tipo *Machine Learning*, il cui obiettivo è la definizione di sistemi capaci di "apprendere" automaticamente le caratteristiche di una o più categorie, sulla base di un insieme di documenti precedentemente classificati (*training set*). Questo approccio presenta numerosi vantaggi rispetto a quello di tipo *Expert Systems* (in cui esperti di dominio sono impiegati nella definizione manuale dei classificatori per le categorie di interesse). I sistemi di apprendimento, infatti, mostrano generalmente un'elevata efficacia, consentono un considerevole risparmio in termini di risorse umane impiegate nel processo di definizione dei classificatori e garantiscono una immediata portabilità verso nuovi domini.

Negli ultimi anni sono stati proposti numerosi metodi, basati su processi di tipo induttivo, per l'apprendimento automatico di classificatori. Questi sistemi sono generalmente basati su tecniche statistiche e spesso sono stati importati nell'ambito del Text Categorization da altre aree dell'Information Retrieval e del Data Mining, come nel caso delle Support Vector Machine, dapprima utilizzate per problemi di regressione e attualmente considerate allo stato dell'arte per il Text Categorization.
Un posto di rilievo nel paradigma dell'induzione di classificatori è occupato dagli algoritmi di apprendimento rule-based. I classificatori, specificati come insiemi di regole, hanno la proprietà desiderabile di essere comprensibili da un lettore umano, al contrario della maggior parte degli altri approcci esistenti, come Support Vector Machine, Neural Network, che sono di tipo black-box, tali, cioè, che un umano non possa interpretare i classificatori prodotti, né intervenire nel processo di apprendimento.

Nell'ambito del Text Categorization, il problema dell'induzione di regole può essere in generale formulato come segue. Dati:

1. Una conoscenza pregressa (background knowledge) B, rappresentata come un insieme di fatti logici ground del tipo $T \in d$ che indicano la presenza del termine t nel documento d (anche altri fatti possono far parte di B);

2. un insieme di esempi positivi, rappresentati come fatti logici ground del tipo $d \in C$, che individuano l'insieme dei documenti manualmente classificati sotto la categoria c, cioè la classificazione ideale di c (l'insieme degli esempi negativi è definito implicitamente secondo la Closed World Assumption, per cui se un documento d non è esplicitamente definito come esempio positivo per c, allora esso è un esempio negativo.);

costruire un insieme di ipotesi (il classificatore di c) che, insieme alla background knowledge, soddisfi tutti gli esempi (positivi e negativi).

Un problema di questo tipo è computazionalmente complesso, a meno che non si rilassi il vincolo per il quale l'algoritmo di learning deve rappresentare con esattezza il concetto target e si consentano, invece, delle approssimazioni. Il teorema di Valiant della PAC-learnability (Probably Approximately Correct) fornisce un modello di "learning polinomiale" per un sottoinsieme della logica preposizionale. Nel framework PAC, la quantità di risorse polinomialmente limitate (sia in termini di numero di esempi che di tempo computazionale) è controbilanciata dall'accuratezza delle ipotesi indotte.
Le regole indotte a partire dalla background knowledge e dagli esempi (sia positivi che negativi) consentiranno predizioni sull'appartenenza di un documento a una categoria, sulla base della presenza/assenza di un insieme di termini nel dato documento. Comunque, mentre nella teoria computazionale del learning si assume che gli esempi di input siano consistenti con qualche ipotesi nello spazio delle ipotesi, nel Text Categorization ciò non è necessariamente vero. Infatti, in generale, non è possibile classificare un documento sotto una data categoria solo sulla base dei termini che appaiono in esso. L'ipotesi indotta, in tal caso, è una tra quelle che massimamente soddisfa sia gli esempi positivi che quelli negativi.

In questa tesi presentiamo Olex, una nuova tecnica per l'induzione di regole di classificazione di testi. Il problema dell'apprendimento di classificatori in Olex è definito come un problema di ottimizzazione, in cui la F-misura è utilizzata come

funzione obiettivo. In particolare, obiettivo del task di ottimizzazione è quello di determinare un insieme ottimo X_c di termini discriminanti (*d-terms*) capaci di caratterizzare i documenti del training set della categoria *c*.

Un termine discriminante T^s è una congiunzione di termini "semplici" con un segno (positivo o negativo). Diciamo che T^s appare nel documento *d* se tutti i termini di cui T^s è composto appaiono in *d*. Intuitivamente, un termine positivo che appare in un documento *d* è indicativo dell'appartenenza di *d* alla categoria *c*; dualmente, un termine negativo è indicativo di non appartenenza. Quindi, un documento che contenga almeno un d-term positivo e non contiene alcun d-term negativo è *classificabile* sotto *c*, secondo X_c.

Il task di ottimizzazione, quindi, può essere definito informalmente come il problema di trovare un insieme X_c di termini tali che l'insieme dei documenti del training set classificabili sotto *c*, secondo X_c, massimizzi la funzione obiettivo (intuitivamente, aderisca quanto più possibile al training set della categoria *c*).

Dato un insieme (ottimo) di termini X_c, l'ipotesi corrispondente (il classificatore di *c*) ha la forma seguente:

$$\begin{aligned} c &\leftarrow T_1 \in d, T_{n+1} \notin d, \cdots, T_{n+m} \notin d \\ &\ldots\ldots \\ c &\leftarrow T_n \in d, T_{n+1} \notin d, \cdots, T_{n+m} \notin d. \end{aligned}$$

e stabilisce la classificazione del documento d sotto la categoria c, se d contiene almeno uno dei termini positivi $T_1, ..., T_k$ e non contiene alcun termine negativo $T_{k+1}, ..., T_n$. Quindi, la presenza di un d-term positivo richiede la contestuale assenza di tutti d-term negativi. I classificatori indotti contengono una regola per ogni d-term positivo e tutte le regole condividono la stessa parte negativa, costituita da un letterale negativo per ogni termine negativo in X_c.
Notiamo che il linguaggio delle ipotesi di Olex, costituito essenzialmente da clausole di Horn estese da congiunzioni negative di termini, non è PAC-learnable.
Siccome l'insieme che massimizza la funzione obiettivo dipende dalla scelta del vocabolario (cioè l'insieme dei termini selezionati per l'induzione dei classificatori), al fine di trovare i classificatori "migliori" l'algoritmo di ottimizzazione viene ripetuto con diversi vocabolari di input e infine i classificatori con le migliori prestazioni vengono scelti.

Il linguaggio delle ipotesi di Olex è originale e, come dimostrato dalla sperimentazione, molto efficace nel produrre classificatori accurati e compatti. Gli esperimenti effettuati su due *corpora di benchmark* generalmente usati in letteratura al fine di confrontare algoritmi di learning, REUTERS-21578 e OHSUMED , hanno confermato le aspettative sul nostro modello. Infatti, su entrambi i *data set*, Olex ha prestazioni molto elevate, tra le migliori in letteratura; inoltre, a differenza di altri algoritmi di learning che mancano di interpretabilità, Olex ottieneinduce modelli di classificazione che possono essere facilmente letti, compresi e modificati da un essere umano.
Le elevate prestazioni ottenute sui *data set* presi in considerazione mostrano che il paradigma "un letterale positivo, zero o più letterali negativi" è molto efficace. Intuitivamente, possiamo dire che esso consente di catturare gran parte dei documenti corretti (attraverso il letterale positivo) senza tuttavia commetter troppi errori (grazie ai letterali negativi).
A differenza di altri sistemi di learning, Olex è basato su idee molto semplici e dirette e perciò fornisce una chiara intuizione del modello alla base del processo di apprendimento. Inoltre, Olex presenta diverse proprietà desiderabili per l'apprendimento di classificatori:

- è accurato anche per categorie piccole, cioè con un basso numero di documenti manualmente associati a esse;
- non richiede tutto l'insieme di termini del training set per l'apprendimento ma, al contrario, lavora bene anche su piccoli vocabolari;
- è robusto, in quanto mostra un comportamento simile su tutti i *data set* considerati.

Inoltre, grazie al fatto di essere rule-based, Olex consente una semplice integrazione della conoscenza di dominio, racchiusa in thesauri, nel processo di apprendimento. L'utilità di tale conoscenza nel processo di learning è stata sperimentata in Olex su due *data set*, relativi al settore assicurativo e fornitici da una società americana, la FCSI (*Full Capture Solutions, Inc*). Questa prima sperimentazione ha mostrato che l'utilizzo di conoscenza di dominio dà solo un piccolo contributo al miglioramento delle prestazioni dei classificatori prodotti. Tuttavia questo risultato deve ritenersi parziale; uteriori test saranno effettuati per stabilire se questo risultato può essere generalizzato oppure l'utilizzo di tesauri più appropriati possa effettivamente apportare un importante contributo nella classificazione documentale.

Infine, il sistema sviluppato supporta l'integrazione dell'approccio manuale nell' apprendimento automatico di classificatori. Grazie all'interpretabilità dei classificatori prodotti, infatti, l'ingegnere della conoscenza può partecipare alla costruzione di un classificatore, specificando un insieme di regole da utilizzare congiuntamente a quelle apprese automaticamente. Più in dettaglio, al fine di supportare un approccio ibrido, il sitema Olex è stato progettato in maniera tale che i classificatori prodotti automaticamente siano modificabili manualmente. Un'ulteriore funzionalità introdotta al fine di sfruttare la conoscenza di dominio è quella che prevede il completamento automatico di un classificatore scritto manualmente. Questa funzionalità consente di:

- scrivere un insieme di regole di classificazione, sulla base delle indicazioni dell' esperto del dominio, e verificarne l'accuratezza
- chiedere al sistema di completare automaticamente il classificatore manuale al fine di migliorarne l'accuratezza.

I risultati sperimentali hanno mostrato che questa cooperazione può avere effettivi sinergici, consentendo di ottenere prestazioni migliori sia rispetto all'approccio manuale che a quello automatico.

In sintesi, in questa tesi vengono affrontatele questioni su riportate e in particolare:

- viene definito formalmente il problema del Text Categorization e vengono rivisitati i principali contesti applicativi nei quali sono sfruttate tecniche di questo tipo;
- vengono discussi i metodi e i sistemi di classificazione documentale, al fine di realizzare una valutazione comparativa delle loro peculiarità nell'ambito della tematica di interesse;
- viene presentato il sistema Olex; in particolare, dopo aver definito il problema di selezione dei termini discriminanti, che rappresenta il cuore del nostro metodo, viene dimostrato che tale problema è computazionalmente difficile e viene poposta un' euristica per la sua soluzione.
- vengono mostrati i risultati sperimentali ottenuti e viene effettuata una valutazione comparativa delle prestazioni del nostro sistema rispetto ad altri sitemi di learning esistenti in letteratura.

Contents

Abstract

Text Categorization is the problem of the automatic categorization (or classification) of texts into pre-specified categories. It dates back to the early 60s, but only in the last ten years it has witnessed a booming interest, both in research area and in applicative contexts. In fact, as the modern information technologies and the web-based services successfully make a great volume of information available, the problem of accessing, selecting and managing this information, usually expressed as textual data, arises. In the research community the dominant approach to this problem is based on the application of 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.

In the last years, a great number of statistical classification and machine learning methods to automatically construct classifiers using labelled training data have been proposed. The common target to all these systems is the definition of *category profiles*, describing the characteristics that a document must have in order to be associated to one o more categories. The differences among them rely on the techniques used to this task, which vary from decision tree to genetic algorithms, from probabilistic techniques to mathematic and geometrical methods. Among them, rule learning algorithms has become a successful strategy for classifier induction. While weighted solutions such as the linear probabilistic methods or nearest-neighbor methods may also prove reasonable, the models they employ are not explicitly interpretable; rule-based classifiers, instead, provide the desirable property of being readable, easy for people to understand. Several approaches (either rule-based or not) exploiting negative information for text classification can

be found in the literature.

A general formulation of the induction problem (for text categorization) is as follows. Given

- a *background knowledge* B as a set of ground logical facts of the form $t \in d$, meaning that term t appears in document d (other ground predicates may occur in B as well)
- a set of *positive examples* expressed as ground logical facts of the form $d \in c$, meaning that document d belongs t o category c (*ideal classification*); *negative examples* are implicitly stated according to the Closed World Assumption (i.e., if $d \in c$ is not a positive example, then it is a negative one)

constructs a hypothetical rule set (the classifier of c) that, combined with the background knowledge B, agrees all (both positive and negative) examples. It is well known that this problem is computationally difficult, unless the requirement that a learning algorithm identifies the target concept *exactly* is relaxed to allow *approximations*. The Valiant's theory of *PAC-learnability* (Probably Approximately Correct) provides a model of "polynomial learning" for a subset of propositional logic. In the PAC framework, the polynomially bounded amount of resources (both number of examples and computational time) is traded-off against the accuracy of the induced hypotheses. The induced rules from both background knowledge and examples will allow prediction about the belonging of a document to a category on the basis of the presence or absence of some given terms in that document. However, while in computational learning theory it is assumed that the input sample is consistent with some hypothesis in the hypothesis space, in text classification this is not necessarily true; indeed, it is not possible, in general, to correctly categorize a document under a category only on the basis of the terms occurring in it. Thus, the expected induced hypothesis in such a case is one which *maximally* satisfies (both positive and negative) examples.

In this thesis we propose Olex, a novel method for the automatic induction of rule-based text classifiers. In Olex, the learning problem is stated as an optimization problem relying on the F-measure as the objective function. In particular, the optimization task is that of determining a best set $X_c = \{T_1^+, \cdots, T_n^+, T_{n+1}^-, \cdots T_{n+m}^-\}$ of *discriminating* terms (d-terms) for c,capable of "best" characterizing the documents in the training set TS_c of c. A d-term is of the form T^s,

where T is a conjunction of (simple) terms with a sign (either positive or negative). We call T *conjunctive* term (co-term). A d-term T^s occurs in a document d if T occurs in d, i.e., if all the simple terms, of with T is made up, occur in d. A positive d-term occurring in a document d is indicative of membership of d in c, while a negative one is indicative of non-membership. Thus, a document d containing *any* positive d-term in X_c and *none* of the negative d-terms in X_c, is *eligible* for classification under c according to X_c. Hence, the aim of the optimization task is that of finding a set X_c of d-terms such that, by classifying under c the set of documents of the training set eligible for classification according to X_c, the resulting F-measure is maximum (intuitively, this corresponds to finding X_c such that the set of eligible documents best "fits" the training set of c). Not surprisingly, the above task is computationally untractable.

Now, given a (best) set $X_c = \{T_1^+, \cdots, T_n^+, T_{n+1}^- \cdots T_{n+m}^-\}$ of d-terms, the corresponding hypothesis (the classifier of c) is of the form

$$c \leftarrow T_1 \in d, T_{n+1} \notin d, \cdots, T_{n+m} \notin d$$

$$\cdots$$

$$c \leftarrow T_n \in d, T_{n+1} \notin d, \cdots, T_{n+m} \notin d$$

and states the condition "if any of the co-terms $T_1, \cdots, T_n$ occurs in d and *none* of the co-terms $T_{n+1}, \cdots, T_{n+m}$ occurs in d then classify d under category c". That is, the occurrence of a co-term T_i, $1 \leq i \leq n$, in a document d requires the contextual *absence* of the (possibly empty) set of co-terms $T_{n+1}, \cdots, T_{n+m}$ in order for d be classified under c. Notice that there is one rule for each positive d-term in X_c and, for each rule, one negative literal for each negative d-term in X_c (thus, all rules share the same negative part $T_{n+1} \notin d, \cdots, T_{n+m} \notin d$).
We remark that the Olex's hypothesis language, essentially Horn clauses extended by negative conjunctions of terms, is not PAC-learnable.
Since the set X_c which maximizes the objective function depends on the choice of the vocabulary (i.e., the set of terms selected for rule induction), to pick the "best" classifier Olex proceeds by repeatedly running the optimization algorithm with different input vocabularies, and eventually selecting the classifier with the best performance.

Olex's hypothesis language is original and, as shown by the experimental results, very effective in producing accurate and compact classifiers. Experiments,

carried out on two standard benchmark data collections, namely, REUTERS-21578 and OHSUMED, confirm the expectations on our model. In fact,Olex achieves very good performance on both data collections, among the best reported in the literature. In particular, Olex showed to outperform traditional classifiers such as k-NN, Naive Bayesian, C4.5, Ripper, etc., and to be competitive with SVM. Further, unlike SVM, that lacks interpretability, Olex yields classification models that can be easily read, understood and modified by humans. The induced classifiers are indeed very compact: on the top ten categories of the REUTERS-21578, the number of rules in a classifier ranges between 2 and 34.
High performance and compactness of classifiers are consequence of highly effective rules; intuitively, the paradigm "one positive literal, more negative literals" allows rules to catch most of the right documents (through the positive literal), while not making "too many" mistakes (thanks to the negative ones).
Unlike other rule learning systems, Olex is based on very simple and straightforward ideas and, thus, provides a clear intuition of what learning is about. Further, it is formally well-defined and understood.
Further, Olex enjoys a number of further desirable properties:

- it is accurate even for relatively small categories (i.e., it is not biased towards majority classes);
- it can learn from small vocabularies;
- it is robust, i.e., shows a similar behavior on both data sets we have experimented.

Further, thanks to its rule-based approach, the implemented prototype allows an immediate and sound integration of background knowledge. The usefulness of domain-specific knowledge has been evaluated on two data sets belonging to an American insurance agency, by performing a *Semantic Analysis* task, whereby documents have been represented in terms of the extracted concepts. This first empirical evaluation showed that knowledge-based feature generation does not substantially contribute to improve learning of text classification rules. This is a partial result; further investigation have to be carried out, in order to state whether this result can be generalized for our learning approach or some contribution is obtained when using more appropriate thesauri.

Lastly, the system supports the integration of a manual approach into the automatic categorization. Thanks to the interpretability of the produced classifiers, indeed, the Knowledge Engineer can participate in the construction of a classifier,

by manually specifying a set of rules to be used in conjunction with those automatically learned. Experimental results showed that this cooperation may bring Text Categorization to an higher performance level.

In this thesis, after having described Text Categorization problem and discussed some interesting related works, we introduce our learning approach. More specifically, this thesis is organized as follows:

- In Part I, we formally define Text Categorization and its various subcases and review the most important tasks to which Text Categorization has been applied; eventually, we discuss the performance measures classically used to evaluate the efficacy of a classifier and describe the benchmark corpora used to test our system.
- In Part II, we give a survey of the state-of-the-art in Text Categorization, describing some of the algorithms that have been proposed and evaluated in the past.
- In Part III, after providing an overview of Olex and giving some preliminary definitions and notation, we state the optimization problem of selecting a best set of discriminating terms (which is the heart of our method) and prove that this task is computationally difficult. Thus, we propose a heuristic approach to solve it and give a description of the whole learning process. Then, we present the experimental results and provide a performance comparison with other learning approaches. Finally, in the light of the obtained results, we provide a discussion (Section 10.1).

Part I

Text Classification

In this part we present the Text Categorization problem (TC).

Text classification (TC) 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 TC, 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 Categorization

Text categorization (TC - also known as *Text Classification* or *Document Classification*) represents the activity of labelling natural language texts with thematic categories from a predefined set. TC has a long history, dating back to the early 60s, but it was not until the early 90s that it became a major subfield of the information systems discipline, largely due to increased applicative interest and to the availability of more powerful hardware. Nowadays TC is used in many applicative contexts, ranging from automatic document indexing based on a controlled vocabulary, to document filtering, automated metadata generation, word sense disambiguation, population of hierarchical catalogues of Web resources, and in general any application requiring document organization or selective and adaptive document dispatching.
In this chapter, we formally define the Text Categorization problem and review the most important tasks to which TC has been applied. In section 1.3 we discuss three types of approach to Text Categorization problem, here summarized:

Expert Systems approach, based on the manual definition of classifiers, has been proposed in the '60. Experimental results showed that this technique achieves very high performances but is a very costly activity;

Machine Learning approach, aiming at the construction not of a classifier, but of an automatic builder of classifiers (the learner), appeared in Text Categorization are since the early '90s and eventually become the dominant one;

Hybrid approach, which exploits the cooperation between the above described approaches for the development of a categorization workbench combining the benefits of domain specific rules with the generality of automatically

learned ones. This approach is of interest for us since our approach, substantially relying on a Machine Learning technique, allows the introduction of domain expert knowledge in the classifier construction. An example of combination of manual and automatic approaches in Olex is provided in chapter 9.

Finally, we explore the use of external knowledge in Text Categorization, aiming at finding an improvement of performance results by using formally represented background knowledge in the form of thesauri. More precisely, the aim is to extend the classical document representation, based on the extraction of terms trough simple linguistic techniques, by means of external vocabularies which should help to "capture" the meaning of words.

1.1 Problem Definition

Text Categorization may be may be seen as the task of determining an assignment of a boolean value to each pair $\langle d_j, c_i \rangle \in D \times C$ 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 T for a_{ij} is interpreted as a decision to file d_j under c_i, while a value of F is interpreted as a decision not to file d_j under c_i. More formally, TC represents the task of approximation of the unknown function $\tilde{\Phi} : D \times \mathcal{C} \rightarrow \{T, F\}$ (that describes how documents ought to be classified) by means of a function $\Phi : D \times \mathcal{C} \rightarrow \{T, F\}$, called the classifier (aka rule, or hypothesis, or model) such that Φ and $\tilde{\Phi}$ coincide as much as possible. In chapter 2, we will show how to precisely define and measure this degree of coincidence.
Basic assumptions are that no additional knowledge about categories is provided (they are just symbolic labels), neither exogenous information about documents (metadata such as e.g. publication date, document type, publication source) is available to help the process of building the classifier. The effect of these assumptions is that the algorithms that we will discuss are completely general and do not depend on the availability of special-purpose resources that might be costly to develop or might simply be unavailable.
Different constraints may be enforced on the categorization task; depending on the application, we may want that:

1. $\{\leq 1 \mid 1 \mid \geq 1 \mid ...\}$ elements of C must be assigned to each element of D. When exactly one category is assigned to each document this is often referred to as the single-label categorization case.

2. each element of C must be assigned to $\{\leq 1 \mid 1 \mid \geq 1 \mid ...\}$ elements of D.

A special case of single-label categorization (or "non-overlapping categories" case) is binary categorization, in which each document d_j must be assigned either to category c_i or to its complement c_i. From a theoretical point of view, the binary case (hence, the single-label case too) is more general than the multi-label case, in the sense that an algorithm for binary classification can also be used for multi-label classification: one needs only transform a problem of multi-label classification under categories $\{c_1, ..., c_m\}$ into m independent problems of binary classification under categories $\{c_i, \bar{c_i}\}$, for $i = 1, ..., m$.

The techniques we will consider here are applicable irrespectively of whether any of above-mentioned constraints are enforced or not and, in the rest of the chapter, unless explicitly specified, we will be dealing with the binary case.

1.2 Application of Text Categorization

Since its first application in 1961, in Marons seminal work on probabilistic text classification, TC has been applied in a number of different contexts. In this section, we briefly review the most important applications, in which it has been used. The borders between the different classes of applications listed here are fuzzy and somehow artificial, and some of these may be considered special cases of others. Other applications we do not explicitly discuss are speech categorization by means of a combination of speech recognition and TC [64] [71], multimedia document categorization through the analysis of textual captions [69], author identification for literary texts of unknown or disputed authorship [32], language identification for texts of unknown language [16], automated identification of text genre [46] and automated essay grading [51].

1.2.1 Automatic Indexing for Boolean IR Systems

The first applications of TC were in the field of automatic document indexing for IR systems relying on a controlled dictionary. Among them, the most prominent is that of Boolean Systems, whose target is the assignment of a set of key words and key phrases to each available document, in order to describe their content. Key words and phrases belong to a finite set called controlled dictionary, often consisting of a thematic hierarchical thesaurus (e.g. the NASA thesaurus for the aerospace discipline, or the MESH thesaurus for medicine). Usually, this is a costly activity because the selection of representative words and expressions is

done by trained human indexers. If the entries in the controlled vocabulary are viewed as categories, text indexing can be considered an instance of document-pivoted TC [73], where new documents may be classified as they become available. Various text classifiers explicitly conceived for document indexing have been described in the literature; see, for example, [34], [68], [77].
Another application, closely related to automatic indexing, is that of *automated metadata generation*, which represent a fundamental tool in building and maintaining digital libraries, where documents are tagged by metadata that describe them under a variety of aspects (e.g., creation date, document type or format, availability, etc.). Some of these metadata is *thematic*, that is, its role is to describe the semantics of the document by means of bibliographic codes, key words or key phrases. The generation of metadata may thus be viewed as a problem of document indexing with controlled dictionary, and thus tackled by means of TC techniques.

1.2.2 Document Organization

Among the applications that may be addressed to TC techniques, there are many issues pertaining to document organization and filing, be it for purposes of personal organization or structuring of a corporate document base. As an instance, we can consider the classification task to which the news are subjected, prior to their publication, in order to be filed under the categories of the scheme adopted by the newspaper; typical categories might be Personals, Cars for Sale, Real Estate, etc. While most newspapers would handle this application manually, those dealing with a high volume of classified ads might prefer an automatic system to choose the most suitable category for a given ad. In this case a typical constraint is that exactly one category is assigned to each document. Similar applications are the organization of patents into categories for making their search easier, the automatic filing of newspaper articles under the appropriate sections (e.g. Politics, Home News, Lifestyles, etc.), or the automatic grouping of conference papers into sessions.

1.2.3 Text Filtering

Text filtering (also known as *document routing*) is the activity of classifying a dynamic collection of texts, i.e. a stream of incoming documents dispatched in an asynchronous way by an information producer to an information consumer [11]. A very useful document routing system is an e-mail filter, whose role is to reject

"junk" mail, keeping only those ones that are relevant to the user. Mail filtering can be seen as a case of single-label categorization, i.e. the classification of incoming documents in two disjoint categories, *relevant* and *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, an e-mail filter might be trained to discard "junk" mail [6] [27] and further classify non-junk mail into topical categories of interest to the user [21].

1.2.4 Word sense disambiguation

Word sense disambiguation (WSD) refers to the activity of finding, given the occurrence in a text of an ambiguous (i.e. polysemous or homonymous) word, the sense this particular word occurrence has. For instance, the English word bank may have (at least) two different senses, as in the Bank of England (a financial institution) or the bank of river Thames (a hydraulic engineering artifact). It is thus a WSD task to decide to which of the above senses the occurrence of bank in "Last week I borrowed some money from the bank" refers to. WSD is very important for a number of applications, including natural language understanding, or indexing documents by word senses rather than by words for IR purposes. WSD may be seen as a categorization task (see e.g. [36] [39]) once we view word occurrence contexts as documents and word senses as categories.

1.2.5 Hierarchical categorization of Web pages

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. 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. Unlike in the previous applications, this is a case in which one might typically want each category to be populated by a set of $k1 \leq x \leq k2$ documents, and one in which category-centered categorization may be aptest.

1.3 Approaches to Text Categorization

1.3.1 Expert Systems Approach

Since the first applications of Text Categorization, during the early '60s and until the '80s, the main approach used to the construction of automatic document categorizers involved *knowledge-engineering* techniques: domain experts used to process and analyze documents to manually build 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** $< category >$

which has the effect of classifying the document under $< category >$, if it satisfies the *disjunctive normal form* $< DNF\ \ Booleanformula >$.
A well known example of an expert system for this task is the CONSTRUE system [38] built by Carnegie Group and used by the Reuters news agency. The drawback of this "manual" approach to the construction of automatic classifiers is the existence of a knowledge acquisition bottleneck. 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. [38] report a .90 "break-even" result (that we will discuss, together with other effectiveness measures for TC, in chapter 2) on a subset of the REUTERS-21578 . While these are exceptionally good results, the test set seems to have been relatively sparse when compared to the number of possible topics.

1.3.2 Machine Learning Approach

Since the early '90s, the *Machine Learning* approach to the construction of text classifiers has gained popularity and eventually become the dominant one, at least in the research community (see [63] for a comprehensive introduction to ML). In this approach a general inductive process (also called the *learner*) 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 or $\overline{c_i}$ by a domain expert; from these characteristics, the inductive process gleans the characteristics that a novel document should have in order to be classified under c_i .
In ML terminology, the classification problem is an activity of *supervised* learning, since the learning process is driven, or "supervised", by the knowledge of the categories and of the training instances that belong to them. 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 a learner is (as it often is) available off-the-shelf, all that is needed is the inductive, *automatic* construction of a classifier from a set of manually classified documents. The same happens if a classifier already exists and the original set of categories is updated, or if the classifier is ported to a completely different domain. In the ML approach the manually classified documents are then the key resource. The most favorable case is the one in which they are already available; this is the typical case of an organization that had already been carrying out the same categorization activity manually and decides to automate the process. The less favorable case is when no manually classified documents are available; this is typically the case of an organization that starts a categorization activity and decides to opt for an automated modality straightaway. In this case, the ML approach is still more convenient than the KE approach. In fact, it is easier to manually classify a set of documents than to build and tune a set of rules, for the simple reason that it is usually easier to characterize a concept extensionally (i.e. to indicate instances of it) than intensionally (i.e. to describe the concept in words, or to describe a procedure for recognizing its instances). Classifiers built by means of ML techniques nowadays achieve impressive levels of effectiveness (see chapter 2), making automatic classification a qualitatively (and not only economically) viable alternative to manual classification.

Training set, test set and validation set

The ML approach relies on the existence of an initial corpus $\Omega = \{d_1, ..., d_\Omega\}$, $(\Omega \subset D)$ of documents previously classified under the same set of categories $C = \{c_1, ..., c_m\}$, with which the system will need to operate. This means that the values of the total function $\tilde{\Phi} : D \times \mathcal{C} \rightarrow \{T, F\}$ are known for every pair $\langle d_j, c_i \rangle \in \Omega \times C$. For a given a category c_i, a document d_j is said

- *positive example* if $\tilde{\Phi}(d_j, c_i) = T$

- *negative example* if $\tilde{\Phi}(d_j, c_i) = F$

In research settings (and in most operational settings too), once a classifier has been built it is desirable to evaluate its effectiveness. In this case, prior to classifier construction the initial corpus is usually split in two sets, not necessarily of equal size:

- a ***training(-and-validation)*** set $TV = \{d_1, .., d_{|TV|}\}$. This is the set of documents observing the characteristics of which the classifiers for the various categories are inductively built;
- a ***test set*** $Te = \{d_{|TV|+1}, .., d_\Omega\}$. This set will be used for the purpose of testing the effectiveness of the classifiers. Each document in Te will be fed to the classifiers, and the classifier decisions $\Phi(d_j, c_i)$ compared with the expert decisions $\tilde{\Phi}(d_j, c_i)$; a measure of classification effectiveness will be based on how often the $\Phi(d_j, c_i)$ values match the $\tilde{\Phi}(d_j, c_i)$ values.

Note that, in order to carry out a scientific realistic evaluation of a learning algorithm, the documents in Te cannot participate in the inductive construction of the classifiers, since if this condition were not satisfied the experimental results obtained would probably be unrealistically good [63]. In an operational setting, after evaluation has been performed one would typically re-train the classifier on the entire initial corpus, in order to boost effectiveness. This means that the results of the previous evaluation would be a conservative estimation of the real performance, since the final classifier has been trained on more data than the evaluated classifier. This approach is called the train-and-test approach.
An alternative approach is the k-fold cross-validation approach (see [63]), whereby k different classifiers $\Phi_1, ..., \Phi_k$ are induced by partitioning the initial corpus into k disjoint sets $Te_1, ..., Te_k$, and then iteratively applying the train-and-test approach on pairs $\langle TV_i = \Omega \backslash Te_i, Te_i \rangle$. The final effectiveness figure is obtained by individually computing the effectiveness of the resulting k classifiers $\Phi_1, ..., \Phi_k$, different among each other because they have been generated from k different training-and-validation sets, and then averaging the individual results in some way.
In both the train-and-test and k-fold cross-validation approaches, it is often the case that in order to optimize the classifier its internal parameters should be tuned by testing which values of the parameters yield the best effectiveness. In order to make this optimization possible, in the train-and-test approach the set $\{d_1, .., d_{|TV|}\}$ is further split into a training set $Tr = \{d_1, .., d_{|Tr|}\}$, from which the classifier is inductively built, and a validation set $Va = \{d_{|Tr|+1}, .., d_{|TV|}\}$ (sometimes called

a *hold-out set*), on which the repeated tests of the classifier aimed at parameter optimization are performed; the obvious variant may be used in the k-fold cross-validation case. Note that, basically for the same reason why we do not test a classifier on the documents it has been trained on, we do not test it on the documents it has been optimized on; that is, test set and validation set must be kept separate.

1.3.3 Hybrid Approach

The Machine Learning and Expert Systems Approaches, described in the above sections, have sometimes been combined for the development of categorization workbench combining the benefits of domain specific rules with the generality of automatically learned ones. This cooperation, in fact, may be very effective, since both approaches have some limits, that can be overcome if used in synergy. As noticed by [75], in real world applications, users of automatic categorization are confronted with two problems:

1. Getting the needed quantity of training samples for a taxonomy can be a laborious task, especially for category topics chosen which are semantically close to each other.

2. Though using automatic categorization, some customers wish to keep control of the assignment of certain documents. Instead, text categorization methods are determined by categorization model generated on the basis of training samples and the customer can only let the model be modified by altering the training data.

These problems awake the need for an integration of manual categorization rules into the overall categorization process with which the sample complexity should be reduced and the user should be enabled to influence the categorization result more directly and effectively. A trivial way to allow the intervention of the knowledge engineer into the classifier definition problem is to let him build some categorization rules and then adding them to the automatic learned ones. A more interesting way to exploit domain knowledge is to use the domain knowledge in the automatic induction of a classifier. As shown in chapter 9, the Olex system supports the integration of a manual approach into the automatic categorization. Thanks to the interpretability of the produced classifiers, indeed, the Knowledge Engineer can participate in the construction of a classifier, by manually specifying a set of rules to be used in conjunction with those automatically learned.

Unlike automatic categorizers, the manual categorization performed by knowledge engineers is based on the semantic of words. Usually, humans associate each category with its characteristics which can be symbolized by and embodied in words. So categories can be discriminated by domain-specific lexicon. Compared with automatic categorizer, manually defined classifiers have lower precision, but often achieve higher recall, since a domain expert has over those of the training set more extensive domain-specific vocabulary.
Conversely, machine learning methods are not able to choose features according to their semantic relevance like humans do. A study on automatic feature selection shows that in order to achieve a precision of more than 90% with decision tree method C4.5 either at least ca. 200 training sample are needed, or applied algorithm is able to determine an appropriate subset with few features [50]. In their study, [65] show that benefiting from the incorporation of user's domain knowledge, the categorization workbench can improve the recall by a factor of two till four with the same number of training samples as the automatic categorizer uses, Further, to get a comparable categorization quality, the categorization workbench just needs an eighth till a quarter of the training samples as the automatic categorizer does.

1.4 Use of external knowledge in Text Categorization

Recently proposed works aim at finding an improvement of text classification results by using formally represented background knowledge in the form of thesauri to extend the classical *bag-of-words* feature representation paradigm. The latter, together with the *multi-words expression* one, often shows to be sufficient for accurate learning, since individual words and their combination carry and important part of the meaning of the text. However, this doesn't always hold, due to the *polysemy* and *synonymy* of words. In fact, synonymous words are mapped into different features while polysemous ones are treated as one single feature (but the may actually have multiple distinct meanings). Further, there is a *lack of generalization* (as an instance, there is no way to generalize similar terms like "beef" and "pork" to their common *hypernym* "meat".
Thus, thesauri have sometimes been introduced in Text Categorization approaches to exploit semantic relations among terms. Formally speaking, a thesaurus is made up of three components, described below.

Definition 1.1 (**Core Component**) It is a structure $T := (C; <_C)$ consisting of a set C, whose elements are called *concept identifiers*, and a partial order $<_C$ on C, called concept hierarchy or taxonomy.

Definition 1.2 (**Relation between Concepts**) If $c_1 <_C c_2$ for any $c_1, c_2 \in C$, then c_1 is a subconcept (specialization) of c_2 and c_2 is a superconcept (generalization) of c_1. If $c_1 <_C c_2$ and there exists no $c_3 \in C$ with $c_1 <_C c_3 <_C c_2$, then c_1 is a direct subconcept of c_2, and c_2 is a direct superconcept of c_1, denoted by $c_1 \prec c_2$.

Definition 1.3 (**Lexicon**) A lexicon for a thesaurus T is a tuple $Lex := (S_C; Ref_C)$ consisting of a set S_C, whose elements are called signs for concepts (symbols), and a relation $Ref_C \subseteq S_C \times C$ called lexical reference for concepts, where $(c, c) \in Ref_C$ holds for all $c \in C \cap S_C$. Based on Ref_C, for $s \in S_C$ we define $Ref_C(s) := \{c \in C | (s, c) \in Ref_C\}$. Analogously, for $c \in C$ it is $Ref_C^{-1}(c) := \{s \in S_C | (s, c) \in Ref_C\}$.

Examples of thesauri used for Text Categorization tasks are WordNet [14] and Mesh [2]. WordNet is a lexical database which organizes simple words and multi-word expressions of different syntactic categories into so called synonym sets (*synsets*), each of which represents an underlying concept and links these through semantic relation. The MeSH Thesaurus is has more complex structure. It is an ontology that has been compiled out of the Medical Subject Headings (MeSH) controlled vocabulary thesaurus of the United States National Library of Medicine (NLM). The ontology contains more than 22000 concepts, each enriched with synonymous and quasi-synonymous language expressions.
Different strategies have been explored in the literature in order to use domain specific knowledge in the automatic induction of category classifiers; some rule-based approaches exploiting thesaurus knowledge based are provided in section 5.3.

Chapter 2

Categorization Effectiveness Evaluation

The evaluation of a text classifier is typically conducted experimentally. The reason to select the experimental way rather than the analytical one 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, due to its subjective character, inherently non-formalizable. 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.

2.1 Precision and Recall Measures

While a number of different effectiveness measures have been used in evaluating text categorization in the past, almost all have been based on the same model of decision making by the categorization system.

Generally, classification effectiveness with respect to a category c_i is measured in term of the classic IR notions of *precision* (*P*) and *recall* (*R*), adapted to the case of text categorization [74]. Intuitively, P indicates the probability that if a random document d_x is classified under c_i, the decision is correct; while R indicates the probability that, if a random document d_x should be associated to the category c_i, then the right decision is taken. More specifically, given a category c_i, the precision P with respect to c_i is defined as the conditional probability $P(ca_{ix} = T|a_{ix} = T)$ and, analogously the recall R is defined as the conditional probability $P(a_{ix} = T|ca_{ix} = T)$. As they are defined here, P and R are to be understoodas 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. These probabilities may be estimated in terms of the contingency table for category ci on a given test set (see Table 2.1).

Category c_i		expert judgment	
		YES	**NO**
classifier judgment	**YES**	TP_i	FP_i
	NO	FN_i	TN_i

Table 2.1: Contingency table for category c_i.

Here, FP_i (false positives wrt c_i, also known as errors of commission) is the number of documents of the test set that have been incorrectly classified under ci; TN_i (true negatives wrt c_i), TP_i (true positives wrt c_i) and FN_i (false negatives wrt c_i, also known as errors of omission) are defined accordingly. Precision wrt ci and recall wrt ci may thus be estimated as

$$P = \frac{TP_i}{TP_i + FP_i}; \tag{2.1}$$

$$R = \frac{TP_i}{TP_i + FN_i}. \tag{2.2}$$

In multi-label TC, when effectiveness is computed for a set of categories the precision and recall results for individual categories may be averaged in two different ways: here, one may opt for

- ***microaveraging***, rewards classifiers that behave well on heavily populated ("frequent") categories, which count proportionally to the number of their positive training examples:

$$\mu\mathbf{P} = \frac{\sum_{i=1}^{|C|} TP_i}{\sum_{i=1}^{|C|}(TP_i + FP_i)}; \tag{2.3}$$

$$\mu\mathbf{R} = \frac{\sum_{i=1}^{|C|} TP_i}{\sum_{i=1}^{|C|}(TP_i + FN_i)}. \tag{2.4}$$

- ***macroaveraging***, emphasizes classifiers that perform well also on infrequent categories, since "all categories count the same". To compute macro averages, precision and recall are first evaluated locally for each category, and then "globally" by averaging over the results of the different categories:

$$M\mathbf{P} = \frac{\sum_{i=1}^{|C|} P_i}{|C|}; \tag{2.5}$$

$$M\mathbf{R} = \frac{\sum_{i=1}^{|C|} R_i}{|C|}. \tag{2.6}$$

Note that these two methods may give quite different results, especially when 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" [84] and thus favour macroaveraging, while others believe that topics should indeed count proportionally to their frequence, and thus lean towards microaveraging. As we will see in chapters 8 and 9, our system achieves a desirable balance between microaverage and macroaverage values, which are very close to each other, on all the test cases considered in our experimentations.

2.2 Combining Precision and Recall

Some of the performance measures may be misleading when examined alone. For example, a trivial algorithm that says YES to every category for any document will have a perfect recall of 100%, but an unacceptably low score in precision. Conversely, if a system rejects every document for every category, it will have a perfect score in precision, but will sacrifice recall to the extreme. Usually, a classifier exhibits a trade-off between recall and precision when the internal parameters or decision threshold in the classifier are adjusted; to obtain a high recall usually means sacrificing precision and vice-versa. If the recall and precision of a classifier can be tuned to have an equal value, then this value is called the *break-even point* (*BEP*) of the system [57]. BEP has been commonly used in text categorization evaluations. If the recall and precision values cannot be made exactly equal, the average of the nearest recall and precision values is used as the interpolated BEP [9, 53]. A problem with the interpolation is that when the nearest recall and precision values are far apart, the BEP may not reflect the true behavior of the system. The most popular way to combine the two is the function F_α function [54], for some $0 \leq \alpha \leq 1$, i.e.:

$$F_\alpha = \frac{1}{\alpha\frac{1}{P} + (1-\alpha)\frac{1}{R}} \tag{2.7}$$

In this formula α may be seen as the relative degree of importance attributed to P and R: if $\alpha = 1$, then F_α coincides with P, if $\alpha = 0$ then F_α coincides with R. Usually, a value of $\alpha = 0.5$ is used, which attributes equal importance to P and R. As shown in [89], for a given classifier Ψ, its breakeven value is always less or equal than its 1 value.

2.3 Other Effectiveness Measures

Other effectiveness measures different from the ones discussed here have occasionally been used in the literature. Together with precision and recall, accuracy and error have been often used to evaluate category classifier performance values. With respect to Table 2.1 for category c_i, they are defined as:

$$accuracy = \frac{TP_i + TN_i}{N} \tag{2.8}$$

$$error = \frac{FP_i + FN_i}{N} \tag{2.9}$$

where N indicates the total number of documents in the test set.
Although accuracy and error are common performance measures in the machine learning literature and have been used in some evaluations of text categorizations systems, there is a potential pitfall in using them to train or evaluate a binary classifier. When the number of categories is large and the average number of categories per document is small, the accuracy or error may not be a sensible measure of the effectiveness or usefulness of a classifier in text categorization.
Fundamentally, these difficulties in using accuracy and error as performance measures raised from their definitions. Unlike recall and precision, accuracy and error have N, the number of test documents, in their divisor. Therefore, a small change in Table 2.1 of the value TP_i or TN_i will produce only a small change in the value of accuracy(likewise a small change in FP_i or FN_i will produce only a small change in the value of error). However, for rare categories the maximum value of TP_i or FN_i is small. Consequently, TP_i and FN_i may range from zero to their maximum value without having much effect on the value of accuracy or error, respectively. Now consider the value of recall, defined as $TP_i/(TP_i+FN_i)$; the potential values of TP_i and FN_i are both small, and furthermore the quantity $TP_i + FN_i$ is always constant and equal to the number of documents that belong to the category in question. Consequently, any change in the value of TP_i will produce a relatively large change in the value recall. So, recall and precision measures are often preferred in classifiers evaluation, as they are more sensitive with respect to rare categories than accuracy or error.

Chapter 3

Benchmark data sets

Text Categorization algorithms are usually tested on public available standard benchmarks test collections. The existence of such corpora is beneficial to research on this task, since they allow different researchers to experimentally compare their own systems by comparing the results they have obtained on this benchmark. In the following sections, we analyze the two most used benchmark data sets in Text Categorization, the REUTERS-21578 and the OHSUMED corpora, and discuss the problem of the existence of different sub-collections. In fact, while using the same data sets, different researchers have "carved" different sub-collections out of the collections, and tested their systems on one of these sub-collections only.
As we will see in chapter 8, we used these two benchmark corpora in order to evaluate our learning approach and to compare it with other existing methods.

3.1 The REUTERS-21578 collection

The REUTERS-21578 test collection, together with its earlier variants, has been such a standard benchmark for the text categorization task throughout the last ten years. REUTERS-21578 is a set of 21,578 news stories appeared in the Reuters newswire in 1987, which are classified according to 135 thematic categories, mostly concerning business and economy. This collection has several characteristics that make it interesting for Text Categorization experimentation:

- similarly to many other applicative contexts, it is multi-label, i.e. each document 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, 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 Text Categorization systems based on machine learning techniques, since several categories have (under any possible split between training and test documents) very few 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 Text Categorization research in the past years.
The data contained in the "REUTERS-21578 , Distribution 1.0 corpus consist of news stories appeared on the Reuters newswire in 1987. The data was originally labelled by Carnegie Group, Inc. and Reuters, Ltd. in the course of developing the CONSTRUE text categorization system [38], and was subsequently collected and formatted by David Lewis with the help of several other people. A previous version of the collection, known as REUTERS-22173 , was used in a number of published studies up until 1996, when a revision of the collection resulted in the correction of several other errors and in the removal of 595 duplicates from the original set of 22173 documents, thus leaving the 21578 documents that now make REUTERS-21578 . The REUTERS-21578 documents actually used in Text Categorization experiments are only 12902, since the creators of the collection found ample evidence that the other 8676 documents had not been considered for labelling by the people who manually assigned categories to documents. In order 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 12902 documents. Apart from very few exceptions, researchers have used the "ModApté" split, in which 9603 documents are selected for training and the other 3299 form the test set. In this thesis we will always refer to the ModApté split. There are 5 groups of categories that label REUTERS-21578 documents: EXCHANGES, ORGS, PEOPLE, PLACES, and TOPICS. Only the

TOPICS group has actually been used in experimental research, since the other four groups do not constitute a very challenging benchmark for Text Categorization. The TOPICS group contains 135 categories. Some of the 12902 "legitimate" documents have no categories attached to them, but unlike the 8676 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 experiment, since the Text Categorization 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.
Since 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 are

R10 the set of the 10 categories with the highest number of positive training examples.

R90 the set of the 90 categories with at least one positive training example and one positive test example

R115 the set of the 115 categories with at least one training example

Reasons for using one or the other subset have been different. Several researchers claim that ***R10*** is more realistic since machine learning techniques cannot perform adequately when positive training examples are scarce, and/or since small numbers of positive test examples make the interpretation of effectiveness results problematic due to high variance. Other researchers claim instead that only by striving to work on infrequent categories too we can hope to push the limits of Text Categorization technology, and this consideration leads them to use ***R90*** or ***R115***. Obviously, systems that have been tested on these different REUTERS-21578 subsets are not immediately comparable. In our experiments, we used the ***R90*** subset, that makes our results directly comparable with those proposed in [43], where learning approaches such as naive Bayes classifiers, Rocchio algorithm, the c4.5 decision tree/rule learner and SVM have been tested and compared on this subset.
A full description of the REUTERS-21578 collection and a discussion of the experimentation results on its subsets can be found in [26].

3.2 OHSUMED

OHSUMED[1] is a bibliographical document collection, developed by William Hersh and colleagues at the Oregon Health Sciences University [40]. The test collection is a subset of the MEDLINE database, which is a bibliographic database of medical documents maintained by the National Library of Medicine (NLM). There are currently over seven million references in MEDLINE dating back to 1966, with about 250000 added yearly. The majority of references are to journal articles, but the test collection also contains a number of references to letters to the editor, conference proceedings, and other reports. About 75% of the references contain abstracts, while the remainder (including all letters to the editor) have only titles. Each reference has been manually assigned to one of more subject headings from the 17000-term Medical Subject Headings (MeSH) thesaurus [2].
As for REUTERS-21578 , different subsets of OHSUMED have been used by researchers for experimental purposes. Among these, a commonly used subset firstly appeared in [43]. Out of 50216 original documents for the year 1991, the first 20000 documents which are classified into the 23 MeSH 'disease' categories and labelled with one or multiple categories have been chosen by T. Joachims [43]. Here, various learning approaches have been compared on this data collection, using the first 10000 for training and the second 10000 for testing the produced classifiers. As for REUTERS-21578 , in our experiments on this corpora we chose the same subset of OHSUMED proposed in [43]. More details on this experimentation phase are provided in chapter 8.

Other researchers used the OHSUMED collection for TC experiments, but the employed document set and categories vary: among the others, Yang in [90] chose only documents of 1991 and 1992, using the 1991 ones for training and the remaining to form the test set.
The various subsets showed that OHSUMED dataset is a "difficult" one. 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, for which classifiers reaches 86% of accuracy. This is because the data are more "noisy" and the word/category correspondences are more "fuzzy" in OHSUMED. Consequently, the categorization is more difficult to learn for a classifier.

[1]The OHSUMED collection may be freely downloaded for experimentation purposes from ftp:// medir.ohsu.edu/pub/ohsumed.

Part II

Machine Learning Approaches to Text Categorization

A growing number of statistical classification methods and machine learning techniques to automatically construct classifiers using labelled training data have been applied to text categorization in recent years. The most of them are devoted to binary problems, where a document is classified as either *relevant* or *not relevant* with respect to a predefined topic, while the common approach for the multi-label case, where a document belong to more than one class, is to break the task into disjoint binary categorization problems, one for each class. In these approaches, the classification of a new document needs the application of all the binary classifiers, whose predictions are combined into a single decision.
In the following chapters, we discuss some classical approaches to Text Categorization; unless specified otherwise, we will refer to binary classification problem. Since it is impossible to give a exhaustive overview of the inductive approaches proposed in Text Categorization literature, we will focus our attention on some of them. These algorithms are organized into classes, following their classical classification or according to the properties they share. In particular:

- In chapter 4, we discuss some of the best known approaches considered the state-of-art in Text Categorization area. Algorithms introduced here are Support Vector Machines [43] and k-NN [88], with which Olex shows to be competitive.

- In chapter 5, we explore the class of rule-based algorithms, of which our method is an example. In particular, we analyze the two subclasses of *decision tree inducers* and *inductive rule learners*, presenting some interesting methods, such C4.5 [67] and Ripper [20].

- In chapter 6, we discuss some approaches (both rule-based and not) that, like Olex does, use the evidence provided by negative training instances in the categorization decision. At first, we discuss a variant of k-NN, proposed in [35], where some weight is given to negative information; then we focus our attention on some rule-based approaches aiming at the construction of rules containing negative information. Finally, we shortly describe some methods for the extraction of positive and negative features from the training data.

Chapter 4

Probabilistic Induction Methods

In this chapter, we discuss some inductive approaches among the most representative of text categorization literature. At first, we describe *Support Vector Machines* (SVMs) approach, which embodies the latest results in statistical learning theory [79] and is considered one of the most accurate classifier. Then, we explore lazy learning approach and we discuss the *k*-NN algorithm, at first applied to pattern recognition problems and introduced in Text Categorization in the early '90s [88]. This algorithm was chosen as representative of this family because it is considered among the top-performing methods in Text Categorization problem.

4.1 Support Vector Machines

The *support vector machines* (SVM) method has been introduced in Text Categorization by Joachims [43,44] and subsequently used in [27,29,28,48,76,89]. The SVMs integrate dimension reduction and classification. This technique has been mostly applied to binary classification tasks and only recently it has been used to multi-class categorization problems [25].
This technique is based on recent advances in statistical learning theory. They map documents into a high dimensional feature space, and try to learn a separating hyperplane, that provides the widest margins between two different types of documents. SVMs use Lagrange multipliers to translate the problem of finding this hyperplane into an equivalent quadratic optimization problem for which efficient algorithms exist, and which are guaranteed to find the global optimum.
In geometrical terms, it may be seen as the attempt to find, among all the surfaces $\sigma_1, \sigma_2, ...$ in $|T|$-dimensional space that separate the positive from the negative training examples (*decision surfaces*), the surface σ_i that separates the positives

from the negatives by the widest possible *margin*, i.e. such that the separation property is invariant with respect to the widest possible translation of σ_i [74].
The simplest case we can take into consideration, which can give an idea about how SVMs work, is that in which the positives and the negatives are linearly separable, i.e. the decision surfaces are $(|T| - 1)$- hyperplanes. As an example, see figure 4.1, which shows a 2-dimensional case: here, various lines may be chosen as decision surfaces. The decision hyperplane chosen by SVMs is the bold solid line, which corresponds to the largest possible separation margins. The squares indicate the corresponding support vectors. The SVM method chooses the middle element from the "widest" set of parallel lines, i.e. from the set in which the maximum distance between two elements in the set is highest. It is noteworthy that this "best" decision surface is determined by only a small set of training examples, called the support vectors. The method described is applicable also to the case in which the positive and the negative examples are not linearly separable.

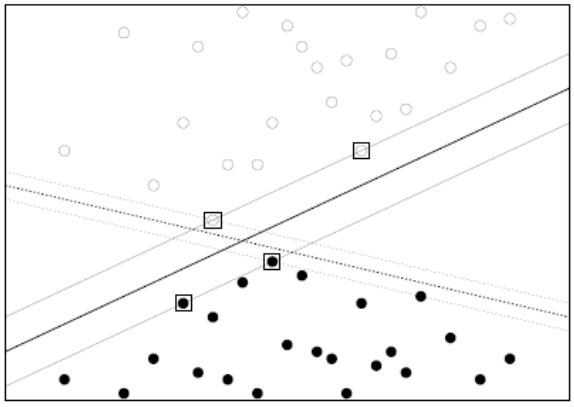

Figure 4.1: Example of a two class, linearly separable problem and two possible separation hyperplanes with corresponding margins.

As argued by Joachims in [43], the main advantages of SVMs are the following: first, term selection is often not needed, as SVMs tend to be fairly robust to overfitting and can scale up to considerable dimensionalities; second, no human and machine effort in parameter tuning on a validation set is needed, as there is a theoretically motivated, "default" choice of parameter settings, which has also been shown to provide the best effectiveness. The main drawback of SVM is that the classifiers generated are not understandable by humans.

4.2 Example-based classifiers

Example-based classifiers are often called *lazy learners*, since they do not build an explicit, declarative representation of the category of interest, but rely on the category labels attached to the training documents similar to the test document. Example-based methods (also known as *memory-based reasoning methods*) have been applied to text categorization since the early stages of the research [61, 87, 41].

A well known example based approach is *k*-NN (for "*k* nearest neighbours") algorithm implemented by Yang in the ExpNet system [88]. The basic idea in *k*-NN algorithm is that of finding, for a given test document, the k nearest neighbors among the training documents, and to use the categories of the k neighbors to weight the category candidates. The similarity score of each neighbor document to the test document is used as the weight of the categories of the neighbor document. If several of the k nearest neighbors share a category, then the per-neighbor weights of that category are added together, and the resulting weighted sum is used as the likelihood score of that category with respect to the test document. By sorting the scores of candidate categories, a ranked list is obtained for the test document. By thresholding on these scores, binary category assignments are obtained. The decision rule in *k*-NN can be written as:

$$y(\overrightarrow{x}, c_j) = \sum_{d_i \in kNN} sim(\overrightarrow{x}, \overrightarrow{d_i}) y(\overrightarrow{d_i}, c_j) - b_j$$

where $y(\overrightarrow{d_i}, c_j) \in \{0, 1\}$ is the classification for document d_i with respect to category c_j, $sim(\overrightarrow{x}, \overrightarrow{d_i})$ is the similarity between the test document $\overrightarrow{x}$ and the training document $\overrightarrow{d_i}$; and b_j is the category-specific threshold, automatically learned by using a specific validation set. The construction of a *k*-NN classifier also involves determining a threshold *k* that indicates how many top-ranked training documents have to be considered for computing $y(\overrightarrow{x}, c_j)$; *k* is usually determined experimentally on a validation set. For instance, Larkey and Croft [52] use k = 20, while Yang [87, 89] has found $30 \leq k \leq 45$ to yield the best effectiveness. Anyhow, various experiments have shown that increasing the value of k does not significantly degrade the performance.

A number of different experiments have shown *k*-NN to be quite effective. However, its most important drawback is its inefficiency at classification time: while e.g. with a linear classifier only a dot product needs to be computed to classify a test document, *k*-NN requires the entire training set to be ranked for

similarity with the test document, which is much more expensive. This is a characteristic of "lazy" learning methods, since they do not have a true training phase and thus defer all the computation to classification time [74].

Chapter 5

Rule Based Approaches

In this chapter, we focus our attention on the class of rule-based algorithms, of which our method is an example. This kind of approach is gaining considerable appeal in research area, since rule-based classifiers provide the desirable property of being readable, easy for people to understand, contrary to most of the other approaches, such as probabilistic induction methods which, even showing to be effective, lack of interpretability. We can distinguish two principal subclasses of rule-based algorithms: the *decision tree inducers* and the *inductive rule learners*, both analyzed in the following sections.

5.1 Decision Tree Inducers

A *decision tree* (DT) text classifier is 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 test document, and leaf nodes are labelled by categories. Such a classifier categorizes a test document d_j by recursively testing for the weights that the terms labelling the internal nodes have in vector $\overrightarrow{d_j}$, until a leaf node is reached; the label of this node is then assigned to d_j . Most such classifiers use binary document representations, and thus consist of binary trees.
There are a number of standard packages for DT induction, and most DT approaches to Text Categorization have made use of one such package. Among the most popular ones are ID3 (used in [33]), C4.5 (used in [22, 24, 43, 56]) and C5 (used in [59]). TC efforts based on experimental DT packages include [29, 57, 83].

A possible procedure for the induction of a DT for category c_i consists in a "divide and conquer" strategy, made up of the following step:

1. check whether all the training examples have the same label (either c_i or $\overline{c_i}$);

2. if not, select a term t_k, partition the training set into classes of documents that have the same value for t_k, and place each such class in a separate subtree.

These step are recursively repeated on the subtrees until each leaf of the tree so generated contains training examples assigned to the same category c_i, which is then chosen as the label for the leaf.

The key step, in DT algorithms, is the choice of the term t_k on which to operate the partition. This choice, generally made according to an information gain (e.g C4.5) or Gini coefficient (e.g. CART), tends to maximize the homogeneity (in terms of attached label) of the produced sets, hence to minimize the depth of the tree. However, such a "fully grow" tree may be prone to overfitting, as some branches may be excessively specific to the training data. In order to avoid overfitting, two strategies are used: either the growth of the tree is interrupted before excessively specific branches are produced, or the tree is pruned, removing the overly specific branches, in a subsequent step. An example of Decision Tree algorithm employing a pruning phase to revisit the produced classifiersis the C4.5 algorithm.

C4.5 Classifier

C4.5 is a decision tree classifier that was developed by Quinlan [66]. The training algorithm constructs a decision tree by recursively splitting the data set using a test of maximum gain ratio, subject to the constraint that information gain due to the split must also be large. The tree can be pruned back based on an estimate of error on unseen cases. During classification a test vector is evaluated according to the chosen tests at each split, and when it arrives at a leave, estimated are given for probabilities of its belonging to each category. In binary classification, for each category a tree is built using all the training data labeled as "yes" or "no" for that category. Although the principle is simple and the construction is very clear, the dilemma between overfitting and achieving maximum accuracy is seldom resolved. As the large feature set of text vector, overfitting is a hard controlled problem.

5.2 Associative Rule Learning

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 for their usefulness in many application domains such as recommender systems, diagnosis decisions support, telecommunication, intrusion detection, etc. The efficient discovery of such rules has been a major focus in the data mining research community. From the original *apriori* algorithm [3], there have been a remarkable number of variants and improvements of association rule mining algorithms i.e. [37].
Formally, association rules are defined as follows: Let $I = i_1, i_2, ...i_n$ be a set of items. Let D be a set of transactions, where each transaction T is a set of items such that $T \subseteq I$. Each transaction is associated with a unique identifier TID. 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 $X \cap Y = \emptyset$. 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. 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 main steps in building an associative classifier when a data set is given are the following:

1. Generating the set of association rules from the training set. In this phase association rules of the form $setoffeatures \Rightarrow class\_label$ are discovered by using a mining algorithm.

2. Pruning the set of discovered rules. In the previous phase a large set of association rules can be generated especially when low support is given. That is why pruning techniques are a challenging task to discover the best set of rules that can cover the training set. This phase is employed to weed out those rules that may introduce errors or are overfitting in the classification stage.

3. Classification phase. At this level a system that can make a prediction for a new object is built. The task here is how to rank and make use of the set of rules from the previous phase to give a good prediction.

The two most known models presented in the literature are CMAR [58] and CBA [60]. Although both of them proved to be effective and achieve high accu-

racy on relatively small UCI datasets, 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. An attempt to overcome this limitation and construct an associative classification model that allows single and multiple-class categorizations of text documents based on term co-frequency counts (i.e. a probabilistic technique that doesnt assume term independence) is provided in [7]. In this approach, given a data collection, a number of steps are followed until the classification model is found. Data preprocessing represents the first step, in which cleaning techniques can be applied such as stopwords removal, stemming or term pruning according to the TF/IDF values (term frequency/inverse document frequency). The next step in building the associative classifier is the generation of association rules using an apriori-based algorithm. Once the entire set of rules has been generated, an important step is to apply some pruning techniques for reducing the set of association rules found in the text corpora. The last stage in this process is represented by the use of the association rules set in the prediction of classes for new documents. The first three steps belong to the training process while the last one represents the testing (or classification) phase. More details on the process are given below. If a document D_i is assigned to a set of categories $C = \{c_1, c_2, ..., c_m\}$ and after word pruning the set of terms $T = \{t_1, t_2, ..., t_n\}$ is retained, the following transaction is used to model the document: $D_i = c_1, c_2, ..., c_m, t_1, t_2, ..., t_n$ 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

The algorithm takes advantage of the apriori algorithm to discover frequent termsets 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. The used apriori-based algorithm is guided by the constraints on the rules to be discovered, i.e. rules that indicate a category label, rules with a consequent being a category label. In other words, given the document model described above, the task is to find rules of the form

$T' \Rightarrow c_i$ where $T' \subseteq T$ and $c_i \in C$. To discover these interesting rules efficiently, the rule shape constraint is used 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 classification. There are two possible approaches in building an associative text classifier. The first one ARCAC (Association Rule-based Classifier with All Categories) [91] 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, it has been showed that 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). The second technique (proposed to solve such problems) is ARC-BC, that stands for *Associative Rule-based Classifier By Category*. In this 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.
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 has been solved by using pruning methods.

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, [7] 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 labelled. 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 pre-

sented 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 [7] 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 [7] is interesting as rules generated are understandable and can easily be manually updated or adjusted if necessary.

5.3 Decision Rule Classifiers

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"), whose premises denote the presence or absence of terms in the test document, while the head denotes the decision whether to classify it or not under c_i. DNF rules are similar to decision trees in that they can encode any Boolean function. However, one of the advantages of DNF rule inducers is that they tend to generate more compact classifiers than DT 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 DTs are typically induced by a top-down, divide-and-conquer strategy, DNF rules are often induced in a bottom-up fashion.

At the beginning of the classifier induction for category c_i, every training example is viewed as a clause $\eta_1, ..., \eta_n \rightarrow \gamma_i$, where $\eta_1, ..., \eta_n$ 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 high in terms of overfitting. The induction algorithm employs then a process of generalization in which 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 DTs is applied, where the ability to correctly classify all the training examples is traded for more generality.

For rule induction, the objective is to find sets of decision rules that distinguish one category of text from the others. Obviously, the set of rules promoted as "best

rule set" has to be, at the same time, accurate and not excessively complex. Accuracy of rule sets can be effectively measured on large numbers of independent test cases. Complexity can be measured in terms of numbers of rules or rule components, where smaller rule sets that are reasonably close to the best accuracy are sometimes preferred to more complex rules sets with slightly greater accuracy.
DNF Rule learners vary widely in terms of methods, heuristics and criteria employed for generalization and pruning. In the following section we will analyze RIPPER algorithm, eventually discussing some attempts of improving its performance results by introducing external knowledge in it.

5.3.1 RIPPER

Ripper algorithm attempts to find a small *hypothesis*, i.e. a set of rules, in the form of a small disjunction of conjunctions, which accurately classifies the training data. The conjunctions included in RIPPER's hypothesis always represent "contexts" that are positively correlated with the class being learned.
The algorithm used by RIPPER consists of two main stages: (1) a greedy process constructs an initial rule set; (2) an optimization phase attempts to further improve the compactness and accuracy of the rule set.

Stage 1: Building an Initial Rule Set

The first stage is a "set-covering" algorithm, called IREP* (based on the earlier rule-learning algorithm called Incremental Reduced Error Pruning IREP). Rules are constructed one at time, and once the construction of the rule is ended, the covered positive example are removed form the training data. In this phase of learning, different ad hoc heuristic measures are used to guide the greedy search for new conditions, and greedy search for simplifications [19]. All the heuristics used in constructing a rule are intended to ensure that the rule covers many positive examples and few negative examples. To construct a rule, the uncovered examples are randomly partitioned into two subsets, a "growing set" containing two-thirds of the examples and a "pruning set" containing the remaining one-third. IREP* will first grow a rule, and then simplify or prune the rule. A rule is "grow" by repeatedly adding conditions to rule r_0 with an empty antecedent. This is done is a greedy fashion: at each stage i, a single condition is added to the rule r_i, producing a longer and more specialized rule r_{i+1}. The greedy addition of new literals continues until the clause covers no negative examples in the growing set, or until no "good" condition is found.

After growing a rule, the rule is pruned (i.e., simplified). This is another greedy process, in which IREP* considers deleting any final sequence of conditions from the rule and chooses the deletion that maximizes the function

$$f(r_i) = \frac{U^+_{i+1} - U^-_{i+1}}{U^+_{i+1} + U^-_{i+1}}$$

where U^+_{i+1} (respectively, U^-_{i+1}) is the number of positive (negative) examples in the pruning set covered by the new rule. After pruning, the pruned clause is added to the rule set, and the examples covered by it are removed [23].

Stage 2: Optimization of a Rule Set.

When the construction of the rule set is finished, it has to be "optimized" to further reduce its size and improve its accuracy. Rules are considered in turn in the order in which they were added. For each rule r, two alternative rules are constructed. The replacement for r is formed by growing and then pruning a rule r', where pruning is guided so as to minimize error of the entire rule set on the pruning data. The revision of r is formed analogously, except that it is grown by greedily adding literals to r, instead of to the empty rule. Finally a decision is made as to whether the final theory should include the revised rule, the replacement rule, or the original rule. This decision is made using the description length heuristic, whereby the definition with the smallest description length after compression is preferred. After optimization, the definition may cover fewer positive examples; thus IREP* is called again on the uncovered positive examples, and any additional rules that it generates are added. This optimization step can be repeated, occasionally resulting in further improvements in a rule set.
Some attempts have been done to introduce in RIPPER the usage of external knowledge, provided by public thesauri. In the next two sections, we will discuss two different approaches to this problem. In the first, Scott and Matwin [72] used RIPPER algorithm in union with pre-processing techniques and WordNet thesaurus, while the second is a true extension on RIPPER build to introduce external knowledge in the learning process.

5.3.2 Using WordNet Thesaurus in RIPPER

An example of exploration of the use of external knowledge in Text Categorization is given by [72]. In this work, the authors analyzed the hypothesis that incorporating linguistic knowledge into text representation can lead to improvements

in classification accuracy. Specifically, they applied RIPPER algorithm to the training data, pre-processed by means of linguistic techniques and using *part of speech* information from the Brill tagger [12]and the *synonymy and hypernymy* relations from WordNet [14]. Trough this pre-processing process the representation of the text has been changed from *bag-of-words* to *hypernym density*, where synsets (*synonyms sets*) replace words. The algorithm for computing hypernym density requires three passes through the corpus:

1. assignment of the part of speech tag to each word in the corpus, through the Brill tagger.

2. creation of a global list of all synonym and hypernym synsets, made up by looking up all nouns and verbs in WordNet. In this phase, infrequently occurring synsets are discarded, and those that remain form the feature set. (A synset is defined as infrequent if its frequency of occurrence over the entire corpus is less than $0.05N$, where N is the number of documents in the corpus.)

3. computation of the density of each synset for each example resulting in a set of numerical feature vectors. The density of a synset is defined as the number of occurrences of the synset in the WordNet output divided by the number of words in the document.

The calculations of frequency and density are influenced by the value of a parameter h that controls the *height of generalization*. This parameter can be used to limit the number of steps upward through the hypernym hierarchy for each word. At height $h = 0$ only the synsets that contain the words in the corpus will be counted. At height $h > 0$ the same synsets will be counted as well as all the hypernym synsets that appear up to h steps above them in the hypernym hierarchy. A special value of $h = max$ is defined as the level in which all hypernym synsets are counted, no matter how far up in the hierarchy they appear. In the new representation, each feature represents a set of either nouns or verbs. At $h = max$, features corresponding to synsets higher up in the hypernym hierarchy represent supersets of the nouns or verbs represented by the less general features. At lower values of h, the nouns and verbs represented by a feature (synset) will be those that map to synsets up to h steps below it in the hypernym hierarchy. The best value of h for a given text classification task will depend on characteristics of the text such as use of terminology, similarity of topics, and breadth of topics. It will also depend on the characteristics of WordNet itself. In general, if the value for h

is too small, the learner will be unable to generalize effectively. If the value for h is too large, the learner will suffer from *overgeneralization* because of the overlap between the features.
Note that no attempt is made at word sense disambiguation during the computation of hypernym density. Instead all senses returned by WordNet are judged equally likely to be correct, and all of them are included in the feature set. The use of the density measurement is an attempt to capture some measure of relevancy. The learner is aided by the fact that many different but synonymous or hyponymous words will map to common synsets, thus raising the densities of the "more relevant" synsets. In other words, a relatively low value for a feature indicates that little evidence was found for the meaningfulness of that synset to the document.
Some experiments have been carried out, to compare the results obtained by RIPPER, using the two different type of text representation. Hypernym density has been observed to greatly improve classification accuracy in some cases, while in others the improvements are not particularly evident. Hypernym density representation brings a side benefit: induced classification rules are often simpler and more comprehensible than rules induced using the bag-of-words. The experiments showed the hypernym density representation can work well for texts that use an extended or unusual vocabulary, or are written by multiple authors employing different terminologies. It is not likely to work well for text that is guaranteed to be written concisely and efficiently, such as the text in Reuters-21578. In particular, hypernym density is more likely to perform well on classification tasks involving narrowly defined and/or semantically distant classes [72].

5.3.3 TRIPPER

TRIPPER is a rule induction algorithm that extends RIPPER, by using external-knowledge. The main goal in TRIPPER (i.e. *Taxonomical Ripper*) is the construction of classifiers at higher levels of abstraction, where rules are generated on the basis of user-supplied knowledge, available in the form of attribute value taxonomies. The extensions to RIPPER can be summerized as follow [80]:

Improvement at rule growth phase (TRIPPER G): Introducing the taxonomical knowledge at the rule-growth phase is a straightforward process called *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 (TRIPPER G+P): 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, TRIPPER chooses from a whole range of levels of specificity for the feature under consideration.

Some experimental results about TRIPPER have been reported in [80]. The main goal of the experiments carried out was to compare TRIPPER and RIPPER performances. Both algorithms have been evaluated on the benchmark dataset REUTERS-21578, with experimental setting similar to those used in [62]. The text-specific taxonomies, used for TRIPPER growing and pruning phases, comes from WordNet [14], using only the hypernimy relation that stands for *"is-a"* relation between concepts.

The experiments showed that TRIPPER generally outperforms RIPPER on the Reuters text classification task in terms of *break-even point*, while generating potentially more comprehensible and concise rule sets than RIPPER, thanks to the improvements in both phases of learning. Further, 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 6

Exploitation of Negative Information

Machine Learning techniques to Text Categorization generally aim at the construction of classifiers, basing their classification decision of the new document on the similarity with the positive training documents. In other words, they use the fact that a test document is "similar" to a training document, representing a positive instance for a given category, as evidence towards the fact that the test document belongs to that category. Generally, the similarity to a negative training instance is not used anyway.
In this chapter, we discuss some approaches, where *negative evidence*, i.e. evidence provided by negative training instances, is not discarded, but used in the categorization decision. At first, we discuss a variant of *k*-NN, based on the use of negative information, proposed in [35], then we focus our attention some rule-based approaches aiming at the construction of rules containing negative information and, finally, we shortly describe some methods for the extraction of positive and negative features from the training data.

6.1 A variant of *k*-NN using negative information

Galavotti et al in [35] proposed a family of variants of Yang's version of K-NN, called $k\text{-}NN^{p}{}_{neg}$. The original method, discussed in section 4.2, is distance-weighted algorithm, since the fact that a training document d'_z similar to the test document d_j belongs to c_i is weighted by the similarity between d'_z and d_j. Mathematically, classifying a document by means of *k*-NN thus comes down to computing

$$CSV_i(d_j) = \sum_{d'_i \in Tr_k(d_j)} RSV(d_j, d'_z) \cdot v_{iz} \tag{6.1}$$

where

- $CSV_i(d_j)$ measures the computed evidence that d_j belongs to c_i, i.,e. the *categorization status value* of document d_j with respect to category c_i
- $RSV(d_j, d'_z)$ represents some measure of semantic relatedness between d_j and d'_z, i.e. *the retrieval status value* of document d'_z with respect to document d_j
- $Tr_k(d_j)$ is the set of the *k* training documents d'_z for which $RSV(d_j, d'_z)$ is highest. The number *k* of training top-ranked documents to be considered is often determined experimentally.
- v_{iz} is the weight of the training document d'_z. The value of v_{iz} is 1 if d'_z is a positive instance for category c_i, 0 otherwise.

The first variant proposed in [35], called *k*-$NN^1{}_{neg}$, is based on the simple intuition of assigning a negative weight to those documents of the training set, that are negative instances for category c_i. This is realized by using, in equation 6.2, a value of -1 for v_{iz}, if d'_z is a negative instance for c_i. Contrary to the expectations, experiments have shown that the use of negative evidence doesn't bring any substantial improvement in *k*-NN classifiers. In fact, the highest performance obtained for *k*-$NN^1{}_{neg}(0.775)$ is practically the same as that obtained for *k*-NN (0.776). An interesting characteristic of *k*-$NN^1{}_{neg}$ is that it needs smaller *similarity document set* than *k*-NN, in fact it peaks at substantially lower values of *k* than *k*-NN (10 vs 50); even if *k*-$NN^1{}_{neg}$ is less robust than *k*-NN with respect to the choice of *k*. In fact, for *k*-$NN^1{}_{neg}$ effectiveness degrades somehow for values of *k* higher than 10, while the original system is hardly influenced by the value of *k*.

The basic intuition of *k*-$NN^1{}_{neg}$ is stressed in *k*-$NN^p{}_{neg}$ methods. In contrasts with *k*-NN, where very dissimilar documents have not much influence, since positive instances are usually far less than negative ones, in *k*-$NN^p{}_{neg}$ they do, since each of the most *k* most similar documents, however semantically distant, brings a little weight to the final sum of which the CSV consists. The *k*-$NN^p{}_{neg}$ methods

are based on the use of CSV functions that downplay the influence of the similarity value in the case of widely dissimilar documents. This class of functions can be represented as:

$$CSV_i(d_j) = \sum_{d'_i \in Tr_k(d_j)} RSV(d_j, d'_z)^p \cdot v_{iz} \tag{6.2}$$

where the larger the value of *p* parameter is, the more the influence of the similarity value is played in the case of widely dissimilar documents.
A small group of experiments have been carried out in order to compare *k*-$NN^2{}_{neg}$ with *k*-$NN^1{}_{neg}$ and with *k*-NN. The experiments showed that *k*-$NN^2{}_{neg}$ outperform both methods: it peaks for higher value of *k* than *k*-$NN^1{}_{neg}$ and it is remarkably more stable for higher values of *k*. This seemingly suggests that negative evidence provided by very dissimilar documents is indeed useful, provided its importance is de-emphasized. Instead *k*-$NN^3{}_{neg}$ slightly underperforms *k*-$NN^2{}_{neg}$, showing that the level of de-emphasization must be chosen carefully.

6.2 Association Rules with Negation

The association rules mining algorithms described in section 5.2 focus on discovering association rules of the form $A \Rightarrow B$, whose support (supp) and confidence (conf) meet some user specified *minimum support* (*minsupp*) and *minimum confidence* (*minconf*) thresholds. Association rules from the support-confidence framework are *positive rules*.
Different techniques for the classification of structured data and texts, aiming at improving traditional associative classification models taking advantage of negative information, have been proposed in the last years. Brin et al. [13] 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 [70], 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. In the following, we focus our attention on two approaches proposed in literature. The first one, described in [86], aims at the construction of classifier

composed by positive and negative rule, on the basis of frequent and infrequent itemsets; the second one, presented in [8] consider another framework that adds to the support-confidence, for the choice of positive and negative items set, some measures based on correlation analysis.

6.2.1 Mining Positive and Negative Associative Rules

The Association Rule Mining approach presented in [86] aims at extending the traditional definition of association rule to support negative rules. Association rules, traditionally defined as implications of the form $A \Rightarrow B$, where A and B are frequent itemsets in a transactional databases (*positive rules*), in [86] include implications of the form $A \Rightarrow \neg B$, $\neg A \Rightarrow B$ and $\neg A \Rightarrow \neg B$, generated from the infrequent itemsets (*negative rules*). A set of condition, that we discuss afterwards, are used to state the interest of an itemset, while the confidence of positive and negative rules is estimated using the increasing degree of the conditional probability relative to the prior probability.

Frequent and infrequent itemsets

As defined in [17], a *frequent itemset* is an itemset that meets the user-specified minimum support. Accordingly, an *infrequent itemset* is defined as an itemset that doesn't meet the user-specified minimum support.
Let $I = i_1, i_2, ..., i_N$ be a set of N distinct literals called items, and D a database of variable-length transactions over I. Each transaction contrains a set of items $i_1, i_2, .., i_k \in I$, called *itemset* of length k and referred to as $k-$itemsets. Each itemset has an associated measure called support, denoted as *supp*. For an itemset $A \subseteq I$, $supp(A) = s$ if the fraction of transactions in D containing A equals to s.
A (positive) association rule in the support-confidence framework is an implication of the form $A \Rightarrow B$, where $A, B \subseteq I$ and $A \cap B = \varnothing$. The support of the rule $A \Rightarrow B$ is defined as $supp(A \cup B)$, while the confidence is defined as the ratio of the $supp(A \cup B)$ of itemset $A \cup B$ over the $supp(A)$ of itemset A. That is, $conf(A \Rightarrow B) = supp(A \cup B)/supp(A)$.
Once defined *support* and *confidence* for a rule, [86] proceeds to the extraction of valid positive and negative rules, according to a set of conditions stating the interest of a rule.
A positive rule $X \Rightarrow Y$ is of interest if and only if

(1) $X \cap Y = \varnothing$

(2) $supp(X \cup Y) \geq minsupp$

(3) $supp(X \cup Y)/supp(X) \geq minconf$

(4) $supp(X \cup Y) - supp(X) \times supp(Y) \geq mininterest$

where $X \cup Y$ is a frequent itemset and $mininterest, minconf$ and $minsupp$ thresholds are specified by the user. Intuitively, rule $X \Rightarrow Y$ is of interest if its support and confidence are greater or equal to the fixed minimum values and the itemset $X \cup Y$ is more interesting than the pair of independent itemset X and Y.

Based on the conditions for frequent itemset for mining positive rules, a set of conditions for a rule of the form $X \Rightarrow \neg Y$ to be a valid negative rule of interest:

(1) $X \cap Y = \varnothing$

(2) $supp(X) \geq minsupp$, $supp(Y) \geq minsupp$, $supp(X \cup \neg Y) \geq minsupp$

(3) $supp(X \cup \neg Y)/supp(X) \geq minconf$

(4) $supp(X \cup \neg Y) - supp(X) \times supp(\neg Y) \geq mininterest$

where $A \cup B$ and B is an infrequent itemset of interest. In fact, the condition $supp(X \cup \neg Y) \geq minsupp$ implies that $supp(X \cup Y) \leq minsupp$ and, when the *minsupp* is high, $X \cup Y$ cannot be generated as frequent itemset. The conditions for rules of the form $\neg X \Rightarrow Y$ and $\neg X \Rightarrow \neg Y$ are defined accordingly. Once frequent and infrequent itemsets are identified, on the basis of the constraints discussed above, positive and negative rules are defined. This approach has been tested on three different datasets and it has been compared with Apriori [5] in the support-confidence framework proposed in [4]. When mining only positive rules of interest, the classifiers produced in the compared approaches are identical (if the same constraints are applied). In the meanwhile, the approach proposed by [86] is more efficient than Apriori algorithm in discovering positive association rules. The proposed approach seems promising, even if no experimental result has been provided in [86] about the introduction of negative information in rule mining.

6.2.2 ARC-PAN Classifier

In this section we introduce *ARC-PAN* (Associative Rule Classification with Positive And Negative), so called because the set of rules generated is the union of PCR (Positive Classification Rules) and NCR (Negative Classification Rules). In

this approach, the generation of positive and negative rules is based on correlation measures, computed by using a *correlation coefficient*. Given two variables X and Y, the correlation coefficient measures the strength of the linear relationship between a pair of two variables, according to the following formula:

$$\rho = \frac{Cov(X,Y)}{\sigma_X \sigma_Y} \tag{6.3}$$

where $Cov(X,Y)$ represents the covariance of the two variables and σ_X stands for the standard deviation. The range of values for ρ is between -1 and $+1$. Here, we report the values of interest for correlation coefficient:

$$\rho = \begin{cases} 0 & X \text{ and } Y \text{ are independent} \\ +1 & X \text{ and } Y \text{ are perfectly positive correlated} \\ +1 & X \text{ and } Y \text{ are perfectly negative correlated} \end{cases}$$

A positive correlation is evidence of a general tendency that when the value of X increases/decreases so does the value of Y. A negative correlation occurs when for the increase/decrease of X value, we discover a decrease/increase in the value of Y.

ARC-PAN algorithm is an apriori-like process for the generation of a set of classification rules of the form $set\_of\_features \Rightarrow class\_label$, which will be used in the subsequent classification stage [8].
It generates first the set of frequent 1-itemsets. Once the 1-frequent itemsets is generated the candidate sets C_2 to C_n 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 positive and negative association rules are generated , using a function based on the item correlation with a class label. This function takes as input an itemset and the set of class labels and, for each pair $(item, class\_label)$ computes the correlation coefficient. 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 based on the values discussed before. First, they consider as high correlation threshold, in order to discover strong correlations, if no strong correlation is

discovered, the threshold can be lowered to discover moderate correlations.
The algorithm described above has been tested on various dataset, in order to evaluate its performance values and comparing it with other learning algorithm, such as CBA and C4.5, whose results have been taken from [60]. When all types of rules are used the classification accuracy increases on three datasets when compared with the state-of-the-art classifier C4.5 and with the CBA. In particular, the experiments showed that the classification accuracy can be improved as well with only the generation of positive association rules that are strongly correlated, while generating only the negative rules only, the results decrease.
As noticed in [8], there is a drastic reduction in rule number when the correlation measure is used to derive interesting rules, without any consequence for the error rate, which remains in the same range.This demonstrates that a much smaller set of positive and negative association rules can perform similar or outperform existing categorization systems.

6.3 Use of Negative Information in Features Selection

In [92] a different approach for feature set selection has been proposed. A set of features is constructed 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 was empirically chosen to obtain optimal performance. This is in contrast with the standard local feature selection approaches that either (1) only select the terms most indicative of membership; or (2) implicitly but not optimally combine the terms most indicative of membership with non-membership.
The proposed feature selection method is based the following two key concepts:

- In literature, many statistical function, such as Chi Square, Odds Ratio or GSS Coefficient, have been successfully used to extract positive features selecting, for each category, a set of terms based on the relevant and irrelevant documents in this category.

- The same method can be used to extract negative information, simply extracting for each category the less indicative of membership, too.

Given a feature selection function f, which measures the relationship between a term *t* and a category c_i as $f(t, C_i)$, this can be used in global feature extraction by computing and comparing the average and maximum of their category-specific values (for more details, see [90]). Given a vocabulary V and a function f that maps terms to real values, two subsets of V with size l are defined as $Max[V, f, l]$ and $Min[V, f, l]$, so that they consist of the l terms $t_j \in V$ with the highest and the lowest $f(t_j)$ values, respectively.
The feature selection is carried out into three steps:

STEP 1: a positive-feature set F_i^+ is generated for each category c_i.
$F_i^+ = Max[V, f(\cdot, c_i, l_1)]$, where $l_1, 0 < l_1 < l$ is a natural number;

STEP 2: a negative-feature set F_i^- is generated for each category c_i.
$F_i^+ = Max[V, f(\cdot, \overline{c}_i, l_2)]$, where $l_2 = l - l_1$ is a not negative number;

STEP 3: $F_i = F_i^+ \bigcup F_i^-$

Some improvement in classification performance have been obtained by experimenting this method on the Reuters-21578 dataset. In particular, in most cases the results achieved show that the combination of positive and negative features, by selecting more negative terms than positive ones, outperform the standard approaches of positive terms selection.

Part III

OLEX: a New Technique for Learning Text Classifiers

In this part we propose Olex, a novel approach to the automatic construction of rule-based text classifiers. Here, a classifier is a set of propositional rules, each characterized by *one* positive literal and (zero or) *more* negative literals. A positive (resp.negative) literal is of the form $T \in d$ (resp. $T \notin d$), where T is a conjunction of terms and d a document. Rule induction relies on an optimization algorithm whereby a set of discriminating terms is generated for the category being learned.

In chapter 7, after providing a concise overview of Olex and giving some preliminary definitions and notation, we state the optimization problem of selecting a best set of discriminating terms (which is the heart of our method) and prove that this task is computationally difficult. Thus, we propose a heuristic approach to solve it. Then, we give a description of the learning process.
In chapters 8 and 9 we present the experimental results and provide a comparison with other learning approaches. The results proposed were obtained using a prototype implementing the machine learning method proposed in chapter 7. Olex classification performances were evaluated on different data sets:

- First, in order to compare Olex with pre-existing well-known systems, we carried out a series of experiments on two benchmark data sets, the REUTERS-21578 and the OHSUMED data collection. In chapter 8, we discuss the experimental results on these benchmark corpora and eventually compare them with the ones of other learning approaches like SVM, K-NN, Rocchio, etc.

- Then, Olex has been tested on two real use-case corpora, both collections of reports about road accidents written down by policemen, belonging to the American FCSI company. On these data sets we carried out different types of experiment. In a first set of experiments, we followed the general methodology used on REUTERS-21578 and OHSUMED corpora; then we tested different features of the system on FCSI by exploiting external knowledge and the integration of manual and automatic approaches. A full description of the experimentation phase on FCSI corpora is given in chapter 9.

Chapter 7

Olex: Effective Rule Learning for TC

In this chapter, we describe Olex, a novel method for the automatic induction of rule-based text classifiers. Olex supports a hypothesis language of the form "if T_1 or $\cdots$ or T_n occurs in document d, and none of $T_{n+1}, \cdots T_{n+m}$ occurs in d, then classify d under c", where each T_i is a conjunction of terms.
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 top-performing (see chapters 8 and 9). Further, it provides classifiers that are very compact and comprehensible.

In this chapter, after providing an overview of Olex (Section 7.1) and giving some preliminary definitions and notation (Section 7.2), we state the optimization problem of selecting a best set of discriminating terms (which is the heart of our method) and prove that this task is computationally difficult (Section 7.3). Thus, we propose a heuristic approach to solve it (Section 7.5) and give a description of the whole learning process (Section 7.6).

7.1 Olex Overview

Olex belongs to the category of the inductive rule learning methods. A general formulation of the induction problem (for text categorization) is as follows. Given

- a *background knowledge* B as a set of ground logical facts of the form $t \in d$, meaning that term t appears in document d (other ground predicates may occur in B as well)

- a set of *positive examples* expressed as ground logical facts of the form $d \in c$, meaning that document d belongs t o category c (*ideal classification*); *negative examples* are implicitly stated according to the Closed World Assumption (i.e., if $d \in c$ is not a positive example, then it is a negative one)

constructs a hypothetical rule set (the classifier of c) that, combined with the background knowledge B, agrees all (both positive and negative) examples. It is well known that this problem is computationally difficult, unless the requirement that a learning algorithm identifies the target concept *exactly* is relaxed to allow *approximations*. The Valiant's theory of *PAC-learnability* (Probably Approximately Correct) provides a model of "polynomial learning" for a subset of propositional logic [78]. In the PAC framework, the polynomially bounded amount of resources (both number of examples and computational time) is traded-off against the accuracy of the induced hypotheses. In [47, 30, 18, 1] it has been shown that also specific subclasses of first-order logic are PAC-learnable.

The induced rules from both background knowledge and examples will allow prediction about the belonging of a document to a category on the basis of the presence or absence of some given terms in that document. However, while in computational learning theory it is assumed that the input sample is consistent with some hypothesis in the hypothesis space, in text classification this is not necessarily true; indeed, it is not possible, in general, to correctly categorize a document under a category only on the basis of the terms occurring in it. Thus, the expected induced hypothesis in such a case is one which *maximally* satisfies (both positive and negative) examples.

Within this framework, the Olex's learning problem is stated as an optimization problem relying on the F-measure as the objective function. In particular, the optimization task is that of determining a best set $X_c = \{T_1^+, \cdots, T_n^+, T_{n+1}^-, \cdots T_{n+m}^-\}$ of *discriminating* terms (d-terms) for c. A d-term is of the form T^s, where T is a conjunction $t_1 \wedge \cdots \wedge t_k$ of (simple) terms, taken from a given *vocabulary*, and $s \in \{+, -\}$ is the sign. We call T *conjunctive* term (co-term). A d-term T^s occurs in a document d if T occurs in d, i.e., if the term t_i occurs in d, for each $i = 1, k$. A positive d-term occurring in a document d is indicative of membership of d in c, while a negative one is indicative of non-membership. Thus, a document d containing *any* positive d-term in X_c and *none* of the negative d-terms in X_c, is *eligible* for classification under c according to X_c. Hence, the aim of the optimization task is that of finding a set X_c of d-terms such that, by classifying under c the set $E(X_c)$ of documents of the training set TS eligible for classification according to X_c, the resulting F-measure is maximum (intuitively,

this corresponds to finding X_c such that $E(X_c)$ best "fits" the examples in TS). Not surprisingly, the above task is computationally untractable.
Now, given a (best) set $X_c = \{T_1^+, \cdots, T_n^+, T_{n+1}^- \cdots T_{n+m}^-\}$ of d-terms, the corresponding hypothesis (the classifier of c) is of the form

$$c \leftarrow T_1 \in d, T_{n+1} \notin d, \cdots, T_{n+m} \notin d$$

$$\cdots$$

$$c \leftarrow T_n \in d, T_{n+1} \notin d, \cdots, T_{n+m} \notin d$$

and states the condition "if any of the co-terms $T_1, \cdots, T_n$ occurs in d and *none* of the co-terms $T_{n+1}, \cdots, T_{n+m}$ occurs in d then classify d under category c". That is, the occurrence of a co-term T_i, $1 \leq i \leq n$, in a document d requires the contextual *absence* of the (possibly empty) set of co-terms $T_{n+1}, \cdots, T_{n+m}$ in order for d be classified under c[1]. Notice that there is one rule for each positive d-term in X_c and, for each rule, one negative literal for each negative d-term in X_c (thus, all rules share the same negative part $T_{n+1} \notin d, \cdots, T_{n+m} \notin d$).
We remark that the Olex's hypothesis language, essentially Horn clauses extended by negative conjunctions of terms, is not PAC-learnable.
Since the set X_c which maximizes the objective function depends on the choice of the vocabulary (i.e., the set of terms selected for rule induction), to pick the "best" classifier Olex proceeds by repeatedly running the optimization algorithm with different input vocabularies, and eventually selecting the classifier with the best performance.
Next is the best classifier induced for category "corn" of the REUTERS-21578 data collection:

$$\begin{aligned} corn \quad &\leftarrow \quad \text{“}corn\text{”} \in d, \text{“}offering\text{”} \notin d, \text{“}international\text{”} \wedge \text{“}mln\text{”} \notin d, \\ &\quad\quad \text{“}animal\text{”} \notin d, \text{“}live\text{”} \notin d, \text{“}fuels\text{”} \notin d, \text{“}ministry\ agriculture\text{”} \notin d. \\ corn \quad &\leftarrow \quad \text{“}maize\text{”} \in d, \text{“}offering\text{”} \notin d, \text{“}international\text{”} \wedge \text{“}mln\text{”} \notin d, \\ &\quad\quad \text{“}animal\text{”} \notin d, \text{“}live\text{”} \notin d, \text{“}fuels\text{”} \notin d, \text{“}ministry\ agriculture\text{”} \notin d. \end{aligned}$$

As we can see, it consists of two rules with the following meaning: classify document d under category "corn" if either term "corn" or term "maize" occurs in d and, further, neither "offering" nor "international" $\wedge$ "mln" nor $\cdots$ nor "ministry agriculture" occur in d. Here, we note that "international" $\wedge$ "mln" is a conjunction of terms, while "ministry agriculture" is a simple term - notably a bigram. In the former case, the two words "international" and "mln" may occur in any order and in any position of the document, whereas the two words composing the

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

bigram must occur consecutively and in the fixed order.

7.2 Preliminary Notation and Definitions

Throughout this and the following chapters, we assume the existence of:

1. a finite set $\mathcal{C}$ of categories, called *classification scheme*;
2. a finite set $\mathcal{D}$ of documents (i.e., sequences of words), called *corpus*; $\mathcal{D}$ is partitioned into a *training set* TS, a *validation set* and a *test set*; the training set along with the validation set represent the so-called *seen data*, used to induce the model, while the test set represents the *unseen data*, used to asses performance of the induced model;
3. a relation $\mathcal{I} \subseteq \mathcal{C} \times \mathcal{D}$ (*ideal classification*) which assigns each document $d \in D$ to a number of categories in $\mathcal{C}$. We denote by TS_c the subset of the training set TS whose documents belong to category c according to $\mathcal{I}$ (the *training set of* c);
4. a set $\Phi = \{\phi_1, \cdots, \phi_k\}$ of scoring functions (or feature selection functions), such as Information Gain, Chi Square, etc. [90, 31], used for vocabulary reduction (we will hereafter refer to them simply as "functions" whenever no ambiguity arises).

Now, the problem is that of automatically inducing, for each $c \in \mathcal{C}$, a set of classification rules (the *classifier* of c) by learning the properties of c from both the documents of the training set (providing the relationship term-document) and the ideal classification (providing the positive examples). We assume that categories in $\mathcal{C}$ are mutually independent, i.e., the classification results of a category c does not depend on the classification results of any other category. Thus, the whole learning task consists of $|\mathcal{C}|$ independent sub-tasks (one for each category). For this reason, in the following we will concentrate on a single category $c \in \mathcal{C}$.
Once a classifier for category c has been constructed, its capability to take the right categorization decision is tested by applying it to the documents of the test set and then comparing the resulting classification to the ideal one. The effectiveness of the predicted classification is measured in terms of the classical notions of *Precision*, *Recall* and *F-measure* [74] defined as follows:

$$Pr = \frac{|TP_c|}{|TP_c| + |FP_c|}; \tag{7.1}$$

$$Re = \frac{|TP_c|}{|TP_c| + |FN_c|}; \tag{7.2}$$

$$F = \frac{Pr \cdot Re}{(1-\alpha)Pr + \alpha Re} \tag{7.3}$$

where $|TP_c|$ is the number of *true positive* documents w.r.t. c (i.e., the number of documents of the test 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, defined accordingly. Further, the parameter $\alpha \in [0 \cdots 1]$ in the definition of the F-measure is the relative degree of importance given to Precision and Recall; notably, if $\alpha = 1$ then F_α coincides with Pr, and if $\alpha = 0$ then F_α coincides with Re (a value of $\alpha = 0.5$ attributes the same importance to Pr and Re)[2].

7.3 Selection of Discriminating terms: problem definition and complexity

In this section we provide a description of the optimization problem aimed at generating a best set of *discriminating terms* (d-terms, for short) for a category $c \in \mathcal{C}$. In particular, we give a formal statement of the problem and show its complexity. To this end, a number of preliminary definitions are needed.

Definition 7.1 ***(reduced vocabulary)***

- A *term* (or n-gram) is a sequence of one or more words, or variants obtained by using *word stems* [], consecutively occurring within a document $d \in \mathcal{D}$;
- The *local vocabulary* V_c of category c over the training set TS is the set of *all* terms occurring in the documents of the training set TS_c of c. Each term in V_c is scored by the functions in Φ[3];

[2] As we will see in the next sections, $\alpha \in [0 \cdots 1]$ is a parameter of our model; for this reason we find convenient using F_α rather than the equivalent $F_\beta = \frac{(\beta^2+1)Pr \cdot Re}{\beta^2 Pr + Re}$, where $\beta \in [0 \cdots \infty]$ has no upper bound

[3] The choice of the actual scoring functions to be used for local vocabulary creation is orthogonal to the learning model; in Section **??** we will specify the ones that have been used for experimentation

Symbol	Description
$\mathcal{C}$	set of categories (classification scheme)
TS, TS_c	training set; training set of category c
t	term (n-gram)
T	co-term (conjunction of terms)
T^+, T^-	positive and negative d-term (discriminating term)
X_c	set of discriminating terms for category c
ϕ, ν, α	model parameters: ϕ is a term scoring function; ν is the number of terms per category chosen in the construction of the reduced vocabulary; α is the Precision/Recall threshold
$V(\phi, \nu)$	reduced vocabulary;
$E(X_c)$	set of documents eligible for classification under c according to X_c

Table 7.1: List of the main symbols used in this chapter

- Given a function $\phi \in \Phi$ and a nonnegative integer ν, let $V_c(\phi, \nu)$ denote the set consisting of the ν terms of the local vocabulary V_c over TS that score highest according to function ϕ. The *(reduced) vocabulary* $V(\phi, \nu)$ over TS, for the given ϕ and ν, is $\cup_{c \in \mathcal{C}} V_c(\phi, \nu)$.

That is, the vocabulary $V(\phi, \nu)$ is the set consisting of the best ν terms, according to a given scoring function ϕ, of each local vocabulary V_c (thus, the size of $V(\phi, \nu)$ is $\nu \cdot |\mathcal{C}|$, where $|\mathcal{C}|$ is the number of categories). As usual, vocabulary reduction is used to remove non-informative words from documents in order to improve both computational learning efficiency and classification effectiveness [].

Definition 7.2 ***(co-terms)*** Let us fix a vocabulary $V(\phi, \nu)$. A *co-term* (conjunctive term) T over $V(\phi, \nu)$ of *degree* k is a conjunction of terms $t_1 \wedge \cdots \wedge t_k$, with $t_i \in V(\phi, \nu)$, $1 \leq i \leq k$. We say that T *occurs* in a document d, denoted $T \in d$, if each term t_i occurs in d, $1 \leq i \leq k$. Let us denote by $T^{\{\}}$ the set $\{t_1, \cdots, t_k\}$ of the terms of T. We say that two co-terms T_1 and T_2 are *independent* if neither $T_1^{\{\}} \subseteq T_2^{\{\}}$ nor $T_2^{\{\}} \subseteq T_1^{\{\}}$.

For an instance, $t_1 \wedge t_2$ and $t_1 \wedge t_3$ are independent, while t_1 and $t_1 \wedge t_2$ are not.

Definition 7.3 ***(d-terms)*** Let us fix a vocabulary $V(\phi, \nu)$. A *d-term* (discriminating term) for c over $V(\phi, \nu)$ is a pair $< T, s >$, where T is a co-term over $V(\phi, \nu)$ and s, the *sign* of T, an element of $\{+, -\}$. We will represent $< T, s >$ as T^s. A

d-term with sign "+" (resp. "-") is called $positive$ (resp. $negative$) d-term. We say that (1) T^s *occurs* in a document d if T occurs in d, (2) T^s has *degree* k if T has degree k and (3) T_1^s and T_2^s are *independent* if T_1 and T_2 are so.

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. Now, the objective is that of determining a set of d-terms for c which best discriminate c from the other categories.

Definition 7.4 ***(eligible documents)*** Given a set of (independent) d-terms $X_c = \{T_1^+, \cdots, T_n^+, T_{n+1}^-, \cdots, T_{n+m}^-\}$ for c, we say that a document $d \in TS$ is *eligible* for classification under c according to X_c if *any* of the positive d-terms $T_1^+, \cdots T_n^+$ occurs in d and *none* of the negative d-terms $T_{n+1}^-, \cdots, T_{n+m}^-$ occurs in d. , i.e., if

$$(T_1 \in d \vee \cdots \vee T_n \in d) \wedge \ not\ (T_{n+1} \in d \vee \cdots \vee T_{n+m} \in d).$$

Notice that, albeit not necessary, the independence between d-terms is a desirable property of X_c (indeed, the presence of dependent d-terms like, for instance, t_1^+ and $(t_1 \wedge t_2)^+$, would be redundant in X_c, as $t_1 \in d \vee (t_1 \wedge t_2) \in d$ is clearly equivalent to $t_1 \in d$).

Example 7.5 (eligible documents) Assume that

$$X_c = \{(t_1 \wedge t_2)^+, t_3^+, (t_4 \wedge t_5)^-, t_6^-\}$$

is the set of d-terms associated with category c. Any document $d \in TS$ characterized by the *presence* of either $t_1 \wedge t_2$ or t_3 and the absence of both $t_4 \wedge t_5$ and t_6, is *eligible* for classification under c.

Following the above definition, the set of documents eligible for classification under c according to X_c is

$$E(X_c) = \bigcup_{T_i^+ \in X_c} \Delta(T_i^+) \setminus \bigcup_{T_j^- \in X_c} \Delta(T_j^-) \tag{7.4}$$

where $\Delta(T^s) \subseteq TS$ denotes the set of documents in TS where d-term T^s occurs. Now, if we classify all documents in $E(X_c)$ under c, the resulting Precision, Recall and F-measure are the following:

$$Pr(X_c) = \frac{|E(X_c) \cap TS_c|}{|E(X_c)|}$$

$$Re(X_c) = \frac{|E(X_c) \cap TS_c|}{|TS_c|}$$

$$\begin{aligned} F_\alpha(X_c) &= \frac{Pr(X_c) \cdot Re(X_c)}{(1-\alpha)Pr(X_c) + \alpha Re(X_c)} \\ &= \frac{|E(X_c) \cap TS_c|}{(1-\alpha)|TS_c| + \alpha|E(X_c)|}. \end{aligned} \tag{7.5}$$

We are now in a position to state the following optimization problem.

PROBLEM *DT-GEN (d-term generation)*. Given a vocabulary $V(\phi, \nu)$ and $\alpha \in [0 \cdots 1]$, find a set X_c of (independent) d-terms over $V(\phi, \nu)$ such that, by classifying under c the set $E(X_c)$ of the documents eligible for classification under c according to X_c, the resulting $F_\alpha(X_c)$ is maximum.

Intuitively, a solution of problem *DT-GEN* is a set of d-terms X_c such that $E(X_c)$ best "fits" the training set TS_c.
Next we show that *DT-GEN* is a computationally difficult problem. To this end, we need the following preliminary result.

Lemma 7.6 Let A be a set of independent co-terms over a given vocabulary $V(\phi, \nu)$. Then, the size of A has no polynomial bound on the size of $V(\phi, \nu)$.

Proof. We first observe that any two (different) co-terms T_1 and T_2 of degree k are independent (indeed, having the same degree, neither $T_1^{\{\}} \subseteq T_2^{\{\}}$ nor $T_2^{\{\}} \subseteq T_1^{\{\}}$ can happen). Now, it suffices to show that the number of co-terms of degree k over $V(\phi, \nu)$ is not polynomially bound on the number n of such terms. To this end, we can regard to a co-term $t_1 \wedge \cdots \wedge t_k$ of degree k as a combination $(t_1 \cdots t_k)$ of k terms, selected from n distinct terms in $V(\phi, \nu)$ without regard to the order in which they appear. Now, it is well known that the number of such combinations is $\frac{n!}{k!(n-k)!}$. □

Proposition 7.7 Problem *DT-GEN* requires exponential time in the size of the vocabulary $V(\phi, \nu)$.

Proof. Even restricting problem *DT-GEN* to admit solutions X_c consisting solely of independent d-terms, X_c has no polynomial bound on the size of $V(\phi, \nu)$ (this immediately follows from lemma 7.6). Thus, an exponential amount of time is needed to generate a solution. □

Next we show that *DT-GEN* remains difficult even if we restrict X_c to consists of d-terms of degree 1.

PROBLEM *1DT-GEN*. We are given a vocabulary $V(\phi, \nu)$ and $\alpha \in [0 \cdots 1]$. Then, find a set X_c of (independent) d-terms of *degree 1* over $V(\phi, \nu)$ such that, by classifying under c the set $E(X_c)$ of the documents eligible for classification under c according to X_c, the resulting $F_\alpha(X_c)$ is maximum.

Proposition 7.8 Problem *1DT-GEN* is NP-hard.

Proof. We next show that the decision version of problem *1DT-GEN* is NP-complete.
Let $\mathcal{DT} = \{\tau_1, ..., \tau_q\}$ be the set of all d-terms of degree 1 obtainable from $V(\phi, \nu)$. Now, problem *1DT-GEN*, in its recognition version, can be formulated as follows:

$-V1$: find a set of d-terms $X_c \subseteq \mathcal{DT}$ such that $F_\alpha(X_c) \geq K$

where K is a constant. Let us call *1DT-GEN*^{DV} this decision problem.
Membership. The set $\mathcal{DT}$ has cardinality $2 \cdot |V(\phi, \nu)|$, as it is made of all d-terms t^+, t^- such that $t \in V(\phi, \nu)$. Hence, any $X_c \subseteq \mathcal{DT}$ is polynomially bound on the size n of $V(\phi, \nu)$. We can therefore verify a YES answer of problem *1DT-GEN*^{DV}, using equation 7.5, in time polynomial in n.
Hardness. Given a generic set $X_c \subseteq \mathcal{DT}$, we denote by $S \subseteq \{1, \cdots, q\}$ the set of the indices of the elements of $\mathcal{DT}$ that are in X_c, and by $\Delta(\tau_i)$ the set of the documents of the training set where τ_i occurs. The set of documents eligible for classification under c according to X_c is $E(X_c) = E^+(X_c) \setminus E^-(X_c)$ (see equation 7.4), where $E^+(X_c) = \cup_{i \in S} \Delta(\tau_i)$ such that τ_i is a positive d-terms ($E^-(X_c)$ is defined accordingly). Further, we define a partition of $E(X_c)$ into the following subsets:

- $\Psi(X_c) = E(X_c) \cap TS_c$, i.e., the set of eligible documents that belong to the training set TS_c (true eligible);
- $\Omega(X_c) = E(X_c) \setminus TS_c$, i.e., the set of eligible documents that do not belong to TS_c (false eligible).

Now, it is straightforward to see that the F-measure $F_\alpha(X_c)$, given by equation 7.5, is proportional to the size $\psi(X_c)$ of $\Psi(X_c)$ and is inversely proportional to the

size $\omega(X_c)$ of $\Omega(X_c)$ (just replace in equation 7.5 the quantities $|E(X_c) \cap TS_c|$ by $\psi(X_c)$ and $|E(X_c)|$ by $\psi(X_c) + \omega(X_c)$). Therefore, problem *1DT-GEN*^{DV} can equivalently be stated as follows:

$-\ V2$: find $X_c \subseteq \mathcal{DT}$ such that $\psi(X_c) \geq V$ and $\omega(X_c) \leq C$

where V and C are constants. That is, to increase the F-measure as much as possible we have find a set X_c that makes $\psi(X_c)$ the largest possible, with $\omega(X_c)$ bound to some value C.
Now, let us restrict *1DT-GEN*^{DV} to the simpler case in which (1) d-terms are only positive (thus, no negative d-term occurs in $\mathcal{DT}$), and (2) (positive) d-terms are pairwise disjoint, i.e., $\Delta(\tau_i) \cap \Delta(\tau_j) = \emptyset$ for all $\tau_i, \tau_j \in \mathcal{DT}$ (let us call $\mathcal{DSJ}+$ this assumption). Next we show that, under $\mathcal{DSJ}+$, *1DT-GEN*^{DV} coincides with the Knapsack problem. To this end, we associate with each $\tau_i \in \mathcal{DT}$ two constants, v_i (the *value* of τ_i) and a c_i (the *cost* of τ_i) as follows:

$$v_i = |\Delta(\tau_i) \cap TS_c|$$

$$c_i = |\Delta(\tau_i) \setminus TS_c|.$$

That is, the value v_i (resp. cost c_i) of a (positive) d-term τ_i is the number of documents containing τ_i that are true eligible (resp. false eligible).
Now we prove that, under $\mathcal{DSJ}+$, the equality $\Sigma_{i \in S} v_i = \psi(X_c)$ holds, for any $X_c \subseteq \mathcal{DT}$. Indeed:

$$\Sigma_{i \in S} v_i = \Sigma_{i \in S} |\Delta(\tau_i) \cap TS_c| =$$
$$|\cup_{i \in S}\, \Delta(\tau_i) \cap TS_c| =$$
$$|(E^+(X_c) \cap TS_c| =$$
$$|E(X_c) \cap TS_c| = \psi(X_c).$$

To get the first line above we apply the definition of v_i. For the second, we exploit point 2 of assumption $\mathcal{DSJ}+$. For the third and the fourth we use the definitions of $E^+(X_c)$ and $E(X_c)$, respectively, (note that, according to point 1 of assumption $\mathcal{DSJ}+$, $E^-(X_c) = \emptyset$ holds).
In the same way as for v_i, the equality $\Sigma_{i \in S} c_i = \omega(X_c)$ can be easily shown.
Therefore, by replacing $\psi(X_c)$ and $\omega(X_c)$ in version $V2$ of our problem, we get the following formulation of problem *1DT-GEN*^{DV}, valid under $\mathcal{DSJ}+$:

$-\ V3$: find $X_c \subseteq \mathcal{DT}$ such that $\Sigma_{i \in S} v_i \geq V$ and $\Sigma_{i \in S} c_i \leq C$.

That is, under $\mathcal{DSJ}+$, *1DT-GEN*^{DV} is the Knapsack problem – a well known NP-complete problem. Therefore, *1DT-GEN*^{DV} under $\mathcal{DSJ}+$ is NP-complete. It turns out that (the general case of) *1DT-GEN*^{DV} (which is at least as complex as *1DT-GEN*^{DV} under $\mathcal{DSJ}+$) is NP-hard.
Having proved both membership (in NP) and hardness, we conclude that *1DT-GEN*^{DV} is NP-complete. From which the thesis follows. □

7.4 Classifier definition

We are now ready to define the *classifier* of c deriving from a solution X_c of problem *DT-GEN*.

Definition 7.9 (**Classifier**) Let the set $X_c = \{T_1^+, \cdots T_n^+, T_{n+1}^- , \cdots , T_{n+m}^-\}$ of d-terms over $V(\phi, \nu)$ be a solution of problem *DT-GEN*, for given values of ϕ, ν and α. Now, the *classifier* of c, given X_c, is the set of rules $CLASS_c(\phi, \nu, \alpha) = \{r_1, \cdots , r_n\}$, where r_i, $1 \leq i \leq n$, is:

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

We call $T_i \in d$ the *positive literal* of r_i , while each $T_{n+j} \notin d$ a *negative literal* of r_i .

Example 7.10 (Classifier) Let

$$X_c = \{(t_1 \wedge t_2)^+, t_3^+, (t_4 \wedge t_5)^-, t_6^-\}$$

be a best set of d-terms for category c, for given values of ϕ, ν and α. Then, according to Definition 7.9, $CLASS_c(\phi, \nu, \alpha)$ consists of the following rules:

$$c \leftarrow (t_1 \wedge t_2) \in d, (t_4 \wedge t_5) \notin d, t_6 \notin d,$$

$$c \leftarrow t_3 \in d, (t_4 \wedge t_5) \notin d, t_6 \notin d.$$

We note that there is one rule in $CLASS_c(\phi, \nu, \alpha)$ for each positive d-terms in X_c.

The meaning of a classification rule $r_i \in CLASS_c(\phi, \nu, \alpha)$ is the following: classify a document d under c if the conjunction $T_i \in d, T_{n+1} \notin d, \cdots , T_{n+m} \notin d$ is true w.r.t. d, i.e., if T_i occurs in d and none of $T_{n+1}, \cdots , T_{n+m}$ occurs in d. Thus, the meaning of $CLASS_c(\phi, \nu, \alpha)$ is: classify a document d under c if any T_i, $1 \leq i \leq n$, occurs in d and none of $T_{n+1}, \cdots , T_{n+m}$ occurs in d.
We note that $CLASS_c(\phi, \nu, \alpha)$ is not PAC-learnable [].

7.5 A Heuristics for dealing with problem DT-GEN

To deal with the complexity of problem *DT-GEN*, we devised the greedy heuristics sketched in Figure 7.1. The input consists of the vocabulary $V(\phi, \nu)$, along with the parameter α. The do-while-loop of lines 3-8 generates the sequence of sets of

Algorithm DT-GEN

Input: the vocabulary $V(\phi, \nu)$, for given values of ϕ and ν; a value of the Precision/Recall threshold α in F_α.
Output: a suboptimal set X_c of d-terms;
Method: perform the following steps:

p1. **function** Generate-New-Set(X_{i-1})
p2. $X_i = X_{i-1}$;
p3. **for each** $t \in V(\phi, \nu)$ **do**
p4. $termFound = false$;
p5. **for each** $T^s \in X_{i-1} \cup \{\epsilon^+, \epsilon^-\}$
p6. $\tau = (T \wedge t)^s$;
p7. $X = X_{i-1} \setminus \{T^s\} \cup \{\tau\}$;
p8. **if** $F_\alpha(X) > f\_opt$ **then**
p9. $f\_opt = F_\alpha(X)$; $t\_opt = t$; $termFound = true$;
$oldTerm = T^s$; $newTerm = \tau$;
p10. **end-for**;
p11. **if** $termFound$ **then** $X_i = X_{i-1} \setminus \{oldTerm\} \cup \{ newTerm\}$;
p12. **end-for**;
p13. **return** X_i.

1. **begin** {main}
2. $i = 0$; $f\_opt = 0$; $X_i = \emptyset$;
3. **do**
4. $i = i + 1$;
5. $X_i =$ Generate-New-Set(X_{i-1});
6. **if** $X_i \neq X_{i-1}$
7. $V(\phi, \nu) = V(\phi, \nu) \setminus \{t\_opt\}$;
8. **while** $(V(\phi, \nu) \neq \emptyset \ \wedge \ X_i \neq X_{i-1})$;
9. $X_c = X_i$;
10. **return** X_c.

Figure 7.1: Algorithm *DT-GEN*

d-terms

$$X_0 = \emptyset \, , \cdots , \; X_i = GNS(X_{i-1}), \; \cdots , X_m = GNS(X_{m-1})$$

by invoking, at each round, the function *Generate-New-Set* (lines p1-p12). The i-th call to this procedure "extends" the current solution X_{i-i} by picking (lines p3-p12) the term $t \in V(\phi, \nu)$ (if any) which, conjuncted to a d-term $T^s \in X_{i-1}$ (line p6), yields the greatest increase of the objective function F_α (notice that the current value of F_α is stored into f_opt, a global variable which is modified by *Generate-New-Set* at line p9). The symbol ϵ at line p5 represents the logical constant *true* (taken both positively and negatively). This formal device is needed in order for the algorithm to capture also d-terms of degree 1. The evaluation of F_α is based on equation 7.5 above. Once the best term t has been selected, it is returned to the main program through the global variable t_opt (which is modified at line p9) and then removed from $V(\phi, \nu)$ (line 7). The do-while-loop iterates as long as the vocabulary $V(\phi, \nu)$ is not empty and a new d-term is generated.
It is straightforward to realize that the size of X_c (i.e., the number of generated d-terms) is less or equal to the number m of calls to *Generate-New-Set* (in particular, $|X_c| = m$ if all d-terms have degree 1 and, in general, $|X_c|$ is equal to m divided by the average degree of the d-terms in X_c). Further, each X_i ($1 \leq i \leq m$) (1) contains at least one positive d-term (the first generated in X_1) and (2) consists of independent d-terms; more precisely, for each $T_1^s, T_2^s \in X_i$, the condition $T_1^{\{\}} \cap T_2^{\{\}} = \emptyset$ holds, i.e., T_1 and T_2 have no term in common. This is because, by discarding at line 7 the selected term t_opt from the current vocabulary $V(\phi, \nu)$, t_opt cannot occur in two different d-terms.

Proposition 7.11 Algorithm *DT-GEN* computes a suboptimal set X_c of d-terms in time $\mathcal{O}(n^3 p)$, where $n = |V(\phi, \nu)|$ and $p = |TS|$.

Proof. Preliminarily we observe that both $E(X_c)$ and TS_c are bound by p, where $p = |TS|$ is the number of training documents, so that the evaluation of $F_\alpha(X_c)$ by equation 7.5 requires time $\mathcal{O}(p)$. Further, we note that $n << p$ holds.
Now let us analyze algorithm *DT-GEN*. The do-while-loop of lines 3-8 iterates (at most) up to $V(\phi, \nu) = \emptyset$; since at each step one term is removed from $V(\phi, \nu)$ (line 7), the while-loop takes time $\mathcal{O}(n)$, where $n = |V(\phi, \nu|$. At each round of this loop, the procedure *Generate-New-Set* is invoked. For each element t in the current $V(\phi, \nu)$, *Generate-New-Set* tries to conjunct t with each co-term of the current set X_{i-1} (lines p5-p10). Since each term t may occur within (at most) one co-term of X_{i-1}, the size of X_{i-1} is bound by the size of $V(\phi, \nu)$, i.e., $|X_{i-1}| \in \mathcal{O}(n)$. Thus, function F_α at line p8 is evaluated (by equation 7.5) $\mathcal{O}(n^2)$ times, each time requiring $\mathcal{O}(p)$ time. It turns out that *Generate-New-Set* takes time $\mathcal{O}(n^2 p)$ and, hence, algorithm *DT-GEN* takes time $\mathcal{O}(n^3 p)$. □

Remark. Note that, in practice, the efficiency of algorithm *DT-GEN* strongly depends on the number $|X_c|$ of d-terms that are generated; in fact, both the do-while-loop of lines 3-8 and the inner loop of procedure *Generate-New-Set* depends on $|X_c|$ (whose upper bound is the size n of $V(\phi, \nu)$). In particular, as we have seen, the do-while-loop iterates m times, with m given by the size of X_c times the average degree of the d-terms in X_c, while the inner loop of *Generate-New-Set* iterates at most $|X_c|$ times. Thus, we may say that Algorithm *DT-GEN* computes a suboptimal set X_c of d-terms in time $\mathcal{O}(|X_c|^2 \cdot n \cdot p)$. It turns out that, whenever $|X_c|$ is negligible w.r.t. n (as it is usually the case), *DT-GEN* tends to be $\mathcal{O}(n^k p)$, with $k \leq 2$.

7.6 The Learning Process

In the previous sections we have seen how the classifier $CLASS_c(\phi, \nu, \alpha)$ of c is learned, for given values of the model parameters ϕ, ν and α. To induce the "best" classifier $CLASS_c$ of c, essentially Olex repeatedly runs Algorithm *DT-GEN* with different values of the model parameters, as shown in Figure 7.2. Here, the input consists of the set Φ of scoring function symbols and a set $\mathcal{N}$ of nonnegative integers, along with the local vocabulary of each category $c \in \mathcal{C}$ (whereby creating the vocabulary $V(\phi, \nu)$). Further, a value of the parameter α is provided (depending on the needs of the application at hand; usually, $\alpha = 0.5$).
As we can see, the learning process is a simple spanning of the search space defined by Φ and $\mathcal{N}$; indeed, the steps described in the previous sections (see lines 4-7) are iterated for each $\phi \in \Phi$ and each $\nu \in \mathcal{N}$. Whenever a classifier $CLASS_c(\phi, \nu, \alpha)$ is generated, it is validated over the validation set (line 9). Finally, the best classifier (i.e., the one with the maximum value of the F-measure) is output (line 12).
Notice that, of the three model parameters ϕ, ν and α, only the former two are actually used for "driving" the search of $CLASS_c$ (for this reason we will often refer to them as "tuning parameters").
Next we provide a time complexity analysis of the learning process.

Proposition 7.12 The time complexity of the learning process of Figure 7.2 is $\mathcal{O}(n^3 p)$, where $n = max\{\nu : \nu \in \mathcal{N}\} \cdot |\mathcal{C}|$ is the maximal input vocabulary size and $p = |TS|$ is the training set size.

Proof. By neglecting issues related to vocabulary generation and classifier construction, the time required to generate $CLASS(\phi, \nu, \alpha)$ (for given values of ϕ, ν

Learning Process for a category c

Input: the set Φ of function symbols; a set $\mathcal{N}$ of nonnegative integers; a value for the Precision/Recall threshold α; the local vocabulary V_c, for each $c \in \mathcal{C}$.
Output: the "best" classifier $CLASS_c$ of c;
Method: perform the following steps:

1. **begin**
2. **for each** $\phi \in \Phi$ **do**
3. **for each** $\nu \in \mathcal{N}$ **do**
4. – create the vocabulary $V(\phi, \nu) = \cup_{c \in \mathcal{C}} V_c(\phi, \nu)$;
5. – % INDUCTION
6. – run Algorithm *DT-GEN* passing $V(\phi, \nu)$ and α;
7. – construct the classifier $CLASS_c(\phi, \nu, \alpha)$ for c;
8. – % VALIDATION
9. – validate $CLASS_c(\phi, \nu, \alpha)$ over the validation set and generate the F-measure for c;
10. **end-for**;
11. **end-for**;
12. **output** the best classifier generated for category c;
13. **end**.

Figure 7.2: Learning Process for a category c

and α - see lines 4-7) is the one required by algorithm *DT-GEN*, i.e., $\mathcal{O}(n^3 p)$ (see Proposition 7.11), where $n = \nu \cdot |\mathcal{C}|$ is the size of the current vocabulary $V(\phi, \nu)$ (see line 4). In turn, the validation phase (line 9), i.e., the execution of $CLASS_c(\phi, \nu, \alpha)$ over the test set, is done in time $\mathcal{O}(q)$, where q is the number of terms occurring in the documents of the test set (here, we reasonably assume that the dimension of $CLASS_c(\phi, \nu, \alpha)$ is negligible w.r.t. to q). Hence, the time needed for lines 4-9 is $\mathcal{O}(n^3 p + q) \in \mathcal{O}(n^3 p)$ (q is indeed negligible w.r.t. $n^3 p$). Now, lines 4-9 are iterated $|\mathcal{N}|$ times with different values of ν (see line 3). Since $|\mathcal{N}| << n$, where $n = max\{\nu : \nu \in \mathcal{N}\} \cdot |\mathcal{C}|$ (i.e., $|\mathcal{N}|$ is negligible w.r.t. the maximal vocabulary size), the time required to execute the **for**-loop of line 3 is $\mathcal{O}(n^3 p)$. Finally, the **for**-loop of line 3 is iterated $|\Phi|$ times (see line 2), for different scoring functions. Again, $|\Phi|$ is negligible w.r.t. n, so that the overall learning time for a category is $\mathcal{O}(n^3 p)$, with $n = max\{\nu : \nu \in \mathcal{N}\} \cdot |\mathcal{C}|$. □

Chapter 8

Benchmark Experimentation and Comparison

To provide an objective basis for comparisons of Olex with other approaches, we have experimentally evaluated our algorithm using two standard benchmark corpora: the REUTERS-21578 [55] and the OHSUMED [40] collections, already described in section 3.
In this chapter, after having described the experimental phase and defined the applied methodology, we show the results obtained on both benchmark corpora. These are eventually compared with the results of other learning approaches like SVM, K-NN, Rocchio, etc.

8.1 Benchmark Corpora

The fiest data set we used is REUTERS-21578. This corpus consists of 21,578 news stories appeared on the Reuters newswire in 1987. The Reuters-21578 documents actually used in text classification experiments are only 12,902, since the other 8,676 documents have not been manually assigned to categories. The TOPICS group of categories contains 135 categories. In our experiments, we considered the subset of the 90 categories with at least one positive training example and one test example (we will refer to it as **R90**).
The second data set we considered is OHSUMED, in particular, the collection consisting of the first 20,000 documents from the 50,216 medical abstracts of the year 1991. The task considered was to assign those documents to one or more categories of the 23 MeSH *diseases*.

8.2 Document Pre-processing

Preliminarily, documents were subjected to the following pre-processing steps:

- First, we removed all words occurring in a list of common stopwords, as well as punctuation marks and numbers.
- Then, we extracted all n-grams, defined as sequences of maximum three words consecutively occurring within a document (after stopword removal).
- At this point we have partitioned each corpus into a training corpus (*seen data*), to build the model, and a test set (*unseen data*), to measure performance of the induced model. In particular:
  - REUTERS-21578 : we have applied the ModApté split in which 9603 documents are used to form the training corpus and 3299 to form the test set.
  - OHSUMED : of the 20,000 documents of the corpus, the first 10,000 were used for training and the second 10,000 for testing.

  In both cases we further split the training corpus into a *training set* (70% of the initial training corpus), on which to carry out the induction process, and a *validation set*, on which tuning the model parameters. We performed the split in such a way that each category was proportionally represented in both sets (stratified holdout).
- Finally, we created the local vocabulary V_c of each category c (see Section 7.3) by scoring all n-grams occurring in the documents of the training set TS_c of c. To this end, we used three functions, namely, Information Gain (IG), Chi Square (CHI) and Odds Ratio (OR), defined as follows [15, 90]:

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

Table 8.1: Term scoring functions used for vocabulary reduction.

where A, B, C and D, according to the two-way contingency table of a term t and a category c (shown in table 2.1). Here, we recall their meaning: A is the number of documents in 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. Further, N is the total number of documents. All the above functions are frequently used to assess term-goodness in the area of machine learning. Function CHI measures the lack of independence between a term t and a category c; its value is zero if t and c are independent. In turn, IG measures the increase of information obtained for category prediction by knowing that term t is present or absent in a document. Yang and Pederson [90] reported IG and CHI performed best in their benchmarks. Function OR measures the odds of term t occurring in the positive class c normalized by that of the negative class. It has been used by Mladenic for selecting terms in text categorization [49].

8.3 Experiments

We have conducted a number of experiments for the induction of the best classifiers on both corpora according to the learning process described in Section 7.6 (an experiment is the sequence of steps 4-9 of Figure 7.2). To this end, we used the set $\Phi = \{CHI, IG, OR\}$ of scoring functions.
Once selected the "best" classifier for a category (i.e., the one with the maximum value of the F-measure on the *validation set*), we have measured its performance on the *test set* (unseen data).
We have further investigated the sensitivity of the system to model parameters.
We remark that, concerning the REUTERS-21578 corpus, we used *all* 9603 documents of the training set for the training phase, and performed the test using *all* 3299 documents of the test set (including those not belonging to any category in **R90**).
We carried out all the experiments by using the value 0.5 for the Precision/Recall threshold α (which attributes equal importance to Precision and Recall).

8.4 Performance Metrics

Classification effectiveness was measured in terms of the classical notions of Precision, Recall and F-measure, as defined in Section 7.2. To obtain global estimates relative to experiments performed over a set of categories, the standard definition of micro-averaged Precision and Recall was used:

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

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

Based on the above formulas, both the micro-averaged F-measure and break-even point (BEP) were consistently calculated (the latter as the arithmetic average of micro-averaged Precision and Recall).

8.5 Results with Reuters

The first data set we considered is the REUTERS-21578 and the task was to assign documents to one or more categories of **R90**.

Best Classifiers

Table 8.2 reports the best performance obtained for the ten most frequent categories in **R90** (in the following we will refer to them as **R10**), and the micro-averaged performance over all **R90** categories.

The columns labeled "Model Parameters" show the values of ϕ and ν yielding the best classifiers. The columns named "Training" show the size of the training set of each category along with the F-measure predicted by algorithm *DT-GEN* (corresponding to the F-measure evaluated over the training set). The column labeled "Validation" (rep. "Test") reports the size of the validation (resp. test) set of each category and the respective F-measure and Break Even Point (BEP) values.

Finally, table 8.3 reports the characteristics of the best classifiers for **R10**, namely, the number of rules and the number of negative literals occurring in each rule of a classifier (notice that the sum of those two values is equal to the number of the induced d-terms).

The results shown in Table 8.2 and 8.3 indicate that Olex is sensitive to both ϕ

Category	**Model Param.**		**Training**		**Validation**			**Test**		
name	ϕ	ν	#D	F	#D	F	BEP	#D	F	BEP
earn	CHI	70	2014	93.73	863	92.61	92.62	1087	95.74	95.74
acq	CHI	80	1155	87.69	495	82.64	82.64	719	88.78	88.78
money-fx	CHI	70	377	87.52	161	78.64	78.64	179	70.45	70.48
grain	CHI	5	303	95.27	130	91.25	91.27	149	92.62	92.62
crude	OR	40	272	90.77	117	77.57	78.25	189	82.41	83.07
trade	IG	10	258	84.95	111	66.91	69.49	117	66.67	66.91
interest	CHI	60	243	79.65	104	72.64	72.67	131	62.76	63.35
wheat	OR	20	148	94.93	64	89.71	90.02	71	90.91	91.46
ship	OR	70	138	91.47	59	77.06	77.59	89	78.26	79.14
corn	IG	30	127	94.86	54	94.64	94.76	50	93.22	93.46
μF (R10)				90.08		82.37	82.79		82.18	82.50
μF (R90)						**83.78**	**83.78**		**85.36**	**85.36**

Table 8.2: Best performance for categories in **R10** and micro-averaged performance over **R90**.

Category	**Classifier**	
	#rules	# neg. lit.
earn	30	14
acq	34	34
money-fx	21	32
grain	9	7
crude	34	19
trade	5	24
interest	17	10
wheat	3	0
ship	20	3
corn	2	6
avg	**17.5**	**14.9**

Table 8.3: Classifiers for REUTERS-21578 top ten categories.

and μ. As we can see, each category has indeed its own optimal parameter values, which means that the performance of Olex substantially depend on appropriate parameter tuning. More specifically, these results can be summarized as follows:

- Olex achieves micro-averaged F-measure and BEP over the *test set* (unseen data) both equal to 85.36; as we will see in the next section, this is a very

remarkable performance result;

- by comparing the above value with the F-measure=BEP=83.78 obtained on the *validation set*, it comes out an excellent generalization capability; also the predicted F-measure is quite reliable - the difference between predicted and test values being on average around 12%. All this shows that the the induced classifiers only slightly *overfit* the training data[1];

- the size of a classifier ranges between a minimum of 2 rules (for category "corn") to a maximum of 34 rules (for category "acquisition"), with 17.5 rules per classifier on average (thus, the induced classification models are very compact);

- the best classifiers for the categories in **R10** are all induced starting from vocabularies $V(\phi, \nu)$ with ν ranging between 5 and 80 (terms/category); that is, Olex can learn from rather small vocabularies (450-7200 terms). Further, the number of the induced d-terms for a category is negligible w.r.t. the vocabulary size (the maximum number of d-terms is 68 for "money-fx" - 34 positive and 34 negative, while the average is 32.4 per category); the combination of these two factors entails a great benefit in terms of efficiency, as confirmed by the run time results reported in Table 8.11 (see also the remark of Section 7.5).

Effect of Category Size on Performance

In Table 8.4 we summarize the accuracy, in terms of BEP, of the best classifiers of the categories in **R90** with at least 5 positive examples in the training set (there are 63 such categories). As we can see, there is a broad range of category sizes (from 5 documents up to 2015), that we have partitioned into four subranges (see column 1). Column 2 gives the number of categories falling in each subrange, while column 3 reports the respective average BEP. The experimental results, as shown in Table 8.4, indicate that performance are substantially constant on the various subsets, i.e., there is no correlation between category size and accuracy measures (this is not the case of other machine learning techniques, e.g., decision tree induction classifiers, which are biased towards frequent classes [42]).

[1]Overfitting is a phenomenon which makes a classifier very good at classifying the training data, but poor at classifying other data [74]

cat size (#docs)	#cat	avg BEP
≥ 127 and < 2015	10	82.50
≥ 50 and < 127	12	80.00
≥ 20 and < 50	16	82.76
≥ 5 and < 20	15	78.86

Table 8.4: Effect of category size on performance

Sensitivity to Model Parameter Settings

Table 8.5 summarizes the micro-averaged F-measure over **R90** obtained by using different values of the tuning parameters ϕ and ν. Notably, we tested all scoring functions with ν varying from 5 up to 200 (terms per category). As we can see, the three functions perform likewise; indeed, the (micro-averaged) performance of the induced classifiers in all cases achieve a maximum value (highlighted in bold), thus starting decreasing. That is, a reduction of the vocabulary size provides a benefit in terms of performance (as noted by several authors - e.g. [90]).

We have also analyzed the tradeoff between Precision and Recall as a function

ν	$\phi = CHI$	$\phi = IG$	$\phi = OR$
5	75.78	74.69	55.91
10	77.34	76,47	61.51
20	79.13	77.86	66.97
40	79.93	79.62	71.34
60	80.59	80.29	74.61
80	80.63	80.36	75.04
100	**81.14**	**81.11**	76.39
150	80.74	81.03	**78.46**
200	80.02	80.04	78.06

Table 8.5: Effect of varying ϕ and ν on the micro-averaged F-measure over R90.

of the parameter α in the objective function F_α. As already mentioned, α can be adjusted to increase the Precision (at the cost of a decreased Recall), or the Recall (at the detriment of Precision). Figure 8.1 shows a Precision/Recall curve for the categories in **R90**, drawn by micro-averaging over all 90 categories the values of Precision and Recall obtained (on the validation set) by setting $\phi = CHI$ and $\nu = 20$ and letting α vary from 0 to 1. This curve exhibits a "good" shape, showing the capability of the Olex classifiers to support high levels of one of the

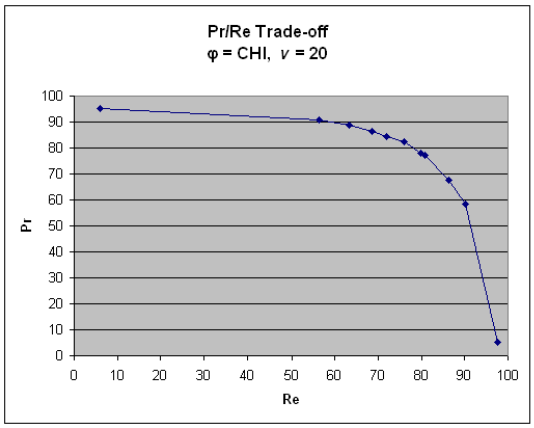

Figure 8.1: Micro-averaged Precision/Recall curve over **R90**

two performance parameters, without paying to much in terms of the other one. Finally, it may worth noticing the equality of micro-averaged F-measure BEP in the experiments of Table 8.2, meaning that Precision and Recall are perfectly balance. This according to the precision-recall threshold α set to 0.5.

Effect of Conjunctions

To investigate the effect of conjunctive literals on the Olex performance we induced the best classifiers consisting of just simple rules (i.e., rules where no conjunctive literals were allowed). Then, we compared their performance with those reported in Table 8.2. Results are summarized in Table 8.6. Here we reported the best performance for **R10** and micro-averaged performance for **R90** obtained over the test set by using simple rules and conjunctive rules separately. As we can see (1) the performance of each classifier in **R10** in case of simple rules is less than that obtained by using conjunctive rules in nine out then categories (only for "wheat" they tie), and (2) the micro-averaged BEP=80.47 over **R90** in case of simple rules is well under the BEP=85.36 obtained by conjunctive rules (thus, the rise in performance due to conjunctive terms is around 6%).

We also analyzed the effect of conjunctions on the relationship between category frequency and performance. This can be easily seen by comparing the results reported in Table 8.4 with those in Table 8.7, the latter obtained by using simple rules. As we can see, Table 8.7 shows a quicker degrade of performance for rare categories.

	Simple Rules		**Conj. Rules**	
Cat.	F-meas	BEP	F-meas	BEP
earn	93.00	93.13	95.74	95.74
acq	83.51	84.32	88.78	88.78
money-fx	67.48	68.01	70.45	70.48
grain	91.28	91.28	92.62	92.62
crude	80.71	80.84	82.41	83.07
trade	64.23	64.28	66.67	66.91
interest	51.23	55.96	62.76	63.35
wheat	90.91	91.46	90.91	91.46
ship	77.50	78.49	78.26	79.14
corn	89.08	89.38	93.22	93.46
avg (R10)	78.89	79.71	82.18	82.50
μF (R90)	**80.44**	**80.47**	**85.36**	**85.36**

Table 8.6: Performance comparison between simple and conjunctive rules on **R10** and **R90**

cat size (#docs)	#cat	avg BEP
≥ 127 and < 2015	10	79.73
≥ 50 and < 127	12	70.02
≥ 20 and < 50	16	74.33
≥ 5 and < 20	15	62.20

Table 8.7: Effect of category size on performance in case of simple rules

8.6 Results with Ohsumed

The second data set we considered is OHSUMED and the task was to assign documents to one or more categories of the 23 MeSH *diseases*.

In Table 8.8 we provide the best performance for the ten most frequent categories and micro-averaged performance over all 23. As we can see, the micro-averaged F-measure and BEP are 61.52 and 61.57, respectively, over the test set, around 3% less than the corresponding values over the validation set (63.16 and 63.19, respectively). Again, we notice the good balancing of Precision and Recall. However, unlike in the case of the REUTERS-21578, the predicted values for the F-measure are on average around 18% larger than the test results, which is indicative of a certain overfitting over the training data.

Other relevant aspects coming out from Tables 8.8 and 8.9 and confirming the

behavior of Olex are: first, the ability to learn from small vocabularies (ν indeed ranges between 30 and 100 terms/category); second, the capability to induce small classifiers (their size ranges between 17 rules, for category C08, and 103 rules, for category C23 – on average around 36 rules/category). Also the absence of a correlation between training set size and performance was observed.
Finally, by comparing the above performance with the ones obtained by using simple rules (i.e., by limiting Olex to induce only co-terms of degree 1), we again observe a large improvement of the micro-averaged F-measure (61.52 against 59.35, around 4% more - see Table 8.10).

Category	**Model Param.**		**Training**		**Validation**			**Test**		
name	ϕ	ν	#D	F	#D	F	BEP	#D	F	BEP
C23	CHI	60	1259	65.79	540	48.60	48.64	1087	48.09	48.12
C14	IG	60	874	83.40	375	77.79	77.80	719	75.46	75.48
C04	CHI	30	814	81.29	349	77.44	77.5	179	78.70	78.74
C10	CHI	60	435	71.13	186	54.46	55.05	149	55.11	55.29
C06	CHI	70	412	77.35	176	68.32	68.92	189	65.51	66.18
C21	OR	100	382	78.24	164	65.81	65.97	117	60.32	60.47
C20	CHI	80	368	84.22	157	73.03	73.11	131	71.96	71.97
C12	OR	100	344	84.97	147	69.63	70.18	71	71.23	71.28
C08	CHI	50	331	75.50	142	61.45	62.76	89	64.98	65.75
C01	CHI	60	296	79.34	127	59.57	59.97	50	57.88	58.39
μF (top 10)				78.12		65.61	65.99		64.92	65.17
μF (all 23)						**63.16**	**63.19**		**61.52**	**61.57**

Table 8.8: Best classifiers for the ten most frequent MeSH "diseases" categories of Ohsumed and micro-averaged performance over all 23.

8.7 Time Efficiency

Next we report the execution times of the various steps for the REUTERS-21578 experimentation.
The pre-processing phase (stopwords and numbers removal, n-gram extraction, training set creation and local vocabularies creation) took around ??? seconds.
The run times of algorithm *DT-GEN* on **R10** and **R90**, for input vocabularies $V(\phi, \nu)$ with $\phi = CHI$ and values of ν ranging from 5 to 100, are reported in Table 8.11. These results show that Algorithm *DT-GEN* is substantially quadratic

Category	**Classifier**	
	#rules	# neg. lit.
C23	103	86
C14	27	35
C04	31	13
C10	30	42
C06	37	29
C21	25	10
C20	18	10
C12	45	9
C08	17	21
C01	30	22
avg	**36.3**	**27.7**

Table 8.9: Classifiers on OHSUMED top ten categories.

	Simple Rules		**Conj. Rules**	
Cat.	F-meas	BEP	F-meas	BEP
C23	47.00	47.32	48.09	48.12
C14	73.67	74.52	75.46	75.48
C04	77.63	77.78	78.70	78.74
C10	54.39	54.72	55.11	55.29
C06	63.17	63.25	65.51	66.18
C21	60.98	61.62	60.32	60.47
C20	67.58	67.81	71.96	71.97
C12	67.80	67.82	71.23	71.28
C08	61.43	61.57	64.98	65.75
C01	53.71	55.59	57.88	58.39
avg (top 10)	62.73	63.20	64.92	65.17
μF (all 23)	**59.35**	**59.38**	**61.52**	**61.57**

Table 8.10: Performance comparison between simple and conjunctive rules on Ohsumed

(in the vocabulary size) for some classes, linear for some others. As remarked in Section 7.5, this essentially stems from the relatively low number of d-terms w.r.t. the vocabulary size (see comments on experimental results in Subsection 8.5).

Finally, the execution time of the best classifiers over the test set took 65 seconds. All experiments were conducted on a 2 GHz Pentium 2 Gb RAM.

Category	**5**	**10**	**20**	**40**	**60**	**80**	**100**
earn	1	3	6	12	33	65	132
acq	1	3	7	21	45	74	145
money-fx	1	4	10	28	51	58	100
grain	1	2	5	10	17	29	38
crude	1	2	5	18	35	54	80
trade	1	5	11	28	54	75	92
interest	1	1	3	7	14	15	30
wheat	1	1	1	2	4	5	4
ship	1	1	2	6	9	10	14
corn	1	1	2	2	4	6	8
tot secs (R10)	10	23	52	133	266	349	643
tot secs (R90)	11	28	63	170	327	420	733

Table 8.11: *DT-GEN* run times (in secs) over **R10** and **R90** with input vocabulary $V(\phi, \nu)$, with $\phi = CHI$ and $5 \leq \nu \leq 100$

8.8 Performance Comparison

Although many results on both the REUTERS-21578 and the OHSUMED data collections are available in the literature (e.g. [9,45]), a direct comparison with our results is impossible, as different training sets and/or experimental settings (e.g., different methods to pre-process and partition the data) have been used. Thus, we used the Weka library of machine learning algorithms, version 3.5.6 [85], to compare Olex to five learning algorithms: SVM, Ripper, k-NN, C4.5 and Bayes. The documents of the training set were preliminarily pre-processed as described in subsection 8.2.

8.8.1 Reuters

All five methods were run after selecting the 500 best, 1000 best, 2000 best, 5000 best, (10000 best,) or all features (see section ??). As in [43], at each number of features, the values $k \in \{1, 15, 30, 45, 60\}$ for the k-NN classifier were tried. The best performance obtained on the test set are reported in Table 8.13. As we can see, on average, Olex (85.3) greatly surpasses Bayes (72.0), Rocchio (79.9), C4.5 (79.4) and k-NN (82.3), but performs slightly worse than both the SVM approaches (86.0 the polynomial SVM, and 86.4 the rbf SVM).

Table 8.13 contrasts the results on **R10** of Olex with those of two well known rule-

Classifier	size	Bayes	Rocchio	C4.5	k-NN	SVM(poly)	SVM(rbf)	Olex
earn	2877	95.9	96.1	96.1	97.3	98.5	98.5	95.74
acq	165	91.5	92.1	85.3	92.0	95.2	95.4	88.78
money	538	62.9	67.6	69.4	78.2	76.2	76.3	70.48
grain	433	72.5	79.5	89.1	82.2	92.4	93.1	92.62
crude	389	81.0	81.5	75.5	85.7	88.9	88.9	83.07
trade	369	50.0	77.4	59.2	77.4	77.1	77.8	66.91
interest	347	58.0	72.5	49.1	74.0	76.2	76.2	63.35
wheat	212	78.7	83.1	80.9	79.2	86.5	85.4	91.46
ship	197	60.6	79.4	85.5	76.6	85.9	85.2	79.14
corn	181	47.3	62.2	87.7	77.9	85.7	85.1	93.46
μF (R90)		72.0	79.9	79.4	82.3	86.0	86.4	85.36

Table 8.12: Recall/Precision breakeven points on R10 and R90

based learning techniques, namely, RIPPER and TRIPPER (their performance values are reported as given in [81]). The results show that Olex outperforms (in terms of BEP) RIPPER in a vast majority of categories (nine out of ten) and both RIPPER and TRIPPER in eight out of ten categories.

Finally, in Table 8.14 we compare Olex with the NeW associative algorithm [10];

Classifier	**earn**	**acq**	**money**	**grain**	**crude**	**trade**	**interest**	**wheat**	**ship**	**corn**
Tripper	95.1	86.3	70.4	87.9	82.5	58.9	71.5	84.5	80.9	85.7
Ripper	94.0	85.3	65.3	90.6	79.3	68.3	58.7	83.0	73.0	83.9
Olex	**96.3**	**89.2**	**70.9**	**92.6**	**85.0**	64.7	62.8	**91.5**	**81.3**	**92.1**

Table 8.13: Recall/Precision breakeven points on R10

in particular, we consider the break even performance along with the characteristics of the induced classifiers (number of rules). As we can see, besides giving BEP values on average higher than those provided by NeW (85.3 against 82.7), Olex generates classifiers that are much more compact than those generated by NeW (on average, 11 rules per classifier on R90, against 471 of NeW).

8.8.2 Ohsumed

In [43] Joachims categorizes the same test collection we used in our experimentation to compare the performance of SVMs with four classifiers. Again, Olex (BEP = 63.3) performs better than Naive Bayes (57.0), Rocchio (56.6), k-NN (59.1) and

Name	NeW #Rules	Olex #Rules	NeW BEP	Olex BEP
earn	1364	13	96.5	96,3
acq	538	34	88.7	89,2
money-fx	541	17	70.4	70.9
grain	223	9	86.6	92,6
crude	446	14	85.7	85.0
trade	574	5	76.1	64,7
interest	461	8	69.5	62,8
wheat	202	4	87.3	91.5
ship	177	4	81.2	81.8
corn	79	2	85.7	92,1
avg #rules (R10)	471.0	11.0		
μF (R90)			82.7	85.3

Table 8.14: Comparison with the association rule classifier NeW on R10 and R90

C4.5 (50.0), while is less effective than SVMs (65.9 the polynomial SVM and 66.0 the rbf SVM).

Chapter 9

Experimentation on real use-case corpora

A set of experiments has been carried out on two real use-case corpora, which were provided to us by an American company, the *Full Capture Solutions, Inc* (FCSI). Both data sets contain documents, concerning insurance agencies; these are called *FROI* (First Report of Injury) and represent reports about road accidents written down by policemen. Insurance agencies are interested in finding, among the whole set of FROI, the so called *"subrogation"* cases. A small percentage of the whole set of accidents, in fact, is caused by third parties with respect to the insured ones which have to recover a part or the whole damage. In these situations, the insurance agency, after having recovered the damage to the insured parties, may claim for damages the actual responsible one. Up to now, the search for "subrogation" cases has been carried out by manually analyzing all reports. This type of investigation obviously requires a lot of human effort. Furthermore, rapidity is an essential factor, since proofs of third party responsibility may suddenly disappear. For these reasons, FCSI is interested in using automatically induced classifiers to make search faster.
In the following sections, we will show the results obtained by Olex on both data sets.

9.1 Data sets

FCSI prepared two distinct and independent collections of documents:

FSCI_6024 which contains 6024 documents.

FSCI_2984 which contains 2984 documents.

Each of them is a collection of FROI, manually classified under two categories: "subrogation_yes" (in the following SUBRO_YES) and "subrogration_no" (SUBRO_NO). The distribution of the documents on FCSI_6024 and FCSI_2984 training corpora is shown in Tables 9.1 and 9.2.

Category	Documents
SUBRO_YES	1625
SUBRO_NO	4399

Table 9.1: Documents distribution with respect to the categories on FSCI_6024 data set

Category	Documents
SUBRO_YES	1779
SUBRO_NO	1205

Table 9.2: Documents distribution with respect to the categories on FSCI_2984 data set

9.2 Document Pre-processing

Linguistic Pre-processing

The documents belonging to FCSI corpora are, for their nature, linguistically different from the ones of the training corpora analyzed in chapter 8. The reports contain some information about the interested parties and a more or less detailed description about the accident; the latter, in particular, is written in the natural language and often contains a lot of expressions belonging to the local slang. Periods are short and concisely describe the situation of the accident ("Fainted after inhaling methane", "Car was rear-ended by another vehicle at a stoplight", "Poked his finger in his eye", "Cut finger while chopping cabbage", "Foreman putting air in tire when rim sprung up and hit him in the head causing a bone fracture"). Frequent are words contractions and abbreviations.
The following is an example of document belonging to FCSI_2984 training corpus:

```
tree fell on iv and camper INSD WAS CAMPING AND
A STORM CAME THROUGH AND LIGHTNING HIT TREE AND
THE coverages loss st-va contri neg 1% bar
eff 012402-012403 veh 02 1994 wildress camp veh
03 2000 f250 cmp 50 rr 20/600 priors 0 Rcvd both
est for iv, cv fwd to adj for pymt...kfc
```

As it can be easily noticed, it is impossible to remove any word or expression from a document like this, because this could imply a loss of information. For this reason, FCSI documents where pre-processed without performing any ***Stop word removal*** and ***Stemming*** step; while n-grams with length $n \leq 3$ were extracted according to the definition provided in section 7.2.

Semantic Pre-processing

In order to exploit the available external knowledge provided by the domain specific thesaurus, a semantic pre-processing step has been performed. In general, this step requires:

- a $n-$gram representation of documents. One can decide whether to use the available linguistic pre-processing or do it again, choosing a different type of $n-$gram (for instance, long $n-$gram are often useful only to extract concepts).
- a lexicon for the thesaurus, providing the correspondences between *terms* in the documents and *concepts* in the thesaurus. This is given through a set of binary associations between $n-$grams and concepts.

After semantic pre-processing, each document in the training corpus is represented as the collection of concepts "discovered" in it.

For FCSI training corpora, we used a thesaurus specially defined by FCSI knowledge engineers to represent the key concepts in the classification of FROI reports. The FCSI thesaurus has a very simple structure: it is a collection of concepts with their representative terms. More in details, each concept is related by a binary relation to a set of terms; the presence of a such term in a document provide evidence towards the fact that the document "deals with" a particular concept. There are about 455.000 pairs of this kind. The following are examples of concept:

SUBRO YES: it is expressed by terms like "*subro potential*" or "*subro claim*" that the compiler of the reports appositely adds to indicate that the case described is a candidate for subrogration.

OPL (Other Part Liability): found by means of expressions like "*has accepted liability*" added by the compiler to indicate that the other driver is responsible. Like the first one, it is indicative of a potential subrogation case, but it is less strong.

INSURED AT FAULT: it is expressed by terms like "*insd was cited*" "*and hit clmt*", representative of the fact that the insured party is probably responsible and, thus, the case is not a subrogation one.

9.3 Experimental Methodology

In both data set, the classification task was that of assigning documents to SUBRO_YES and SUBRO_NO categories. According to the semantic of the FCSI domain, a document is classified under SUBRO_NO if it can't be proved it is SUBRO_YES (*negation by failure*). The manually defined classifier for SUBRO_NO is the following:

$$classify(d, \text{“}SUBRO\_NO\text{”}) \ :- \ onegram(d, X), \\ not\ classify(d, \text{“}SUBRO\_YES\text{”}).$$

Hence, the automatic classifier induction was carried out only for SUBRO_YES; once the best classifier has been find out, it was used, in conjunction with the SUBRO_NO one, to classify the test set documents with respect to both categories.

On both FCSI_6024 and FCSI_2984 training data, we carried out three types of experiments, described below.

1. ***Experiments with only linguistically pre-processed documents.*** The first group of experiments was based on the unique use of n-grams. In this case, FCSI documents were preprocessed according to the linguistic preprocessing step described in section 9.2, so that documents are represented as collection of n-grams with a maximum length 3.

2. ***Experiments with semantically pre-processed documents.*** The second type of experiments was carried out after having performed a semantic analysis task, whereby documents were represented as collection of n-grams and concepts. Both n-grams and concept were, thus, used by Olex to infers classification rules.

3. ***Experiments exploiting the combination of manual and automatic approaches.*** The last group of experiments aimed at exploring the possibility of gaining accuracy through a cooperation of humans efforts and machine learning capability. These experiments, based on the combination of the automatic and manual approaches, exploit the special feature of Olex for the *automatic completion* of manual classifiers, which allows the user to enhance the accuracy of a manually defined set of rules, by performing a subsequent automatic learning process.

9.4 Experimental Results on FCSI_6024

Experiments with only linguistically pre-processed documents

The set of experiments illustrated in Table 9.3 has been carried out for the automatic induction of the best classifier of SUBRO_YES. In particular, for each scoring function (Odds Ratio, Information Gain and Chi Square), we considered reduced vocabularies containing a number of terms ranging between 100 e 300 for each category. Here, the best results were obtained by using Odds Ratio as scoring function. Olex showed to be very effective using the plain n-grams representation of documents: precision and recall values were very balanced and the F-measure was high for both categories (see Table 9.4).

Experiments with the exploitation of external knowledge

For this type of experiments, we used a representation of the training data deriving from the semantic analysis process described in section 9.2. Thus, each document was represented as a set of n-grams and concepts. Consequently, the automatically induced classifiers were expressed in form of both concepts and n-grams. Here, we report a fragment of classifier obtained for SUBRO_YES using this technique:

$$
\begin{array}{lcl}
positive(d, \text{“}SUBRO\_YES\text{”}) & :- & concept(d, \text{“}subro\_open\text{”}). \\
positive(d, \text{“}SUBRO\_YES\text{”}) & :- & onegram(d, \text{“}pip5\text{”}). \\
positive(d, \text{“}SUBRO\_YES\text{”}) & :- & onegram(d, \text{“}srs\text{”}). \\
 & \ldots & \\
negative(d, \text{“}SUBRO\_YES\text{”}) & :- & threegram(d, \text{“}advpaysubro\text{”}). \\
negative(d, \text{“}SUBRO\_YES\text{”}) & :- & threegram(d, \text{“}advpaidsubro\text{”}). \\
classify(d, \text{“}SUBRO\_YES\text{”}) & :- & positive(d, \text{“}SUBRO\_YES\text{”}), \\
 & & not\ negative(d, \text{“}SUBRO\_YES\text{”}).
\end{array}
$$

Scoring function	**Terms/cat**	**category**	**Pr**	**Re**	**F**
Odds Ratio	100	SUBRO_YES	97,38	89,19	93,1
Odds Ratio	150	SUBRO_YES	97,11	90,69	93,79
Odds Ratio	200	SUBRO_YES	96,56	92,79	94,64
Odds Ratio	**250**	**SUBRO_YES**	**95,71**	**93,69**	**94,69**
Odds Ratio	300	SUBRO_YES	95,68	93,09	94,37
Information Gain	100	SUBRO_YES	90,21	91,29	90,75
Information Gain	150	SUBRO_YES	93,52	90,99	92,24
Information Gain	200	SUBRO_YES	93,87	91,89	92,87
Information Gain	250	SUBRO_YES	94,43	91,59	92,99
Information Gain	300	SUBRO_YES	94,67	90,69	92,64
Chi Square	100	SUBRO_YES	91,37	92,19	91,78
Chi Square	150	SUBRO_YES	91,29	91,29	91,29
Chi Square	200	SUBRO_YES	89,88	93,39	91,61
Chi Square	250	SUBRO_YES	94,65	90,39	92,47
Chi Square	300	SUBRO_YES	94,69	90,99	92,8

Table 9.3: Experiments on FCSI_6024 training corpus, using linguistically pre-processed documents.

category	**Pr**	**Re**	**F**
SUBRO_YES	95,71	93,69	94,69
SUBRO_NO	97,29	98,74	98,01
μF	96,86	97,34	97,10

Table 9.4: Micro average values on FCSI_6024 obtained using only linguistically pre-processed documents.

The experiments carried out using concepts representation of documents are summarized in Table 9.5.

The role played by concepts in the learning phase depends on the scoring function used to create the reduced vocabularies. In fact, Odds Ratio prefers n-grams to concepts, while Information Gain and Chi Square are more inclined to select concepts. So, in the first case, the use of concepts has almost no influence and the produced classifiers often coincide with the ones induced by using only n-grams. Nevertheless, the performance results obtained with Information Gain and Chi Square are quite similar to those obtained using only n-grams.

Scoring function	**Terms/cat**	**category**	**Pr**	**Re**	**F**
Odds Ratio	150	SUBRO_YES	97,42	90,69	93,93
Odds Ratio	**200**	**SUBRO_YES**	**96,87**	**92,79**	**94,79**
Odds Ratio	250	SUBRO_YES	95,71	93,69	94,69
Odds Ratio	300	SUBRO_YES	94,58	94,29	94,44
Information Gain	150	SUBRO_YES	91,92	92,19	92,05
Information Gain	200	SUBRO_YES	91,72	93,09	92,4
Information Gain	250	SUBRO_YES	91,47	93,39	92,42
Information Gain	300	SUBRO_YES	90,67	93,39	92,01
Chi Square	150	SUBRO_YES	91,34	91,89	91,62
Chi Square	200	SUBRO_YES	91,69	89,49	90,58
Chi Square	250	SUBRO_YES	91,39	92,49	91,94
Chi Square	300	SUBRO_YES	91,57	91,29	91,43

Table 9.5: Experiments on FCSI_6024 training corpus exploiting external knowledge

Combination of manual and automatic approaches

The third type of experiments carried out on this training corpus aimed at exploring the possibility of combining classifiers written down by knowledge engineers with the ones automatically learned by Olex . To this end, a manual classifier was defined on the basis of FCSI thesaurus concepts. We decided to use concepts, rather than simple n-grams, since they allow an higher level of abstraction. Note that the manual definition of such a classifier requires a depth knowledge about the semantic concepts and their relations. So, we proceeded at defining a set of positive and negative concepts for SUBRO_YES and used a classifier that simply associates a document to SUBRO_YES if it contains any positive concept and no negative one:

$$
\begin{aligned}
neg\_concept(d, \text{“}SUBRO\_YES\text{”}) &:- concept(d, \text{“}insured\_at\_fault\text{”}).\\
neg\_concept(d, \text{“}SUBRO\_YES\text{”}) &:- concept(d, \text{“}subro\_no\text{”}).\\
neg\_concept(d, \text{“}SUBRO\_YES\text{”}) &:- concept(d, \text{“}adverse\text{”}).\\
neg\_concept(d, \text{“}SUBRO\_YES\text{”}) &:- concept(d, \text{“}no\_id\text{”}).\\
neg\_concept(d, \text{“}SUBRO\_YES\text{”}) &:- concept(d, \text{“}hit\_run\text{”}).
\end{aligned}
$$

$$
\begin{aligned}
classify(d, \text{“}SUBRO\_YES\text{”}) &:- concept(d, \text{“}subro\_yes\text{”}),\\
&\quad not\ neg\_concept(d, \text{“}SUBRO\_YES\text{”}).\\
classify(d, \text{“}SUBRO\_YES\text{”}) &:- concept(d, \text{“}opl\text{”}),\\
&\quad not\ neg\_concept(d, \text{“}SUBRO\_YES\text{”}).
\end{aligned}
$$

The performance values obtained by this classifier are:

$$Pr = 95,59 \quad Re = 39,04 \quad F = 55,44$$

which are heavily unbalanced: the classifier is highly precise but has a very low recall. Thus, we used the appropriate Olex feature of the *automatic completion* to automatically complete the manually defined classifier. The automatically generated component of the final classifier, induced by using Odds Ratio scoring function, achieves the following performance values:

$$Pr = 96,33 \quad Re = 94,59 \quad F = 95,45$$

It is worth noticing that the automatic completion helps to improve not only recall but precision too, with a clear increase of the F-measure.
The best performance results obtained with this method are provided in Table 9.6; we point out that this is the overall best classifier.

category	**Pr**	**Re**	**F**
SUBRO_YES	96,33	94,59	95,45
SUBRO_NO	97,95	98,62	98,29
μF	97,51	97,51	97,51

Table 9.6: Micro average values on FCSI_6024 obtained combining manual and automatic approaches

9.5 Experimental Results on FCSI_2984

Experiments using linguistically pre-processed documents

The experiments carried out on this training corpus confirmed the results obtained on FCSI_6024. In order to discover the best classifier for SUBRO_YES we carried out the experiments reported in Table 9.7.

Note that, as for FCSI_6024 training corpus, the best results are obtained using Odds Ratio to extract the input vocabularies. In particular, the best classifier found for SUBRO_YES is that induced by using a vocabulary with 150 terms for category ($Pr = 93,04$, $Re = 87,11$ and $F = 89,98$). Performance values for category SUBRO_NO (using the classifier by negation) are the following: $Pr = 93,04$, $Re = 87,11$ and $F = 88,98$. The micro averaged values are shown

Scoring function	**Terms/cat**	**category**	**Pr**	**Re**	**F**
Odds Ratio	100	SUBRO_YES	93,26	86,26	89,62
Odds Ratio	**150**	**SUBRO_YES**	**93,04**	**87,11**	**89,98**
Odds Ratio	200	SUBRO_YES	89,51	88,24	88,87
Odds Ratio	250	SUBRO_YES	91,57	87,68	89,58
Chi Square	100	SUBRO_YES	91,06	85,13	87,99
Chi Square	150	SUBRO_YES	87,52	88,39	87,95
Chi Square	200	SUBRO_YES	87,35	90,93	89,1
Chi Square	250	SUBRO_YES	86,64	92,78	89,6
Information Gain	100	SUBRO_YES	85,71	89,24	87,44
Information Gain	150	SUBRO_YES	86,98	88,95	87,96
Information Gain	200	SUBRO_YES	84,36	93,2	88,56
Information Gain	250	SUBRO_YES	86,51	92,63	89,47

Table 9.7: Experiments on FCSI_2984 training corpus

in Table 9.8.

category	**Pr**	**Re**	**F**
SUBRO_YES	93,04	87,11	88,98
SUBRO_NO	83,56	90,38	86,97
μF	88,88	88,43	88,65

Table 9.8: Micro average values on FCSI_2984 obtained using only n-grams

Experiments with the exploitation of external knowledge

A set of experiments on FCSI_2984 was carried out to evaluate whether the generation of classifiers based on external knowledge gives a contribution in terms of performance values. Results are shown in Table 9.9.

Unlike the other cases, here the best value of the F-measure for SUBRO_YES is achieved using a reduced vocabulary of 150 terms for category selected using Chi Square as scoring function ($Pr = 88,41,\ Re = 92,92,\ F = 90,61$).

Table 9.10 reports the micro averaged performance values, which are slightly better than those obtained using only n-grams.

Scoring function	**Terms/cat**	**category**	**Pr**	**Re**	**F**
Odds Ratio	100	SUBRO_YES	93,26	86,26	89,62
Odds Ratio	150	SUBRO_YES	92,83	87,96	90,33
Odds Ratio	200	SUBRO_YES	90,26	89,24	89,74
Odds Ratio	250	SUBRO_YES	90,86	90,08	90,47
Chi Square	100	SUBRO_YES	88,05	92,92	90,42
Chi Square	**150**	**SUBRO_YES**	**88,41**	**92,92**	**90,61**
Chi Square	200	SUBRO_YES	86,19	91,08	88,57
Chi Square	250	SUBRO_YES	86,64	90,93	88,74
Information Gain	100	SUBRO_YES	88,54	91,93	90,2
Information Gain	150	SUBRO_YES	88,18	89,8	88,98
Information Gain	200	SUBRO_YES	83,4	93,2	88,03
Information Gain	250	SUBRO_YES	91,47	88,1	89,75

Table 9.9: Experiments on FCSI_2984 using concepts

category	**Pr**	**Re**	**F**
SUBRO_YES	88,41	92,92	90,61
SUBRO_NO	88,91	82,01	85,78
μF	88,96	88,51	88,74

Table 9.10: Micro average values on FCSI_2984 obtained using n-grams and concepts

Combination of manual and automatic approaches

The automatic completion feature was evaluated on this corpus, using the manual classifier designed for SUBRO_YES in corpus FCSI_6024. This was possible since the manual classifier was based only on the concepts defined in FCSI thesaurus (which is the same for both training corpora). Here, the starting performance values of the manual classifier were:

$$Pr = 54,69 \quad Re = 54,37 \quad F = 54,33$$

The usefulness of automatic completion feature was confirmed by its evaluation on this training corpus. In fact, the performance achieved by adding the automatically learned component are:

$$Pr = 88,41 \quad Re = 92,92 \quad F = 90,61$$

which are the same of the ones obtained by using the automatic approaches on vocabularies containing n-grams and concepts.

Chapter 10

Discussion and Conclusion

10.1 Discussion

The experimental results reported in the previous chapters show that Olex can induce classifiers that are highly accurate (compared to other state-of-the-art systems) and compact (which entails faster classification and human readability). Those experiments also suggest a good capacity to learn from low-frequency categories. Interestingly, those properties have consistently been observed on both benchmark corpora and FCSI data sets, on which Olex shows a uniform behavior. Given the very different application domains the corpora refer to, this is a clear proof of robustness. We think that the observed behavior is general, not corpus-dependent.
How can we explain all that?

The hypothesis language

Quite obviously, the above properties are consequence of a very expressive hypothesis language. Basically, what is effective, and original, in our approach is the combination of negative and conjunctive literals within the general framework "one positive literal, more negative literals". Intuitively, it seems that positive literals allow rules to catch most of the right documents, while negative ones help them not to make "too many" mistakes. In other words, negative literals allow Olex to achieve high precision values while retaining the high recall coming from positive literals (a similar effect has been observed in [86, 10, 8]).
Conjunctions of terms, like negative literals, specialize rules to be more precise. The presence of conjunctions is pretty frequent in the Olex rules, albeit they are

usually limited to just two terms (three-term conjunctions are indeed very rare). For instance, in the best classifier for "earn", there are 11 two-term positive literals (over 30 rules) and 3 two-term negative literals (in each rule). No three-term literal occurs. We note that higher co-term degree might cause overfitting (which might indeed be important if conjunctions were "too long"). Also, the low degree of co-terms represents a good compromise between the need of specializing rules to get better precisions and their capability to "cover" (positive) examples. In general, the presence of "short" conjunctions tends to make rules more reliable, and this may also explain why conjunctive rules can learn well even from low-frequency categories (contrary to simple rules). All this results in a remarkable increase in performance, as shown in Subsection 8.5. On both benchmark corpora, indeed, the rise in performance due to conjunctions is relevant (around 6% for the REUTERS-21578 and 4% for OHSUMED). To the best of our knowledge, no other rule-based classifiers allow conjunctive literals.

Optimization heuristics

Effective rules are the consequence of a quite strong optimization technique in finding discriminating terms. As we have seen, it is based on a greedy heuristics which efficiently induces accurate classifiers. In Section 8.8, we have examined a number of other machine learning techniques for document classification, from which it clearly appears that Olex is more than competitive. Can we enhance our optimization heuristics to get even better results than those reported in this thesis? Although we cannot say *a priori* how much those results are improvable (we do not know indeed what is the performance upper bound) we suspect that there is still room for enhancement. This is because there are some restrictions on the d-terms generated by the proposed heuristics which may actually limit rule effectiveness. One of the major drawbacks is that a term can appear in at most one conjunctive term, so that the positive literals occurring in the rules of a classifier cannot have terms in common. As an instance, a classifier like the following

(see [82])

$$
\begin{aligned}
wheat &\leftarrow \text{“}wheat\text{”} \in d, \text{“}farm\text{”} \in d.\\
wheat &\leftarrow \text{“}wheat\text{”} \in d, \text{“}commodity\text{”} \in d.\\
wheat &\leftarrow \text{“}wheat\text{”} \in d, \text{“}agriculture\text{”} \in d.\\
wheat &\leftarrow \text{“}wheat\text{”} \in d, \text{“}tonnes\text{”} \in d.\\
&\ldots
\end{aligned}
$$

where the word “wheat” occurs positively in more rules, could not be generated by Olex. This may seriously limit the effectiveness of the induced classifiers. We point out that this is a restriction of the chosen heuristics, not of the proposed model. We are currently investigating alternative heuristics for our optimization task, e.g., genetic algorithms.

Relation to other inductive rule learners

Because of the computational complexity of the learning problem, real systems employ heuristic search strategies which prunes vast parts of the hypothesis space. Conventional inductive rule learners (such as, FOIL [20], RIPPER [24], Swap-1 [82]) usually adopt, as their general search method, a covering approach based on a divide-and-conquer strategy. Starting from an empty rule set S, a training set TS and a background knowledge B, they learn a set of rules, one by one. The learning of a single rule r, from TS and B, starts with a clause with an empty body which is specialized by repeatedly adding a body literal to the clause built so far. Then, r is added to S and all the examples explained by r are removed from TS. This process is reiterated until TS is empty or S satisfies all positive examples. Different learners essentially differ in how they find a single rule. In RIPPER, for instance, the construction of a single rule is a two-stage process: a greedy heuristics constructs an initial ruleset (IREP*) and, then, an optimization phase improves compactness and accuracy of the ruleset. In the first stage, to build a rule, the uncovered examples are randomly partitioned into two subsets: a growing set and a pruning set. That is, after growing a rule, it is simplified by another greedy technique.
Also decision tree techniques, e.g., C4.5 [66], rely on a two-stage process. After the decision tree has been transformed into a rule set, C4.5 implements a pruning stage which requires more steps to produce the final rule set - a rather complex and time consuming task.

As we have seen, Olex does not follows the aforementioned scheme. Rather, it relies on a simple, yet effective, optimization algorithm for the computation of discriminating terms. Unlike both RIPPER and C4.5, which requires lengthy optimization processes, 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.

10.2 Concluding Remarks

In this thesis, we have presented Olex, a novel approach to the automatic induction of rule-based text classifiers. We provided a formal description of the method. In particular, we described the problem of determining a best set of discriminating terms for a category and proved its intractability. Then, we showed how rules are derived from a given set of discriminating terms.
The Olex's hypothesis language consists of rules with one positive conjunction of terms and (zero or) more negative ones. Thus, Olex predictions require testing the simultaneous presence of several terms (forming the positive conjunction) along with the simultaneous absence of several sets of terms (each forming a negative conjunction).
While there is in the literature a wide range of rule learning algorithms, one contribution of our approach is the form of the hypothesis language.
We performed experiments on two standard data collections: REUTERS-21578 and OHSUMED . We found that Olex consistently achieves high performance results, significantly outperforming most of the traditional approaches.Further, it provides very compact and comprehensible classifiers.
Olex enjoys a number of further desirable properties:

1. it is accurate even for relatively small categories (i.e., it is not biased towards majority classes);

2. it is robust, i.e., shows a similar behavior on both data sets we have experimented.

Further, thanks to its rule-based approach, Olex allows an immediate and sound integration of background knowledge, in order to enhance the accuracy of the induced classifiers with features generated on the basis of domain-specific and common-sense knowledge. Experiments have been carried out the evaluate usefulness of external knowledge on FCSI corpora, by performing a *Semantic*

Analysis task, whereby documents have been represented in terms of the extracted concepts. This first empirical evaluation, provided in chapter 9, showed that knowledge-based feature generation gives a small contribution in the improvement of classification performances. This is a partial result; further investigation have to be carried out, in order to state whether this result can be generalized for our learning approach or some improvement is obtained when using more structurally complex or more appropriate thesauri.

Lastly, the system supports the integration of a manual approach into the automatic categorization. Thanks to the interpretability of the produced classifiers, indeed, the Knowledge Engineer can participate in the construction of a classifier, by manually specifying a set of rules to be used in conjunction with those automatically learned. Experimental results showed that this cooperation may bring Text Categorization to an higher performance level.

All this makes Olex an interesting approach for learning text classifiers from training sets.

Bibliography

[1]

[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, 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.

[5] 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.

[6] Ion Androutsopoulos, John Koutsias, Konstandinos V. Chandrinos, and Constantine D. Spyropoulos. An experimental comparison of naive bayesian and keyword-based anti-spam filtering with personal e-mail messages. In Nicholas J. Belkin, Peter Ingwersen, and Mun-Kew Leong, editors, *Proceedings of SIGIR-00, 23rd ACM International Conference on Research and Development in Information Retrieval*, pages 160–167, Athens, GR, 2000. ACM Press, New York, US.

[7] 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.

[8] 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.

[9] 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.

[10] 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.

[11] 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.

[12] Eric Brill. A simple rule-based part-of-speech tagger. In *Proceedings of ANLP-92, 3rd Conference on Applied Natural Language Processing*, pages 152–155, Trento, IT, 1992.

[13] 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.

[14] Fellbaum C. *Wordnet: An Electronic Lexical Database*. The MIT Press, 1998.

[15] 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.

[16] William B. Cavnar and John M. Trenkle. N-gram-based text categorization. In *Proceedings of SDAIR-94, 3rd Annual Symposium on Document Analysis and Information Retrieval*, pages 161–175, Las Vegas, US, 1994.

[17] Ming-Syan Chen, Jiawei Han, and Philip S. Yu. Data mining: an overview from a database perspective. *Ieee Trans. On Knowledge And Data Engineering*, 8:866–883, 1996.

[18] W. Cohen. PAC-learning recursive logic programs: Negative result. *Journal of Artificial Intelligence Research*, 2:541–573, 1995.

[19] William W. Cohen. Learning to classify english text with ilp methods. In Luc De Raedt, editor, *Advances in inductive logic programming*, pages 124–143. IOS Press, Amsterdam, NL, 1995.

[20] William W. Cohen. Text categorization and relational learning. In Armand Prieditis and Stuart J. Russell, editors, *Proceedings of ICML-95, 12th International Conference on Machine Learning*, pages 124–132, Lake Tahoe, US, 1995. Morgan Kaufmann Publishers, San Francisco, US.

[21] William W. Cohen. Learning rules that classify E-mail. In *Papers from the AAAI Spring Symposium on Machine Learning in Information Access*, pages 18–25. AAAI Press, 1996.

[22] 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.

[23] 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 [24].

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

[25] Koby Crammer and Yoram Singer. A new family of online algorithms for category ranking. In Micheline Beaulieu, Ricardo Baeza-Yates, Sung Hyon Myaeng, and Kalervo Järvelin, editors, *Proceedings of SIGIR-02, 25th ACM International Conference on Research and Development in Information Retrieval*, pages 151–158, Tampere, FI, 2002. ACM Press, New York, US.

[26] Franca Debole and Fabrizio Sebastiani. An analysis of the relative difficulty of reuters-21578 subsets. In *Proceedings of LREC-04, 4th International Conference on Language Resources and Evaluation*, Lisbon, PT, 2004. Forthcoming.

[27] Harris Drucker, Vladimir Vapnik, and Dongui Wu. Support vector machines for spam categorization. *IEEE Transactions on Neural Networks*, 10(5):1048–1054, 1999.

[28] Susan T. Dumais and Hao Chen. Hierarchical classification of web content. In Nicholas J. Belkin, Peter Ingwersen, and Mun-Kew Leong, editors, *Proceedings of SIGIR-00, 23rd ACM International Conference on Research and Development in Information Retrieval*, pages 256–263, Athens, GR, 2000. ACM Press, New York, US.

[29] Susan T. Dumais, John Platt, David Heckerman, and Mehran Sahami. Inductive learning algorithms and representations for text categorization. In Georges Gardarin, James C. French, Niki Pissinou, Kia Makki, and Luc Bouganim, editors, *Proceedings of CIKM-98, 7th ACM International Conference on Information and Knowledge Management*, pages 148–155, Bethesda, US, 1998. ACM Press, New York, US.

[30] Saso Dzeroski, Stephen Muggleton, and Stuart J. Russell. PAC-learnability of determinate logic programs. In *Proceedings of the Fifth Annual ACM Workshop on Computational Learning Theory (COLT-92)*, Pittsburgh, Pennsylvania, 1992. ACM Press.

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

[32] Richard S. Forsyth. New directions in text categorization. In Alex Gammerman, editor, *Causal models and intelligent data management*, pages 151–185. Springer Verlag, Heidelberg, DE, 1999.

[33] 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.

[34] Norbert Fuhr and Gerhard Knorz. Retrieval test evaluation of a rule-based automated indexing (AIR/PHYS). In Cornelis J. Van Rijsbergen, editor, *Proceedings of SIGIR-84, 7th ACM International Conference on Research and Development in Information Retrieval*, pages 391–408, Cambridge, UK, 1984. Cambridge University Press.

[35] 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.

[36] William A. Gale, Kenneth W. Church, and David Yarowsky. A method for disambiguating word senses in a large corpus. *Computers and the Humanities*, 26(5):415–439, 1993.

[37] Bart Goethals and Mohammed Javeed Zaki, editors. *FIMI '03, Frequent Itemset Mining Implementations, Proceedings of the ICDM 2003 Workshop on Frequent Itemset Mining Implementations, 19 December 2003, Melbourne, Florida, USA*, volume 90 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2003.

[38] 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, Boston, US, 1990. AAAI Press, Menlo Park, US.

[39] Marti A. Hearst. Noun homograph disambiguation using local context in large corpora. In *Proceedings of the 7th Annual Conference of the University of Waterloo Centre for the New Oxford English Dictionary*, pages 1–22, Oxford, UK, 1991.

[40] 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.

[41] Makoto Iwayama and Takenobu Tokunaga. Cluster-based text categorization: a comparison of category search strategies. 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 273–281, Seattle, US, 1995. ACM Press, New York, US.

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

[43] 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, Heidelberg, DE. Published in the "Lecture Notes in Computer Science" series, number 1398.

[44] Thorsten Joachims. Transductive inference for text classification using support vector machines. In Ivan Bratko and Saso Dzeroski, editors, *Proceedings of ICML-99, 16th International Conference on Machine Learning*, pages 200–209, Bled, SL, 1999. Morgan Kaufmann Publishers, San Francisco, US.

[45] 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.

[46] Brett Kessler, Geoff Nunberg, and Hinrich Schütze. Automatic detection of text genre. In Philip R. Cohen and Wolfgang Wahlster, editors, *Proceedings of ACL-97, 35th Annual Meeting of the Association for Computational Linguistics*, pages 32–38, Madrid, ES, 1997. Morgan Kaufmann Publishers, San Francisco, US.

[47] Jörg-Uwe Kietz and Sašo Džeroski. Inductive logic programming and learnability. *SIGART Bull.*, 5(1):22–32, 1994.

[48] Ralf Klinkenberg and Thorsten Joachims. Detecting concept drift with support vector machines. In Pat Langley, editor, *Proceedings of ICML-00, 17th International Conference on Machine Learning*, pages 487–494, Stanford, US, 2000. Morgan Kaufmann Publishers, San Francisco, US.

[49] W. Kloesgen. Explora: A multipattern and multistrategy discovery assistant, 1996.

[50] Daphne Koller and Mehran Sahami. Hierarchically classifying documents using very few words. In Douglas H. Fisher, editor, *Proceedings of ICML-97, 14th International Conference on Machine Learning*, pages 170–178, Nashville, US, 1997. Morgan Kaufmann Publishers, San Francisco, US.

[51] Leah S. Larkey. Automatic essay grading using text categorization techniques. In W. Bruce Croft, Alistair Moffat, Cornelis J. Van Rijsbergen, Ross Wilkinson, and Justin Zobel, editors, *Proceedings of SIGIR-98, 21st ACM International Conference on Research and Development in Information Retrieval*, pages 90–95, Melbourne, AU, 1998. ACM Press, New York, US.

[52] Leah S. Larkey and W. Bruce Croft. Combining classifiers in 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 289–297, Zürich, CH, 1996. ACM Press, New York, US.

[53] David D. Lewis. An evaluation of phrasal and clustered representations on a text categorization task. In Nicholas J. Belkin, Peter Ingwersen, and Annelise Mark Pejtersen, editors, *Proceedings of SIGIR-92, 15th ACM International Conference on Research and Development in Information Retrieval*, pages 37–50, Kobenhavn, DK, 1992. ACM Press, New York, US.

[54] 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.

[55] David D. Lewis. Reuters-21578 text categorization test collection. Distribution 1.0, 1997. Available as `http://www.daviddlewis.com/resources/testcollections/reuters21578/readme.txt`.

[56] 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.

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

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

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

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

[61] Briji Masand, Gordon Linoff, and David Waltz. Classifying news stories using memory-based reasoning. In Nicholas J. Belkin, Peter Ingwersen, and Annelise Mark Pejtersen, editors, *Proceedings of SIGIR-92, 15th ACM International Conference on Research and Development in Information Retrieval*, pages 59–65, Kobenhavn, DK, 1992. ACM Press, New York, US.

[62] Andrew McCallum and K. Nigam. A comparison of event models for naive bayes text classification. In *Proceedings of AAAI-98, Workshop on Learning for Text Categorization*, 1998.

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

[64] Kary Myers, Michael Kearns, Satinder Singh, and Marilyn A. Walker. A boosting approach to topic spotting on subdialogues. In Pat Langley, editor,

Proceedings of ICML-00, 17th International Conference on Machine Learning, pages 655–662, Stanford, US, 2000. Morgan Kaufmann Publishers, San Francisco, US.

[65] Christoph Schommer Qin Sun and Alexander Lang.

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

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

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

[69] Carl L. Sable and Vasileios Hatzivassiloglou. Text-based approaches for non-topical image categorization. *International Journal of Digital Libraries*, 3(3):261–275, 2000.

[70] Ashoka Savasere, Edward Omiecinski, and Shamkant B. Navathe. Mining for strong negative associations in a large database of customer transactions. In *ICDE*, pages 494–502, 1998.

[71] Robert E. Schapire and Yoram Singer. BOOSTEXTER: a boosting-based system for text categorization. *Machine Learning*, 39(2/3):135–168, 2000.

[72] Sam Scott and Stan Matwin. Text classification using WordNet hypernyms. In Sanda Harabagiu, editor, *Use of WordNet in Natural Language Processing Systems: Proceedings of the Conference*, pages 38–44. Association for Computational Linguistics, Somerset, New Jersey, 1998.

[73] 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 [74].

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

[75] Qin Sun, Christoph Schommer, and Alexander Lang. Integration of manual and automatic text categorization. a categorization workbench for text-based email and spam. In *KI*, pages 156–167, 2004.

[76] Hirotoshi Taira and Masahiko Haruno. Feature selection in svm text categorization. In *Proceedings of AAAI-99, 16th Conference of the American Association for Artificial Intelligence*, pages 480–486, Orlando, US, 1999. AAAI Press, Menlo Park, US.

[77] Konstadinos Tzeras and Stephan Hartmann. Automatic indexing based on bayesian inference networks. In Robert Korfhage, Edie Rasmussen, and Peter Willett, editors, *Proceedings of SIGIR-93, 16th ACM International Conference on Research and Development in Information Retrieval*, pages 22–34, Pittsburgh, US, 1993. ACM Press, New York, US.

[78] L. G. Valiant. A theory of the learnable. In *STOC '84: Proceedings of the sixteenth annual ACM symposium on Theory of computing*, pages 436–445, New York, NY, USA, 1984. ACM.

[79] Vladimir N. Vapnik. *The nature of statistical learning theory*. Springer-Verlag New York, Inc., New York, NY, USA, 1995.

[80] Flavian Vasile, Adrian Silvescu, Dae-Ki Kang, and Vasant Honavar. Tripper: Rule learning using taxonomies. In *PAKDD*, pages 55–59, 2006.

[81] 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.

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

[83] Sholom M. Weiss, Chidanand Apté, Fred J. Damerau, David E. Johnson, Frank J. Oles, Thilo Goetz, and Thomas Hampp. Maximizing text-mining performance. *IEEE Intelligent Systems*, 14(4):63–69, 1999.

[84] 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.

[85] Ian H. Witten and Eibe Frank. *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann, San Francisco, 2 edition, 2005.

[86] 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.

[87] 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.

[88] Yiming Yang and Christopher G. Chute. An example-based mapping method for text categorization and retrieval. *ACM Transactions on Information Systems*, 12(3):252–277, 1994.

[89] Yiming Yang and Xin Liu. A re-examination of text categorization methods. In Marti A. Hearst, Fredric Gey, and Richard Tong, editors, *Proceedings of SIGIR-99, 22nd ACM International Conference on Research and Development in Information Retrieval*, pages 42–49, Berkeley, US, 1999. ACM Press, New York, US.

[90] 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.

[91] Osmar R. Zaïane and Maria-Luiza Antonie. Classifying text documents by associating terms with text categories. In *Proceedings of the 13th Australasian Conference on Database Technologies*, volume 5, pages 215–222, Melbourne, AU, 2002. ACM Press, New York, US. This paper has also been published in *Australian Computer Science Communications*, 24(2), 2002.

[92] 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.