Università della Calabria

Dipartimento di Matematica e Informatica

Dottorato di Ricerca in Matematica ed Informatica

Settore Disciplinare INF/01 - Informatica

XXV Ciclo

Tesi di Dottorato

GAMoN

Discovering M-of-N Hypotheses
for Text Classification
by a Lattice-based Genetic Algorithm

Adriana Pietramala

Supervisore	**Coordinatore**
Prof. Pasquale Rullo	Prof. Nicola Leone

A.A. 2013 - 2014

UNIVERSITÀ DELLA CALABRIA

Università della Calabria

Dipartimento di Matematica e Informatica

Dottorato di Ricerca in Matematica ed Informatica

Settore Disciplinare INF/01 - Informatica

XXV Ciclo

Tesi di Dottorato

GAMoN

Discovering M-of-N Hypotheses
for Text Classification
by a Lattice-based Genetic Algorithm

Adriana Pietramala

Adriana Pietramala

Supervisore

Prof. Pasquale Rullo

Coordinatore

Prof. Nicola Leone

A.A. 2013 - 2014

GAMoN
Discovering M-of-N Hypotheses for Text Classification by a Lattice-based Genetic Algorithm

AdrianaPietramala

Dipartimento di Matematica e Informatica
Università della Calabria
87036 *Rende, Italy*

email : *a.pietramala@mat.unical.it*

Sommario

Lo sviluppo delle moderne tecnologie informatiche, nonché la diffusione dei servizi per il Web, ha portato ad una considerevole produzione di informazioni e dati di diversa natura: documenti testuali (dati non strutturati), basi di dati (dati strutturati) e pagine Html (dati semi-strutturati). La disponibilità, sempre più crescente, di considerevoli quantità di dati ha posto, di conseguenza, il problema della loro memorizzazione, della loro organizzazione e del loro reperimento. Inoltre, se non ci fossero strumenti idonei a trattare le sole informazioni di interesse, tutti questi dati rischierebbero di essere inutilizzabili. Le informazioni, infatti, rappresentano il punto di partenza per l'estrazione di conoscenza, attività che, in passato, ha fatto riferimento all'analisi e all'interpretazione manuale, fondata sull'attività di uno o più esperti addetti a prendere le decisioni sul caso corrente. L'analisi manuale, chiaramente, presenta molteplici aspetti negativi. Prima tra tutti essa è caratterizzata da lunghi tempi di analisi e da alti costi di realizzazione; infine, risulta altamente soggettiva e in accurata. Tali aspetti negativi vengono ulteriormente aggravati dall'enorme mole di dati da dover trattare. Aggregare, classificare e recuperare le informazioni di interesse con tempestività, efficacia e a costi ridotti è sicuramente più vantaggioso rispetto ai tradizionali approcci di analisi manuale. In particolare, la possibilità di poter classificare automaticamente enormi quantità di documenti, potendoli poi ritrovare facilmente sulla base dei concetti espressi e sulle tematiche trattate, piuttosto che affidarsi ad un'analisi manuale, è una necessità che viene sentita non solo dalla comunità scientifico/accademica, ma anche da quella aziendale, commerciale e finanziaria.

Il Text Classification (TC) o Text Categorization è una disciplina che coniuga diverse aree di ricerca, dall'Information Retrieval (IR), al Machine Learning (ML), al Natural Language Processing (NLP) e mira alla costruzione di sistemi per la classificazione automatica dei dati in categorie tematiche di interesse. In particolare, nel TC, i dati sono costituiti da una collezione di documenti testuali non strutturati, i quali vengono suddivisi in gruppi sulla base del contenuto, attraverso l'assegnamento del testo ad una o più categorie tematiche predefinite. Le prime ricerche nell'ambito del TC risalgono all'inizio degli anni '60. Tuttavia, è solo nell'ultimo decennio che tale problema sta suscitando un interesse crescente sia nel settore della ricerca scientifica che in contesti industriali. Possibili applicazioni del TC spaziano dall'indicizzazione automatica di articoli scientifici, all'organizzazione delle e-mail, al filtraggio dello spam, ecc.

Negli ultimi decenni, sono stati proposti un gran numero di sistemi per la classificazione di documenti testuali suddivisibili, principalmente, in tre macro-tipologie sulla base dell'approccio seguito nella costruzione dei classificatori:

- approccio di tipo Expert Systems (ES);

- approccio di tipo Machine Learning (ML);
- approccio di tipo Ibrido.

Il primo approccio, affermatosi all'inizio degli anni '60 prevede l'impiego di esperti di dominio (classificazione manuale) nella definizione dei classificatori per le categorie di interesse. Questo tipo di approccio ha consentito la definizione di classificatori molto efficaci. Di contro, però, l'approccio di tipo ES presenta due svantaggi principali: risulta molto dispendioso in termini di risorse umane utilizzate e poco flessibile. Infatti, nel momento in cui cambia il contesto di riferimento, i nuovi classificatori devono essere nuovamente definiti manualmente. Per questo motivo, a partire dagli anni '90, l'approccio di tipo ES è stato quasi completamente sostituito dall'approccio di tipo ML, il cui obiettivo principale non è la definizione dei classificatori, quanto la costruzione di sistemi in grado di generare automaticamente i classificatori. Più in particolare, nell'ambito di questo paradigma, l'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. 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 svariati sistemi per la classificazione automatica di documenti testuali basati, essenzialmente, su processi di tipo induttivo. Tali sistemi sfruttano, generalmente, misure statistiche e, talvolta, vengono importati nell'ambito del TC da altre aree dell'Information Retrieval e del Data Mining. Un esempio emblematico è il caso delle Support Vector Machine (SVM) utilizzate, dapprima, per la risoluzione di 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 "a regole" o "rule-based", dove i classificatori vengono specificati come insiemi di regole. Tali classificatori hanno la proprietà desiderabile di essere comprensibili da un lettore umano, mentre la maggior parte degli altri approcci esistenti, come SVM e Neural Network, producono classificatori che difficilmente un lettore umano riesce ad interpretare. Classificatori con queste caratteristiche vengono spesso chiamati di tipo *black-box*. Infine, l'approccio di tipo Ibrido combina il metodo Expert System con quello Machine Learning, per ottenere un sistema di categorizzazione che sfrutta sia i benefici derivanti da una conoscenza di dominio, sia i benefici derivanti dalla costruzione di sistemi automatici.

Ultimamente, la comunità scientifica sta adottando tecniche di TC sempre più innovative che, generalmente, si discostano di molto dagli approcci classici di tipo deterministico. In effetti, una recente tendenza nell'ambito del TC è quella di sfruttare tecniche di apprendimento basate su *meta-euristiche*, come gli Algoritmi Evoluzionistici o Genetici. Tecniche di questo tipo sono, general-

mente, costituite da tre componenti essenziali:

- un insieme di soluzioni candidate, chiamato *popolazione*, costituito da individui o cromosomi. Questi evolvono durante un certo numero di iterazioni (generazioni) generando, alla fine dell'evoluzione, la soluzione migliore;
- una funzione obiettivo, chiamata *funzione di fitness*, usata per assegnare a ciascun individuo un peso (score) che indica la bontà dell'individuo stesso;
- un meccanismo evolutivo, basato su operatori evoluzionistici come *crossover*, *mutazione* ed *elitismo*, che consentono di modificare il materiale genetico degli individui che costituiscono la popolazione.

Approcci di questo tipo introducono notevoli vantaggi rispetto alle tecniche classiche. Ad esempio, il meccanismo evolutivo è noto per essere un metodo robusto e di successo, infatti, è utilizzato per la risoluzione di molti problemi di ottimizzazione intrinsecamente difficili da risolvere. Inoltre, il meccanismo evolutivo riduce sensibilmente lo spazio di ricerca delle soluzioni ammissibili e molte tecniche evolutive riescono a risolvere problemi complessi senza conoscere il preciso metodo di soluzione.

In questo lavoro di tesi proponiamo un modello di classificazione a regole, denominato GAMoN, basato sull'utilizzo di Algoritmi Genetici per l'induzione delle regole di classificazione. Un classificatore $\mathcal{H}$ generato dal sistema GAMoN per una data categoria $\boldsymbol{c}$ assume la forma di una disgiunzione di atomi $\mathcal{H}_c^i$ del tipo:

$$\mathcal{H}_c = \mathcal{H}_c^1 \vee \cdots \vee \mathcal{H}_c^r$$

dove ciascun atomo $\mathcal{H}_c^i$ è una quadrupla $< Pos, Neg, m_i, n_i >$, dove:

- $Pos = \{t_1, .., t_n\}$ è l'insieme dei termini positivi, ovvero l'insieme dei termini che sono rappresentativi per la categoria *c* di riferimento;
- $Neg = \{t_{n+1}, , t_{n+m}\}$ è l'insieme dei termini negativi, ovvero l'insieme dei termini che sono indicativi della non appartenenza alla categoria;
- m_i e n_i sono numeri naturali, chiamati *soglie*, tali che $m_i >= 0$ e $n_i > 0$.

Intuitivamente, il significato attribuito a ciascun atomo $\mathcal{H}_c^i$ è il seguente: "classifica il generico documento *d* sotto la categoria *c* se almeno m_i termini positivi compaiono in *d* e meno di n_i termini negativi compaiono in *d*". Infatti, il linguaggio delle ipotesi introdotto da GAMoN è chiamato $MofN^+$, una estensione dei classificatori di tipo $MofN$ con la componente dei termini negativi. Da qui nasce l'acronimo "GAMoN", che sta ad indicare un sistema di classificazione testuale basato su "Algoritmi Genetici" di tipo "M of N". GAMoN è un sistema di classificazione che

nasce come estensione di "Olex-GA", un modello di classificazione "a regole" basato sul paradigma evoluzionistico e realizzato in precedenti lavori di ricerca. Un classificatore generato da GAMoN coincide con quello di Olex-GA quando m_i=1 e n_i = 1. Infatti, un classificatore Olex-GA assume il significato "se almeno uno dei termini positivi $t_1, ..., t_n$ appare nel documento *d* e nessuno dei termini negativi $t_{n+1}, , t_{n+m}$ appare in *d*, allora classifica *d* sotto la categoria *c*".

Il sistema GAMoN è stato testato su 13 *corpora di benchmark* (Reuters-21578, Ohsumed, OH5, OH0, OH10, OH15, Blogs Gender, Ohscale, 20 Newsgroups, Cade, SRAA, ODP e Market) e messo a confronto con altri 5 sistemi di classificazione: BioHEL [18, 48] e Olex-GA [101], che sono sistemi di classificazione *a-regole* basati sul paradigma evoluzionistico; Ripper [37] e C4.5 [105], che sono sistemi di classificazione *a-regole* non evoluzionistici; infine, SMO che è una implementazione di SVM lineare [76]. Gli studi sperimentali mettono in evidenza come GAMoN induca classificatori che sono, al tempo stesso, accurati e compatti. Tale proprietà è stata osservata su tutti i corpora utilizzati nella sperimentazione, dove GAMoN ha mostrato sempre un comportamento uniforme. Poiché i corpora utilizzati si riferiscono a contesti applicativi notevolmente diversi, possiamo affermare che GAMoN ha dato prova di essere un sistema robusto. Complessivamente, GAMoN ha dimostrato un buon bilanciamento tra accuratezza e complessità del modello generato; inoltre, è risultato molto efficiente per la classificazione di corpora di grandi dimensioni.

Il seguito della tesi è organizzato in tre parti principali di seguito elencate:

- nella Parte I verrà definito formalmente il problema del Text Categorization e verranno rivisitati i principali contesti applicativi nei quali sono sfruttate tecniche di questo tipo;

- nella Parte II verranno presentati diversi metodi e sistemi di classificazione documentale, al fine di realizzare una valutazione comparativa delle loro peculiarità nell'ambito della tematica di interesse;

- nella Parte III verrà presentato dettagliatamente il sistema GAMoN. In particolare, verranno riportate alcune definizioni formali quali, ad esempio, il linguaggio e lo spazio delle ipotesi, gli operatori di crossover utilizzati dal sistema e verranno descritti e mostrati i risultati sperimentali ottenuti, attraverso un'analisi comparativa con i sistemi di learning sù citati.

Abstract

The development of modern information technology and the diffusion of services for the Web, has led to a considerable production of information and data, of a different kind: textual documents (unstructured data), databases (structured data) and HTML pages (semi-structured data). The availability, more and more increasing, of a considerable amounts of data has place, consequently, the problem of storing them, of their organization and their retrieval. Furthermore, if there were no instruments that treat only the information of interest, all of these data would risk to being unusable. The information, in fact, represent the starting point for the knowledge extraction, activity that, in the past, made reference to the manual analysis and interpretation, consisting in the manual definition of a classifier by one ore more domain experts. The manual analysis, of course, introduces many negative aspects. First of all it is characterized by long analysis times and high costs of implementation and, finally, it is highly subjective and not accurate. These negative aspects are further aggravated by the huge amount of data to be treated. Aggregate, classify and retrieve the information of interest with a timeliness, effectiveness and at reduced cost is certainly more advantageous than traditional approaches of manual analysis. In particular, the possibility to automatically classify a huge amounts of documents, rather than relying on manual analysis, it is a necessity that is felt not only by the scientific/academic community, but also by the commercial and financial companies.

The Text Classification (TC) or Text Categorization is a discipline that combines different research areas like Information Retrieval (IR), Machine Learning (ML), Natural Language Processing (NLP) and aims to build systems for the automatic classification of data into predefined thematic categories of interest. Specifically, in the TC, the data consists of a collection of textual unstructured documents, which are divided into groups based on the content, through the assignment of the text to one or more predefined thematic 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. Applications of the TC range to automated indexing of scientific articles, to e-mail routing, spam filtering, authorship attribution, and automated survey coding.

In the last years, a large number of systems for the classification of textual documents have been proposed, but three are the main approaches to Text Categorization problem:

- Expert Systems (ES) approach;
- Machine Learning (ML) approach;
- Hybrid approach.

The first approach, has been proposed in the '60s and it is based on the manual definition of classifiers by one or more domain experts(manual classification). Experimental results showed that this technique can give very good effectiveness results. However, the ES approach presents two main disadvantages: is a very costly activity and it is low flexible. In fact, if the set of categories change, new classifiers must be manually redefined by the domain experts. For this reason, since the early '90s, the Machine Learning approach to the construction of text classifiers has gained popularity. The ML approach, aiming at the construction not of a classifier, but of an automatic builder of classifiers (the learner).

More in particular, in this approach a general inductive process (also called the learner) automatically builds 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 $\overline{c_i}$ (training set) 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. This approach presents numerous advantages compared to that of Expert Systems: generally show high efficacy, is less expensive and provides immediate portability to new domains (categories).

In the last years, a great number of statistical classification and machine learning methods have been proposed, based on an inductive process. These systems exploit, in general, statistical measures that, sometimes, are imported under the TC from other areas of Information Retrieval and Data Mining. Is the case of Support Vector Machine (SVM), initially used for the resolution of regression problems and, currently, considered the state of the art for Text Categorization.

An important place among the inductive paradigm is represented by the *rule-based* models, where the classifiers are specified as sets of rules. Rule-based classifiers, instead, provide the desirable property of being readable and easy for people to understand, while most of the other existing approaches, such as SVM and Neural Network, produce classifiers that are difficult to interpret by a human reader. Classifiers with these characteristics are, often, called *black-box classifiers*. Finally, the hybrid approach 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.

Lately, the scientific community is increasingly adopting innovative TC techniques, which differ from classical deterministic approaches. In fact, a recent trend in the TC is the usage of learning techniques based on meta-heuristics approaches, such as genetic or evolutionary algorithms. This techniques are, generally, made up of three components:

- a set of candidate solutions, called *population*, consisting of *individuals* or *chromosomes*. These evolve during a certain number of iterations (generations) generating, at the end of the evolution, the best solution;

- an objective function, called *fitness* function, that assigns a weight (score) to each individual. The fitness function indicates the goodness to the individuals;

- an *evolutionary mechanism*, based on evolutionary operators such as *crossover*, *mutation* and *elitism*, which modify the genetic material of the individuals that make up the population.

This approaches introduce considerable advantages over the classic techniques. For example, the evolutionary mechanism is known to be a robust and successful method, in fact, it is used for the resolution of many optimization problems inherently difficult to solve. Furthermore, the evolutionary mechanism reduces significantly the search space of admissible solutions and many evolutionary techniques fail to solve complex problems without knowing the precise method of solution.

In this thesis we propose a model of rule-classification, called GAMoN, based on the use of Genetic Algorithms for induction of rules classification. A classifier H generated by the system GAMoN for a given category *c* takes the form of a disjunction of atoms $\mathcal{H}_c^i$ of the type:

$$\mathcal{H}_c = \mathcal{H}_c^1 \vee \cdots \vee \mathcal{H}_c^r$$

where each atom $\mathcal{H}_c^i$ is made up of the quadruple $< Pos, Neg, m_i, n_i >$, where:

- $Pos = \{t_1, .., t_n\}$ is the set of *positive* terms, ie the set of terms which are representative for the category *c* of reference;

- $Neg = \{t_{n+1}, , t_{n+m}\}$ is the set of *negative* terms, ie all the terms that are not representative for the category *c* of interest;

- m_i e n_i are integers, called *thresholds*, such that $m_i >= 0$ and $n_i > 0$.

Intuitively, the meaning of each atom $\mathcal{H}_c^i$ is the following: "classify the generic document *d* under the category *c* if at least m_i positive terms appear in *d* and less than n_i negative terms appear in *d*". Indeed, the hypothesis language introduced by GAMoN is called $MofN^+$, an extension of $MofN$ classifiers with negative terms. Hence, the acronym "GAMoN" indicates a textual classification system based on "Genetic Algorithms" of the type "M-of-N".

GAMoN arises as an extension of "Olex-GA", a classification system for the induction of rule-based text classifiers and implemented in previous research work. A classifier generated by GAMoN coincides with that of Olex-GA when m_i=1 and n_i = 1. In fact, an Olex -GA classifier has the

meaning "if at least one of the positive terms $t_1, ..., t_n$ appears in the document *d* and no negative terms $t_{n+1}, , t_{n+m}$ appears in *d*, then classify *d* under the category *c*" . Benchmarking was performed over 13 real-world text data sets (Reuters - 21578 , Ohsumed, OH5, OH0, OH10, OH15, Blogs Gender, Ohscale, 20 Newsgroups, Cade, SRAA, ODP and Market) and compared with other 5 classification systems: BioHEL [18, 48] and Olex-GA [101], which are evolutionary rule-based systems, Ripper [37] and C4.5 [105], which are not evolutionary rule-based systems and SMO, a linear SVM classifier [76].

Experimental results demonstrate that GAMoN delivers state-of-the-art classification performance, providing a good balance between accuracy and model complexity. Further, they show that GAMoN can scale up to large and realistic real-world domains better than both C4.5 and Ripper.

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;
- 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 GAMoN and giving some preliminary definitions and notation, like the language and the hypothesis space, we provide a detailed description of the crossover operators used by the system; we present the experimental results and provide a performance comparison with other learning approaches.

Contents

Part I

Text Classification

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.

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 $\bar{c_i}$. 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 [97] [114], multimedia document categorization through the analysis of textual captions [112], author identification for literary texts of unknown or disputed authorship [47], language identification for texts of unknown language [29], automated identification of text genre [77] and automated essay grading [80].

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 [116], 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, [52], [110], [125].

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 [24]. 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 [13] [42] and further classify non-junk mail into topical categories of interest to the user [34].

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. [55] [64]) 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 [63] 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. [63] 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 [95] 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 [95]. 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 [95]), 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 $\langle TV_i = \Omega \setminus Te_i, Te_i \rangle$. 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 [121], 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. 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 [79]. In their study, [104] 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 [27] and Mesh [5]. 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 [117]. 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 $\mathbf{c_i}$		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") cate-

gories, 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" [131] and thus favour macroaveraging, while others believe that topics should indeed count proportionally to their frequence, and thus lean towards microaveraging.

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 [86]. 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 [16, 83]. 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 [84], 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 [136], 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.

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 [63], 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. A full description of the REUTERS-21578 collection and a discussion of the experimentation results on its subsets can be found in [39].

3.2 OHSUMED

OHSUMED[1] is a bibliographical document collection, developed by William Hersh and colleagues at the Oregon Health Sciences University [66]. 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 [5].

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 [71]. 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.

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

Joachims [71]. 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. Other researchers used the OHSUMED collection for TC experiments, but the employed document set and categories vary: among the others, Yang in [137] 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.

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 [71] and k-NN [135], 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 [108] and Ripper [33].

- In chapter 7, 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 [54], 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 [127] 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 [135]. 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 [71, 73] and subsequently used in [42–44, 78, 122, 136]. 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 [38].
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 [117].
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.

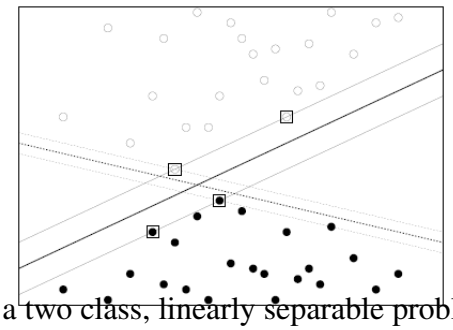

Figure 4.1: Example of a two class, linearly separable problem and two possible separation hyperplanes with corresponding margins.

As argued by Joachims in [71], 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 [70, 91, 134].

A well known example based approach is *k*-NN (for "*k* nearest neighbours") algorithm imple-

mented by Yang in the ExpNet system [135]. 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 [81] use k = 20, while Yang [134, 136] 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 [117].

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 [51]), C4.5 (used in [35, 37, 71, 85]) and C5 (used in [88]). TC efforts based on experimental DT packages include [44, 86, 130].

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.

5.1.1 C4.5 Classifier

C4.5 is a decision tree classifier that was developed by Quinlan [106]. 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 [7], there have been a remarkable number of variants and improvements of association rule mining algorithms i.e. [58].
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 [87] and CBA [89]. Although both of them proved to be effective and achieve high accuracy 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 [14].
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 term-sets in documents. Eventually, these frequent itemsets associated with text categories represent the discriminate features among the documents in the collection. The association rules discovered in this stage of the process are further processed to build the associative classifier. Using the apriori algorithm on transactions representing the documents would generate a very large number of association rules, most of them irrelevant for classification. 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) [139] 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, [14] 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 presented for categorization by employing a dominance factor (proportion of rules of the most dominant category in the applicable rules for a document to classify).

Experimental results reported in [14] 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 [14] 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 [32]. 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 [36].

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 [115] 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 [115]. 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 [25]and the *synonymy and hypernymy* relations from WordNet [27]. 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 [115].

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 [128]:

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 [128]. 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 [92]. The text-specific taxonomies, used for TRIPPER growing and pruning phases, comes from WordNet [27], 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

Evolutionary Approaches

Text Classification is the task of assigning natural language texts to one or more thematic categories on the basis of their contents. A number of machine learning methods have been proposed in the last few years, including k-nearest neighbors (k-NN), probabilistic Bayesian, neural networks and SVMs. In a different line, rule learning algorithms, have become a successful strategy for classifier induction. Rule-based classifiers provide the desirable property of being readable and, thus, easy for people to understand (and, possibly, modify). Genetic Algorithms (GA's) are stochastic search methods inspired to the biological evolution [59, 94]. Their capability to provide good solutions for classical optimization tasks has been demonstrated by various applications, including TSP [6, 75] and Knapsack [69]. Rule induction is also one of the application fields of GA's [12, 49, 99, 100]. The basic idea is that each individual encodes a candidate solution (i.e., a classification rule or a classifier), and that its fitness is evaluated in terms of predictive accuracy. In the various GA-based approaches to rule induction used in the literature (e.g., [49, 99, 100]), an individual of the population may either represent a single rule or a rule set. The former approach (single-rule-per-individual) makes the individual encoding simpler, but the fitness of an individual may not be a meaningful indicator of the quality of the rule. On the other hand, the several-rules- per-individual approach, where an individual may represent an entire classifier, requires a more sophisticated encoding of individuals, but the fitness provides a reliable indicator. So, in general, there is a tradeoff between simplicity of encoding and effectiveness of the fitness function In the following section we will analyze two GA's models, BioHEL [18, 48] and Olex-GA [101].

6.1 BIOHEL

The BioHEL (Bioinformatics-oriented Hierarchical Evolutionary Learning) system is a *rule-based* evolutionary learning system based on the Pittsburgh approach, where each individual encodes a candidate classifier. BioHEL applies an iterative rule learning (IRL) approach to evolve individuals

```
Procedure BioHEL general workflow
Input : TrainingSet
RuleSet = ∅
stop = false
Do
    BestRule = null
    For repetition=1 to NumRepetitionsRuleLearning
        CandidateRule = RunGA(TrainingSet)
        If CandidateRule is better than BestRule
            BestRule = CandidateRule
        EndIf
    EndFor
    Matched = Examples from TrainingSet matched by BestRule
    If class of BestRule is the majority class in Matched
        Remove Matched from TrainingSet
        Add BestRule to RuleSet
    Else
        stop = true
    EndIf
While stop is false
Output : RuleSet
```

Figure 6.1: BioHEL general workflow

that are a set of rules. Each set of rules represents a classifier. The learning process creates a rule set by iteratively learning one rule at a time using a GA approach. Each time the system learns a new rule, adds it to the theory and removes all covered examples from the training set. This process is repeated iteratively until all examples are covered.

Figure 6.1 contains the pseudo-code of the general workflow of BioHEL.

6.2 Olex-GA

Olex-GA is a *GA-based* approach for the induction of *rule-based* text classifiers. An Olex-GA classifier $\mathcal{H}_c$ for a category c is a pair $\langle Pos, Neg \rangle$, where $Pos = \{t_1, \cdots t_n\}$ is the set of positive terms and $Neg = \{t_(n+1), \cdots t_(n+m)\}$ the set of negative terms. The informal meaning of such a classifier is "classify document d under category c if any of the positive terms occurs in d and none of the negative terms occurs in d". The formal meaning of $\mathcal{H}_c$ is given by the following classification rules:

$$c \leftarrow (t_1 \in d \vee \cdots \vee t_n \in d) \wedge \neg(t_{n+1} \in d \vee \cdots \vee t_{n+m} \in d)$$

where c is a category, d a document and each t_i a term (n-gram) taken from a given vocabulary. We denote a classifier for c as above by $\mathcal{H}_c(Pos, Neg)$, where $Pos = \{t_1, \cdots, t_n\}$ and $Neg =$

$\{t_{n+1} \cdots t_{n+m}\}$. Positive terms in Pos are used to cover the training set of c, while negative terms in Neg are used to take precision under control.

The problem of learning $\mathcal{H}_c(Pos, Neg)$ is formulated as an optimization task (MAX-F) aimed at finding the sets Pos and Neg which maximize the F-measure when $\mathcal{H}_c(Pos, Neg)$ is applied to the training set. MAX-F can be represented as a 0-1 combinatorial problem and, thus, the GA approach turns out to be a natural candidate resolution method.

In Olex-GA an individual represents a candidate classifier (instead of a single rule). It relies on the Pittsburgh approach for (variable-length) individual encoding. The fitness of an individual is expressed in terms of the F-measure attained by the corresponding classifier when applied to the training set. This *several-rules-per-individual* approach (as opposed to the *single-rule-per-individual* approach) provides the advantage that the fitness of an individual reliably indicates its quality, as it is a measure of the predictive accuracy of the encoded classifier rather than of a single rule.

Once the population of individuals has been suitably initialized, evolution takes place by iterating elitism, selection, crossover and mutation, until a pre-defined number of generations is created.

Figure 6.2 contains the pseudo-code of the general workflow of Olex-GA for category c.

Algorithm Olex-GA

Input: vocabulary $V(f, k)$ over the training set TS; number n of generations;
Output: "best" classifier $\mathcal{H}_c(Pos, Neg)$ of c over TS;

- **begin**
- Evaluate the sets of candidate positive and negative terms from $V(f, k)$;
- Create the population $oldPop$ and initialize each chromosome;
- **Repeat** n times
- Evaluate the fitness of each chromosome in $oldPop$;
- $newPop = \emptyset$;
- Copy in $NewPop$ the best r chromosomes of $oldPop$ (elitism - r is determined on the basis of the elitism percentage)
- **While** $size(newPop) < size(oldPop)$
- select *parent1* and *parent2* in *oldPop* via roulette wheel
- generate *kid1*, *kid2* through $crossover(parent1, parent2)$
- apply mutation, i.e., $kid1 = mut(kid1)$ and $kid2 = mut(kid2)$
- apply the repair operator ρ to both $kid1$ and $kid2$;
- add *kid1* and *kid2* to *newPop*;
- **end-while**
- $oldPop = newPop$;
- **end-repeat**;
- Select the best chromosome K in $oldPop$;
- Eliminate redundancies from K;
- **return** the classifier $\mathcal{H}_c(Pos, Neg)$ associated with K.

Figure 6.2: Evolutionary Process for category c

Chapter 7

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 [54], 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.

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

Galavotti et al in [54] proposed a family of variants of Yang's version of K-NN, called *k*-$NN^p{}_{neg}$. 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

$$CSV_i(d_j) = \sum_{d'_i \in Tr_k(d_j)} RSV(d_j, d'_z) \cdot v_{iz} \tag{7.1}$$

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 [54], 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 7.2, 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:

$$CSV_i(d_j) = \sum_{d'_i \in Tr_k(d_j)} RSV(d_j, d'_z)^p \cdot v_{iz} \tag{7.2}$$

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.

7.2 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. [26] 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 [113], 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 [133], 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 [15] consider another framework that adds to the support-confidence, for the choice of positive and negative items set, some measures based on correlation analysis.

7.2.1 Mining Positive and Negative Associative Rules

The Association Rule Mining approach presented in [133] 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 [133] 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 [30], 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, [133] 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 [9] in the support-confidence framework proposed in [8]. 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 [133] 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 [133] about the introduction of negative information in rule mining.

7.2.2 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:

$$\rho = \frac{Cov(X,Y)}{\sigma_X \sigma_Y} \tag{7.3}$$

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 [15].

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 [89]. 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 [15], 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.

7.3 Use of Negative Information in Features Selection

In [140] 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 [137]). 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

GAMoN: a GA-based Approach for Learning Text Classifiers

While there has been a long history of rule-based text classifiers, to the best of our knowledge no M-of-N-based approach for text categorization has so far been proposed. In this paper we argue that M-of-N hypotheses are particularly suitable to model the text classification task because of the so called 'family resemblance" metaphor: 'the members (i.e., documents) of a family (i.e., category) share some small number of features, yet there is no common feature among all of them. Nevertheless, they resemble each other". Starting from this conjecture, we provide a sound extension of the M-of-N approach with negation and disjunction, called M-of-$N^{\{\neg,\vee\}}$, which enables to best fit the true structure of the data. Based on a thorough theoretical study, we show that the M-of-$N^{\{\neg,\vee\}}$ hypothesis space has two partial orders that form complete lattices.
GAMoN is the task-specific Genetic Algorithm (GA) which, by exploiting the lattice-based structure of the hypothesis space, efficiently induces accurate M-of-$N^{\{\neg,\vee\}}$ hypotheses.
In chapter 8, we provide a complete overview of the GAMoN language, the hypothesis space and the refinement operators. Then, we give a description of the learning process. In chapters 9 and 10 we present the experimental results and provide a comparison with other learning approaches. Benchmarking was performed over 13 real-world text data sets, by using four rule induction algorithms: two GAs, namely, BioHEL and OlexGA, and two non-evolutionary algorithms, namely, C4.5 and Ripper. Further, we included in our study linear SVM, as it is reported to be among the best methods for text categorization.

Chapter 8

GAMoN: effective rule learning for TC

In this chapter we define a task-specific Genetic Algorithm (GA), called GAMoN, relying on specialized evolutionary operators representing a stochastic implementation of the refinement operators defined over the subsumption lattices. At a glance, the following are the main characteristics of GAMoN:

- It relies on a variable-length individual representation, where each individual encodes a candidate classifier (Pittsburgh approach [65]).
- It combines the standard search strategy of GAs with *ad hoc* generalizing/specializing (GS) reproduction operators which exploit the structure of the hypothesis space.
- It dynamically adapts the probability of selecting the GS operators over the standard ones.
- It maintains a number of competing sub-populations.
- It uses the F-measure to assess the fitness of an individual.

Unlike in the classical approach, where the feature space is simply a subset of terms from the vocabulary, the one on which GAMoN builds its hypotheses consists of both a set of positive and a set of negative candidate features.

In this chapter, we provide a concise introduction of the M-of-N hypothesis (section 8.2), a complete overview of the GAMoN language (section 8.3), the language definition and the hypothesis space (section 8.4) and the refinement operators (section 8.5). In the 8.6 section we give a definition of the learning problem and show its complexity. Thus, we provide an effective algorithm for learning classifiers in the M-of-$N^{\{\neg,\vee\}}$ hypothesis space. In particular, we propose a heuristic approach based on a Genetic Algorithm (GA) (Section 8.7).

8.1 Background

Various supervised machine learning techniques have been applied to document classification. An excellent overview can be found in [118].
SVMs are a class of learning algorithms that showed to be highly accurate in many data mining tasks. In [72, 74], Joachims has investigated their application to text classification. The results of the empirical study showed that SVMs are more effective than other learning algorithms, namely, Naive Bayes, Rocchio, C4.5 and k-Nearest Neighbor. Further, linear SVM showed to perform as well as non-linear kernels, but substantially more efficiently.
Naive Bayes (NB) has been a very popular technique to classify texts due to its computational efficiency and simplicity. McCallum and Nigam [93] investigated the two main document representations for NB text classification, the Bernoulli and multinomial. They concluded that the latter is superior in accuracy in most cases. However, one problem with Multinomial NB (MNB) is that, when one class has more training examples than another, it selects poor weights for the decision boundary. One additional problem is that MNB does not model text well. To improve the performance of MNB, Rennie *et al.* [109] proposed Complement Naive Bayes (CNB). While learning the conditional probability of one class, CNB uses the frequency information pertaining to all other classes (that is, uses negative information).
In a different view, rule learning algorithms have become a successful strategy for classifier induction. Direct methods extract rules directly from data, while indirect methods extract rules from other classification models, such as decision trees (e.g., C4.5 [105]). Representative examples of direct methods include Inductive Rule Learning (IRL) systems, such as FOIL [107] and Ripper [37], and Associative Rule Learning (ARL) systems, such as CMAR [62], CPAR [138] and TFPC [31].
A sub-class of the inductive rule learners is that of Genetics-Based Machine Learning algorithms (GBML) [45], which rely on the Evolutionary Algorithms as search mechanisms. Examples of such systems are XCS [132], SIA [129], GAssist [19] and BioHEL [18, 48]. Many GBML systems have explicit generalisation/specialisation operators [20, 28, 41, 56, 57, 90].
The most well known rule-based classifiers used to learn from texts, notably, Ripper and C4.5, actually originate from non-text data mining (see, e.g., [37, 53, 72]). Among the few examples of rule-based systems specifically designed to classify texts, we mention the associative classifier NeW [23] and the IRL systems Olex [111] and OlexGA [101]. Olex induces rules consisting of one positive conjunction and (zero or) more negative conjunctions. It relies on a search technique that greedily selects at each step the conjunct, either positive or negative, that maximizes the F-measure over the training set. OlexGA is a GBML which is a special case of GAMoN, where a classifier is a M-of-$N^{\{\neg,\vee\}}$ atom with thresholds $p = 1$ and $n = 1$[1]. A peculiarity of such systems is that of explicitly dealing with negated features.

[1]The Olex and OlexGA suite is downloadable from `http://www.mat.unical.it/OlexGA`

Even if prior studies found SVMs and Complement Naive Bayes to be particularly effective for text categorization, rule-based text classifiers are often preferred in real-world applications as they provide interpretable models. Readability is indeed a very desirable property of classification models, which allows a human being to understand and possibly modify them based on his a-priori knowledge.
However, one drawback with most rule-based systems is the high computational cost, especially on high dimensional data sets. In ARL systems, the time cost for frequent pattern mining may increase very sharply when the size of data set grows. In addition, the high number of rules generated usually requires an additional pruning step where redundant rules are discarded. Also IRL systems typically rely on a two-stage process: a greedy heuristic constructs an initial rule set and, then, one or more optimization phases improve compactness and accuracy of the rule set (a similar approach is used for decision tree as well). All this makes it difficult for most rule induction methods to scale up to large and realistic real-world data sets.

8.2 The M-of-N hypothesis

A M-of-N hypothesis, also called Boolean threshold function, may be thought of intuitively as follows. Given a set of N features, whenever an example satisfies at least M of such features, it is a positive example; otherwise, it is a negative one. That is, a M-of-N hypothesis is a description that involves "counting properties". There is quite a literature on methods for building M-of-N hypotheses. For instance, in [119] algorithms for extracting M-of-N hypotheses from neural networks are reported. M-of-N concepts are also constructed as tests for the induction of decision trees [82, 96, 120, 124, 141].
However, to the best of our knowledge, no M-of-N-based approach for text classification has been so far proposed. Despite this, we conjecture that M-of-N hypotheses are well suited to model the text classification task.
Text categorization (TC) is aimed at assigning natural language texts to one or more thematic categories on the basis of their contents. It is a difficult task essentially because of two main factors: on one hand, TC has to do with the complexity and richness of the natural language, which allows a concept to be expressed by a variety of constructs and words. This aspect is often amplified by the presence in a category of documents which are not about a single narrow subject with limited vocabulary. On the other hand, the TC task deals with highly dimensional data sets (i.e., with many features). Both such factors concur to make quite unlikely the existence of a set of features, or even a single feature, that occur in all documents of a given category. It may even happen that documents that belong to the same category do not share any content words. However, as argued in [74], the relationship of "family resemblance" holds. That is, documents under the same category share a (usually small) set of N features, yet this set is not present in every document. Instead, each doc-

ument contains $M \leq N$ of such features, and different documents may not share features at all. That is, the text classification task deals with the kind of data that M-of-N hypotheses are able to explain.

A shortcoming of the M-of-N approach, however, is that its propositions handle positive information only, whereas negative evidence is deemed to play a crucial role in text categorization. This is mainly because natural languages are intrinsically ambiguous, and negation helps to disambiguate concepts - e.g., the word "ball" may ambiguously refer to either the concept "sport" or "dance", whereas the conjunction "ball and not ballroom" much likely refers to "sport".

To overcome this drawback, we extend classical M-of-N hypotheses by negation. In addition, to best fit the true structure of the data, we allow disjunctions of hypotheses. That is, we define a new hypothesis language for text classification, called M-of-$N^{\{\neg,\vee\}}$, which generalizes the classical M-of-N language through negation and disjunction (a preliminary description of the proposed approach can be found in [103]).

In our approach, a classifier is a propositional formula of the form $\mathcal{H}_c = \mathcal{H}_c^1 \vee \cdots \vee \mathcal{H}_c^r$, where each $\mathcal{H}_c^i = p_i\text{-}of\text{-}Pos \wedge \neg n_i\text{-}of\text{-}Neg$ is an *atom* (note that all atoms forming $\mathcal{H}_c$ share the *same* sets Pos and Neg.). Here, Pos is the set of *positive* terms, Neg the set of *negative* terms, and $p_i \geq 0$ and $n_i > 0$ are integers called *thresholds*. The meaning of an atom $\mathcal{H}_c^i$ is: classify document d under category c if *at least* p_i positive terms occur in d and (strictly) *less* than n_i negative terms occur in d. That is, M-of-$N^{\{\neg,\vee\}}$ provides support for explicitly modeling the interactions between positive and negative features. Of course, $\mathcal{H}_c$ classifies document d under c if any of $\mathcal{H}_c^1, \cdots, \mathcal{H}_c^r$ classifies d under c.

The special case of M-of-$N^{\{\neg,\vee\}}$where a hypothesis is an atom with thresholds $p = n = 1$ is OlexGA [101].

There is a natural ordering in the space of M-of-$N^{\{\neg,\vee\}}$ hypotheses determined by two kinds of subsumption relationships: the *feature* and the *threshold* relationships. The feature relationship is determined by the feature sets Pos and Neg appearing in a classifier $\mathcal{H}_c$. As an example, assume that $\mathcal{H}_c$ is the atomic classifier $p\text{-}of\text{-}Pos \wedge \neg n\text{-}of\text{-}Neg$, with $p = 2$ and $n = 1$. Clearly, the larger Pos, the higher the probability that the condition "at least two positive features occur in a document" is satisfied. Dually, the smaller Neg, the more likely a document will contain no negative feature in Neg. In summary, the larger Pos, the smaller Neg, the more general $\mathcal{H}_c$. The threshold relationship, in turn, is determined by the thresholds appearing in $\mathcal{H}_c$. For an instance, if in the above classifier we replace $p = 2$ by $p = 1$, we get a new classifier which is more general than the previous one - intuitively, only one instead of two positive features is necessary for classifying a document. These relationships define two hierarchies of hypotheses (more precisely, complete lattices) exploitable for an effective exploration of the hypothesis space. To this end, we provide suitable refinement operators whereby "navigating" the hypothesis lattices.

As argued in [82], the evolutionary approach seems to be particularly suited for the M-of-N learn-

ing task, as the global search style of GAs (as opposed to the "one-attribute-at-a-time" of the greedy approach) makes them capable of catching the hidden interactions among attributes that strongly characterize the induction of M-of-N hypotheses. However, the purely non-deterministic nature of conventional genetic operators does not enable the search strategy to benefit of the structure of the hypothesis space. To overcome this drawback, we define a *task-specific* Genetic Algorithm (GA), called GAMoN, relying on specialized evolutionary operators representing a stochastic implementation of the refinement operators defined over the subsumption lattices. At a glance, the following are the main characteristics of GAMoN:

- It relies on a variable-length individual representation, where each individual encodes a candidate classifier (Pittsburgh approach [65]).
- It combines the standard search strategy of GAs with *ad hoc* generalizing/specializing (GS) reproduction operators which exploit the structure of the hypothesis space.
- It dynamically adapts the probability of selecting the GS operators over the standard ones.
- It maintains a number of competing sub-populations.
- It uses the F-measure to assess the fitness of an individual.

Unlike in the classical approach, where the feature space is simply a subset of terms from the vocabulary, the one on which GAMoN builds its hypotheses consists of both a set of positive and a set of negative candidate features. One main issue that in general arises when inducing a classifier is that of selecting the appropriate dimensionality of the feature space, i.e., how many features the classifier can access during the learning process. This is a very important design choice, as the quality of the selected features strongly determines the quality of the learned classifier, especially in text classification, where data sets are usually highly dimensional, noisy and ambiguous. For most systems, the size of the feature space is managed as a tuning parameter, that is, the learning process is rerun over feature spaces of different dimensions and the best results are eventually taken. Unfortunately, this may require very long training times, especially over large data sets. To get over this inconvenience, GAMoN was provided with techniques to automatically detect an appropriate dimensionality of the feature space. This way, no manual feature selection is preliminarily needed. GAMoN was designed as a binary classification system. We use the "one-vs-all" approach to produce one (independent) model for each class in a multi-class classification task (this technique is frequently used in multi-label classification, where each example may have more than one label - as it is the case in text classification).

8.3 Language Overview

The M-of-$N^{\{\neg,\vee\}}$ representation generalizes the classical notion of M-of-N concepts by allowing negation and disjunction. A M-of-$N^{\{\neg,\vee\}}$ classifier for category c is a propositional formula of the form $\mathcal{H}_c = \mathcal{H}_c^1 \vee \cdots \vee \mathcal{H}_c^r$, where each $\mathcal{H}_c^i = p_i\text{-}of\text{-}Pos \wedge \neg n_i\text{-}of\text{-}Neg$ is an *atom* expressing the following condition: classify document d under category c if *at least* p_i positive features in Pos and *less* than n_i negative features in Neg occur in d. Integers $p_i \geq 0$ and $n_i > 0$ are called *thresholds*. Of course, $\mathcal{H}_c$ classifies document d under c if any among $\mathcal{H}_c^1, \cdots, \mathcal{H}_c^r$ classifies d under c.
Since all atoms forming $\mathcal{H}_c = \mathcal{H}_c^1 \vee \cdots \vee \mathcal{H}_c^r$ share the same sets of features Pos and Neg, a convenient notation for $\mathcal{H}_c$ is $\langle Pos, Neg, \mathcal{T}\rangle$, where $\mathcal{T} = \{(p_1, n_1), \cdots, (p_r, n_r)\}$ is the set of threshold pairs appearing in the atoms of $\mathcal{H}_c$ ($\mathcal{T}$ is called *threshold set*). For example, $(1\text{-}of\text{-}Pos \wedge \neg 2\text{-}of\text{-}Neg) \vee (2\text{-}of\text{-}Pos \wedge \neg 3\text{-}of\text{-}Neg)$ can be simpler represented as $\langle Pos, Neg, \{(1,2),(2,3)\}\rangle$.
As a concrete example, consider the classifier constructed by GAMoN for category "grain" from the Reuters data set:

$$\mathcal{H}_{grain} = \langle Pos = \{barley, cereals, corn, grain, maize, rice, sorghum, wheat\},$$

$$Neg = \{acquisition, bank, earning, pay, profit, tax, york\},$$

$$\mathcal{T} = \{(1,1),(2,2)\}\rangle$$

This is a classifier of order 2 (as its threshold set has two elements, i.e., $(1,1)$ and $(2,2)$), with 8 positive features (barley, cereals, etc.) and 7 negative ones (acquisition, bank, etc.). The meaning of $\mathcal{H}_{grain}$ is the following: classify document d under category "grain" if either one of the following conditions hold: (1) d contains (exactly) one positive feature and no negative features, or (2) d contains more than one positive feature and less than two negative ones. That is to say, one single positive feature has no effect on predicting the category "grain" if any negative feature occurs in d, while one single negative feature has no effect in denying the classification of d if more positive features occur in d.
As the above example shows, one beneficial aspect of the M-of-$N^{\{\neg,\vee\}}$ representation is readability. This is a very important feature, as it makes possible for people to visually inspecting and understanding the induced model.
The M-of-$N^{\{\neg,\vee\}}$ hypothesis space has a structure determined by two kinds of subsumption relationships: the *feature* and the *threshold* subsumptions.
Intuitively, positive features are indicative of membership for a category, contrary to negative ones that are indicative of non-membership. Thus, the more elements are in Pos, the less are in Neg, the more general a classifier $\langle Pos, Neg, \mathcal{T}\rangle$ is (i.e., it classifies more documents). Feature subsumption encodes this intuition. As an example, $\langle\{t_0, t_1\}, \{t_3, t_4\}, \mathcal{T}\rangle$ subsumes $\langle\{t_0, \}, \{t_3, t_4\}, \mathcal{T}\rangle$ and is subsumed by $\langle\{t_0, t_1\}, \{t_3\}, \mathcal{T}\rangle$.

The threshold subsumption relationship is in turn determined by the threshold sets appearing in the classifiers. For an instance, the hypothesis $\langle Pos,\ Neg,\ \{(1,1)\}\rangle$ subsumes $\langle Pos,\ Neg,\ \{(2,1)\}\rangle$ as only one, instead of two positive features, is necessary for it to classify a document.
Thus, both the above hierarchies capture the intuitive notion of general-to-specific ordering, that is, if $\mathcal{H}_c$ subsumes $\mathcal{H}'_c$ in either hierarchy, then whatever is classified by $\mathcal{H}'_c$ is classified by $\mathcal{H}_c$ as well. One interesting property of such relationships is that they form complete lattices in the hypothesis space (thus, any hypothesis can be reached in the search space).
We can take advantage of this general-to-specific ordering in order to selectively search the hypothesis space. For an instance, if the classifier $\langle Pos, Neg, \{(2,1)\}\rangle$ is too specific (i.e., it covers too few positive examples) it can be generalized either (i) through the threshold subsumption, by replacing the threshold set $\{(2,1)\}$ by one less restrictive, say, $\{(1,1)\}$, or (ii) by the feature subsumption, i.e., by adding some term to Pos or removing some term from Neg.
Another way of generalizing or specializing a hypothesis is by "interaction" with another one. To this end, we exploit the lattice structure of the hypothesis space. That is, the least upper bound (resp. greatest lower bound) of two hypotheses can be taken, in any of the two lattices, in order to get a more general (resp. specific) one. As an example, given two hypotheses sharing the same threshold sets, say, $\langle\{t_0,t_1\},\{t_3\},\{(2,1)\}\rangle$ and $\langle\{t_0,t_4\},\{t_5\},\{(2,1)\}\rangle$, we can specialize both by taking the greatest lower bound in the feature subsumption lattice, that is, $\langle\{t_0\},\{t_3,t_5\},\{(2,1)\}\rangle$ - a classifier whose sets of positive and negative features are $\{t_0\} = \{t_0,t_1\} \cap \{t_0,t_4\}$ and $\{t_3,t_5\} = \{t_3\} \cup \{t_5\}$, respectively. Likewise, given two hypotheses sharing the same feature sets, say, $\langle Pos, Neg, \{(1,1)\}\rangle$ and $\langle Pos, Neg, \{(2,2)\}\rangle$, we can specialize both by taking the greatest lower bound in the threshold subsumption lattice, that is, $\langle Pos, Neg, \{(2,1)\}\rangle$ - a classifier whose threshold set is $\{max(1,2), min(1,2)\}$ (see Figure 8.1). It can be easily verified that both greatest lower bounds are more specific than the respective parents.
As we will see later on this paper, the above concepts are at the basis of the definition of the *refinement operators*. These are the abstract tools for searching the hypothesis space, that find concrete application in the definition of the reproduction operators of GAMoN.

8.4 Language Definition and Hypothesis Space

Now that we have an intuitive view of the basic ideas, in the next subsections we will provide formal definitions of them. In particular, we will start from the notion of *feature space*, i.e., the set of features which provide the lexicon from which hypotheses are built. Then we formalize the M-of-$N^{\{\neg,\vee\}}$language and define the feature $\succeq_\phi$ and the threshold $\succeq_\tau$ subsumption relationships, showing a number of interesting properties. In particular, we will prove that they form complete lattices in the hypothesis space and provide a constructive definition of the meet and the join operators in both lattices. Finally, we will give a deep insight into the structure of the hypothesis space, and show the notion of *decision boundary* for M-of-$N^{\{\neg,\vee\}}$ classifiers.

8.4.1 Feature Space

We are given a set T of training documents (also called "examples") and a set $\mathcal{C}$ of categories (also called "concepts"). A document is a set of features (also called "terms"), a feature being a sequence of one or more words (or word stems). Each document in T is associated with a category in $\mathcal{C}$. We denote by $T_c \subseteq T$ the training set of c, i.e., the set of training documents associated with category c. We call *vocabulary* the set of features occurring in the documents of T.
Unlike in the classical definitions, where the feature space is simply a subset of the vocabulary, in our definition the feature space consists of both a set of positive and a set of negative features. This is because M-of-$N^{\{\neg,\vee\}}$ hypotheses explicitly models the interaction between positive and negative features, the latter being regarded as "first class citizens".

Definition 8.1 (Feature space) We are given a vocabulary V, a non-negative integer k and a scoring function σ which assigns a score to every feature in V based on its correlation with category c (e.g., CHI Square [46]) . Define the *feature space* $\mathcal{F}(k)$ (of size k) for category c as the pair $\langle Pos_c^*(k), Neg_c^*(k)\rangle$, where $Pos_c^*(k) \subseteq V$ and $Neg_c^*(k) \subseteq V$ are as follows:

- $Pos_c^*(k)$ is the set of the k highest scoring features in V for category c, according to σ; we say that $t \in Pos_c^*(k)$ is a *candidate positive feature* of c.
- given $Pos^*(k)$, consider the set N of terms co-occurring with positive candidate features within negative examples, i.e.,
  $N = \{t \in V \mid t \notin Pos_c^*(k)\ and\ (\Theta^+ \cap \Theta(t) \setminus T_c) \neq \emptyset\}$ where $\Theta(t) \subseteq T$ is the set of training documents containing feature t, $\Theta^+ = \cup_{t \in Pos_c^*(k)} \Theta(t)$ and T_c is the training set of c. With each feature $t \in N$ we assign a score $\eta(t)$ as follows:

$$\eta(t) = \frac{|\Theta^+ \cap \Theta(t) \setminus T_c|}{|\Theta^+ \setminus T_c| + |\Theta(t) \cap T_c|}.$$

It can be easily seen that $0 < \eta(t) \leq 1$. In particular, a term t occurring in all negative examples and in no positive one containing any positive feature has score $\eta(t) = 1$. On the other hand, $\eta(t) > 0, \forall t \in N$ as, by definition, t co-occurs with a candidate positive feature in some negative example. Then, we define $Neg_c^*(k)$ as the set of the best k elements of N according to η; we say that $t \in Neg_c^*(k)$ is a *candidate negative feature* of c.

The rationale behind the above definition is rather intuitive: candidate positive features are supposed to capture most of the positive examples, as they are characterized by high scoring values. On the contrary, candidate negative features, defined as terms co-occurring with positive candidate terms within negative examples, are supposed to discard most of the (potentially) false positive examples. For an instance, if the feature "ball" is a candidate positive and "ballroom" co-occurs with "ball" within some negative examples, "ballroom" becomes a negative candidate feature. Clearly, the higher the scoring $\sigma(t)$ (resp. $\eta(t)$) of a term t, the higher its value as a candidate positive (resp. negative) feature.

8.4.2 Hypothesis Space

A hypothesis is a propositional formula used to describe the examples of a given concept. The hypothesis language we propose in this section is an extension of the M-of-N language, called M-of-$N^{\{\neg,\vee\}}$.

Definition 8.2 (Hypothesis language) We are given the feature space $\mathcal{F}(k) = \langle Pos_c^*(k), Neg_c^*(k)\rangle$, along with two integers, P and N, called *threshold bounds*. A M-of-$N^{\{\neg,\vee\}}$ *hypothesis* (or "classifier") for category c over $\mathcal{F}(k)$ is inductively defined as follows:

- Basis: $p\text{-}of\text{-}Pos \wedge \neg n\text{-}of\text{-}Neg$ is an *atom* (or *1-order* classifier), where $0 \leq p \leq P$ and $0 < n \leq N$ are integers called *positive* and *negative thresholds*, respectively, and $Pos \subseteq Pos_c^*(k)$ and $Neg \subseteq Neg_c^*(k)$ are (possibly empty) sets of features. In particular, Pos is the set of *positive* features and Neg the set of *negative* features. This classifier classifies a document d under category c if *at least* p positive features occur in d and *less* than n negative features occur in d. A convenient notation for $p\text{-}of\text{-}Pos \wedge \neg n\text{-}of\text{-}Neg$ is $\langle Pos, Neg, \{(p, n)\}\rangle$.

- induction: let $\mathcal{H}_c^1 = \langle Pos, Neg, \mathcal{T}_1\rangle$ be a r-order classifier and $\mathcal{H}_c^2 = \langle Pos, Neg, \mathcal{T}_2\rangle$ a s-order classifier (note that $\mathcal{H}_c^1$ and $\mathcal{H}_c^2$ share the same sets of features). Then $\mathcal{H}_c = \mathcal{H}_c^1 \vee \mathcal{H}_c^2$ is a classifier of order $q \leq r + s$. $\mathcal{H}_c$ classifies document d under c if either $\mathcal{H}_c^1$ or $\mathcal{H}_c^2$ classifies d under c. A convenient notation for $\mathcal{H}_c$ is $\langle Pos, Neg, \mathcal{T}\rangle$, where $\mathcal{T} = \mathcal{T}_1 \cup \mathcal{T}_2$.

As already noticed, all atoms forming a classifier hold the same sets Pos and Neg. That is why we can denote $\mathcal{H}_c = \mathcal{H}_c^1 \vee \cdots \vee \mathcal{H}_c^r$ by using the compact notation $\langle Pos, Neg, \mathcal{T}\rangle$, where $\mathcal{T} = \{(p_1, n_1),$

$\cdots, (p_r, n_r)\}$ is the set of all threshold pairs appearing in the atoms of $\mathcal{H}_c$. Clearly, the size of $\mathcal{T}$ is the order of the classifier.

Example 8.3 The 2-order classifier $(1\text{-}of\text{-}Pos \wedge \neg 2\text{-}of\text{-}Neg) \vee (2\text{-}of\text{-}Pos \wedge \neg 3\text{-}of\text{-}Neg)$ can be represented as $\langle Pos, Neg, \{(1,2),(2,3)\}\rangle$.

An atom with $p = 0$ and $n > |Neg|$ acts as an acceptor, while one with $p > |Pos|$ is to be understood as a rejector. An atom $\langle Pos, Neg, \{(1,1)\}\rangle$ coincides with an OlexGA classifier [101]. In general, a (non-acceptor, non-rejector) atom $\langle Pos, Neg, \{(p,n)\}\rangle$ is logically equivalent to the following propositional formula:

$$c \leftarrow (T_1 \vee \cdots \vee T_k) \wedge \neg(T_{k+1} \vee \cdots \vee T_{k+m})$$

where $T_1 \cdots T_k$ are all possible conjunctions made of p positive terms in Pos, and $T_{k+1} \cdots T_{k+m}$ are all possible conjunctions made of n negative terms in Neg. The rule-based semantics of a r-order classifier $\mathcal{H}_c = \mathcal{H}_c^1 \vee \cdots \vee \mathcal{H}_c^r$ is the obvious generalization of the base case: $\mathcal{H}_c$ is equivalent to the union of the rule sets of all $\mathcal{H}_c^i$, $1 \leq i \leq r$.
We finally provide the definition of hypothesis space.

Definition 8.4 The *hypothesis space* $\mathbf{H}(\mathcal{F}(k), P, N)$ is the set of all hypotheses constructible over a feature space $\mathcal{F}(k)$ and for given thresholds bounds P and N.

8.4.3 Ordering the hypothesis space

There is a natural ordering in the hypothesis space determined by two kinds of subsumption relationships, namely, the *feature* and the *threshold* subsumption.

Ordering along the feature dimension

Let $\mathbf{H}_{\mathcal{T}}(\mathcal{F}(k)) \subseteq \mathbf{H}(\mathcal{F}(k), P, N)$ be the hypothesis subspace consisting of all hypotheses in $\mathbf{H}(\mathcal{F}(k), P, N)$ having the same given threshold set $\mathcal{T}$. Hypotheses in $\mathbf{H}_{\mathcal{T}}(\mathcal{F}(k))$ are said *τ-homogeneous*. On $\mathbf{H}_{\mathcal{T}}(\mathcal{F}(k))$ there exists a binary relation that we call *feature-subsumption* (*ϕ-subsumption*, for short).

Definition 8.5 (Feature-subsumption) We are given two classifiers in $\mathbf{H}_{\mathcal{T}}(\mathcal{F}(k))$, say, $\mathcal{H}_c^1 = \langle Pos_1, Neg_1, \mathcal{T}\rangle$ and $\mathcal{H}_c^2 = \langle Pos_2, Neg_2, \mathcal{T}\rangle$. $\mathcal{H}_c^1$ *ϕ-subsumes* $\mathcal{H}_c^2$ (and $\mathcal{H}_c^2$ *is ϕ-subsumed by* $\mathcal{H}_c^1$) if both $Pos_2 \subseteq Pos_1$ and $Neg_1 \subseteq Neg_2$ (write $\mathcal{H}_c^1 \succeq_\phi \mathcal{H}_c^2$). $\mathcal{H}_c^1$ is called a *ϕ-generalization* of $\mathcal{H}_c^2$ (and $\mathcal{H}_c^2$ a *ϕ-specialization* of $\mathcal{H}_c^1$).

Example 8.6 According to the definition 8.5, the following classifier $\mathcal{H}_c = \langle\{t_0, t_1\}, \{t_3, t_4\}, \{(1,1)\}\rangle$ ϕ-subsumes $\mathcal{H}'_c = \langle\{t_0\}, \{t_3, t_4\}, \{(1,1)\}\rangle$ and is ϕ-subsumed by $\mathcal{H}''_c = \langle\{t_0, t_1\}, \{t_4\}, \{(1,1)\}\rangle$. Intuitively, the former subsumption holds as the classification condition on the positive features "at least one of t_0 and t_1 must occur in d" is clearly weaker than "t_0 must occur in d", so as (*ceteris paribus*) more documents will be classifier by $\mathcal{H}_c$ than by $\mathcal{H}'_c$. Dually, $\mathcal{H}_c$ is ϕ-subsumed by $\mathcal{H}''_c$ as the condition on the negative features expressed by the latter "t_4 must not occur in d" is weaker than that expressed by the former "neither t_3 nor t_4 can occur in d".

Next we show that if $\mathcal{H}^1_c \succeq_\phi \mathcal{H}^2_c$ then $\mathcal{H}^1_c$ classifies all documents classified by $\mathcal{H}^2_c$. In the following, we will denote by $\mathcal{D}(\mathcal{H}_c) \subseteq \mathcal{D}$ the set of documents classified by $\mathcal{H}_c$, for a document set $\mathcal{D}$.

Proposition 8.7 Let $\mathcal{H}^1_c$ and $\mathcal{H}^2_c$ be two classifiers in $\mathbf{H}_\mathcal{T}(\mathcal{F}(k))$. Then $\mathcal{H}^1_c \succeq_\phi \mathcal{H}^2_c$ implies $\mathcal{D}(\mathcal{H}^1_c) \supseteq \mathcal{D}(\mathcal{H}^2_c)$

The following proposition shows that $(\mathbf{H}_\mathcal{T}(\mathcal{F}(k)), \succeq_\phi)$ is a complete lattice.

Proof Let $\mathcal{H}^1_c$ and $\mathcal{H}^2_c$ be two classifiers in $\mathbf{H}_\mathcal{T}(\mathcal{F}(k))$. Next we show that $\mathcal{H}^1_c \succeq_\phi \mathcal{H}^2_c$ implies $\mathcal{D}(\mathcal{H}^1_c) \supseteq \mathcal{D}(\mathcal{H}^2_c)$. The proof proceeds by induction. (*Basis*) $\mathcal{H}^1_c$ and $\mathcal{H}^2_c$ are atoms in $\mathbf{H}_\mathcal{T}(\mathcal{F}(k))$ of the form, say, $\langle Pos_1, Neg_1, \{(p,n)\}\rangle$ and $\langle Pos_2, Neg_2, \{(p,n)\}\rangle$. A document d is classified by $\mathcal{H}^2_c$ iff $|d \cap Pos_2| \geq p$ and $|d \cap Neg_2| \leq n$ (recall that a document is a set of features - see Subsection 8.4.1). It can be easily seen that, since both $Pos_2 \subseteq Pos_1$ and $Neg_1 \subseteq Neg_2$ hold by hypothesis, $|d \cap Pos_1| \geq p$ and $|d \cap Neg_1| \leq n$ is verified as well, that is, d is classified by $\mathcal{H}^1_c$. (*Inductive step*) $\mathcal{H}^1_c$ and $\mathcal{H}^2_c$ are two generic classifiers $\langle Pos_1, Neg_1, \mathcal{T}\rangle$ and $\langle Pos_2, Neg_2, \mathcal{T}\rangle$ in $\mathbf{H}_\mathcal{T}(\mathcal{F}(k))$ such that $\mathcal{H}^1_c \succeq_\phi \mathcal{H}^2_c$. Thus, they can be expressed in the following form: $\mathcal{H}^1_c = \mathcal{H}^{1,1}_c \vee \mathcal{H}^{1,2}_c$ and $\mathcal{H}^2_c = \mathcal{H}^{2,1}_c \vee \mathcal{H}^{2,2}_c$, where $\mathcal{H}^{1,1}_c = \langle Pos_1, Neg_1, \mathcal{T}_1\rangle$, $\mathcal{H}^{1,2}_c = \langle Pos_1, Neg_1, \mathcal{T}_2\rangle$, $\mathcal{H}^{2,1}_c = \langle Pos_2, Neg_2, \mathcal{T}_1\rangle$ and $\mathcal{H}^{2,2}_c = \langle Pos_2, Neg_2, \mathcal{T}_2\rangle$. By inductive hypothesis, since both $Pos_2 \subseteq Pos_1$ and $Neg_1 \subseteq Neg_2$ hold, any document classified by $\mathcal{H}^{2,1}_c$ is classified by $\mathcal{H}^{1,1}_c$ and any document classified by $\mathcal{H}^{2,2}_c$ is classified by $\mathcal{H}^{1,2}_c$. It turns out that $\mathcal{H}^1_c$ classifies all documents classified by $\mathcal{H}^2_c$, i.e., $\mathcal{D}(\mathcal{H}^1_c) \supseteq \mathcal{D}(\mathcal{H}^2_c)$. ∎

Proposition 8.8 $(\mathbf{H}_\mathcal{T}(\mathcal{F}(k)), \succeq_\phi)$ is a complete lattice. Indeed, for any $\mathcal{H}^1_c,\ \mathcal{H}^2_c \in \mathbf{H}_\mathcal{T}(\mathcal{F}(k))$, there are both the greatest lower bound $glb_\phi(\mathcal{H}^1_c, \mathcal{H}^2_c)$ and the least upper bound $lub_\phi(\mathcal{H}^1_c, \mathcal{H}^2_c)$ as follows:

a) $lub_\phi(\mathcal{H}^1_c, \mathcal{H}^2_c) = \langle Pos_1 \cup Pos_2, Neg_1 \cap Neg_2, \mathcal{T}\rangle$.

b) $glb_\phi(\mathcal{H}^1_c, \mathcal{H}^2_c) = \langle Pos_1 \cap Pos_2, Neg_1 \cup Neg_2, \mathcal{T}\rangle$.

It is easy to recognize that the bottom element of $\mathbf{H}_{\mathcal{T}}(\mathcal{F}(k)$ is $\langle \emptyset, Neg^*(k), \mathcal{T}\rangle$ and the top $\langle Pos^*(k), \emptyset, \mathcal{T}\rangle$.

Proof Next we show that $(\mathbf{H}_{\mathcal{T}}(\mathcal{F}(k)), \succeq_\phi)$ is a complete lattice. To this end, we first prove statement (a) - $lub_\phi(\mathcal{H}_c^1, \mathcal{H}_c^2) = \langle Pos_1 \cup Pos_2, Neg_1 \cap Neg_2, \mathcal{T}\rangle$. From Definition 8.5 it immediately follows that both $lub_\phi(\mathcal{H}_c^1, \mathcal{H}_c^2) \succeq_\phi \mathcal{H}_c^1$ and $lub_\phi(\mathcal{H}_c^1, \mathcal{H}_c^2) \succeq_\phi \mathcal{H}_c^2$ hold. Now let us assume, by absurd, the existence of $\mathcal{H}_c' = \langle Pos, Neg, \mathcal{T}\rangle$ such that $lub_\phi(\mathcal{H}_c^1, \mathcal{H}_c^2) \succeq_\phi H_c'$ and, further, $H_c' \succeq_\phi \mathcal{H}_c^1$ and $H_c' \succeq_\phi \mathcal{H}_c^2$. From $lub_\phi(\mathcal{H}_c^1, \mathcal{H}_c^2) \succeq_\phi H_c'$ we have that $Pos \subseteq Pos_1 \cup Pos_2$ (see Definition 8.5). However, if so, the conditions $H_c' \succeq_\phi \mathcal{H}_c^1$ and $H_c' \succeq_\phi \mathcal{H}_c^2$ cannot hold, as $Pos_1 \subseteq Pos$ and $Pos_2 \subseteq Pos$ cannot be both true (a contradiction). From which statement (a) follows. Now we prove statement (b) - $glb_\phi(\mathcal{H}_c^1, \mathcal{H}_c^2) = \langle Pos_1 \cap Pos_2, Neg_1 \cup Neg_2, \mathcal{T}\rangle$. By Definition 8.5 we have that $\mathcal{H}_c^1 \succeq_\phi glb_\phi$ and $\mathcal{H}_c^2 \succeq_\phi glb_\phi$. Now let us assume, by absurd, the existence of $\mathcal{H}_c' = \langle Pos, Neg, \mathcal{T}\rangle$ such that $H_c' \succeq_\phi glb_\phi(\mathcal{H}_c^1, \mathcal{H}_c^2)$ and, further, $\mathcal{H}_c^1 \succeq_\phi H_c'$ and $\mathcal{H}_c^2 \succeq_\phi H_c'$. From $H_c' \succeq_\phi glb_\phi(\mathcal{H}_c^1, \mathcal{H}_c^2)$ it turns out that $Pos_1 \cap Pos_2 \subseteq Pos$. But, if so, the conditions $\mathcal{H}_c^1 \succeq_\phi H_c'$ and $\mathcal{H}_c^2 \succeq_\phi H_c'$ cannot hold, as Pos is not a subset of both Pos_1 and Pos_2 (a contradiction). From which statement (b) follows. ∎

Ordering along the threshold dimension

Let $\mathbf{H}_\Phi(P, N) \subseteq \mathbf{H}(\mathcal{F}(k), P, N)$ be the hypothesis subspace consisting of all hypotheses having the same $\Phi = \langle Pos, Neg\rangle$, with Pos and Neg over $\mathcal{F}(k)$. We say that two classifiers in $\mathbf{H}_\Phi(P, N)$ are *ϕ-homogeneous*. Next we show that a subsumption hierarchy there exists in $\mathbf{H}_\Phi(P, N)$. We call it *threshold-subsumption*, or *τ-subsumption*, for short.

Notation. Since *ϕ-homogeneous* hypotheses share all the same feature sets, in the following, whenever no ambiguity arises, we shall represent a classifier $\langle Pos, Neg, \mathcal{T}\rangle$ simply by $\mathcal{T}$.

Example 8.9 The 2-order classifier $\langle Pos, Neg, \{(1, 2), (2, 3)\}\rangle$ may be represented simply as $\{(1, 2), (2, 3)\}$.

Definition 8.10 (Threshold-subsumption) Let $\mathcal{T}_1 = \{(p_1, n_1)\}$ and $\mathcal{T}_2 = \{(p_2, n_2)\}$ be two threshold sets (of size 1). Then $\mathcal{T}_1$ *τ-subsumes* $\mathcal{T}_2$, denoted $\mathcal{T}_1 \succeq_\tau \mathcal{T}_2$, if both $p_1 \leq p_2$ and $n_1 \geq n_2$ hold. More in general, given two threshold sets (of any size), we say that $\mathcal{T}_1$ *τ-subsumes* $\mathcal{T}_2$ if, for each element $(p, n) \in \mathcal{T}_2$, there exists an element $(p', n') \in \mathcal{T}_1$ such that $\{(p', n')\} \succeq_\tau \{(p, n)\}$.

The relation $\succeq_\tau$ induces a relation on $\mathbf{H}_\Phi(P, N)$ as follows. Given $\mathcal{H}_c^1 = \langle Pos, Neg, \mathcal{T}_1\rangle$ and $\mathcal{H}_c^2 = \langle Pos, Neg, \mathcal{T}_2\rangle$ in $\mathbf{H}_\Phi(P, N)$, $\mathcal{H}_c^1$ *τ-subsumes* $\mathcal{H}_c^2$ (and $\mathcal{H}_c^2$ is *τ-subsumed by* $\mathcal{H}_c^1$), if $\mathcal{T}_1 \succeq_\tau \mathcal{T}_2$ (write $\mathcal{H}_c^1 \succeq_\tau \mathcal{H}_c^2$). $\mathcal{H}_c^1$ is called a *τ-generalization* of $\mathcal{H}_c^2$ (and $\mathcal{H}_c^2$ a *τ-specialization* of $\mathcal{H}_c^1$).

Example 8.11 Given the *ϕ-homogeneous* atoms $\mathcal{H}_c^1 = \{(1, 2)\}$ and $\mathcal{H}_c^2 = \{(2, 1)\}$, $\mathcal{H}_c^1 \succeq_\tau \mathcal{H}_c^2$ holds as the positive threshold of $\mathcal{H}_c^1$ is smaller than that of $\mathcal{H}_c^2$, whereas the viceversa holds for

the negative thresholds. As a more general case, let $\mathcal{H}_c^1 = \{(1,2),(2,1)\}$ and $\mathcal{H}_c^2 = \{(1,1)\}$ be ϕ-*homogeneous* classifiers. Since $\{(1,2)\} \succeq_\tau \{(1,1)\}$ holds, $\mathcal{H}_c^1 \succeq_\tau \mathcal{H}_c^2$ follows.

Next we show that if $\mathcal{H}_c^1 \succeq_\tau \mathcal{H}_c^2$ then any document classified by $\mathcal{H}_c^2$ is classified by $\mathcal{H}_c^1$ as well.

Proposition 8.12 Let $\mathcal{H}_c^1$ and $\mathcal{H}_c^2$ be two classifiers in $\mathbf{H}_\Phi(P,N)$. Then $\mathcal{H}_c^1 \succeq_\tau \mathcal{H}_c^2$ implies $\mathcal{D}(\mathcal{H}_c^1) \supseteq \mathcal{D}(\mathcal{H}_c^2)$.

Proof Let $\mathcal{H}_c^1$ and $\mathcal{H}_c^2$ be two classifiers in $\mathbf{H}_\Phi(P,N)$. Next we show that $\mathcal{H}_c^1 \succeq_\tau \mathcal{H}_c^2$ implies $\mathcal{D}(\mathcal{H}_c^1) \supseteq \mathcal{D}(\mathcal{H}_c^2)$. The proof proceeds by induction. (*Basis*) $\mathcal{H}_c^1$ and $\mathcal{H}_c^2$ are atoms of the form, say, $\langle Pos, Neg, \{(p_1,n_1)\}\rangle$ and $\langle Pos, Neg, \{(p_2,n_2)\}\rangle$. By Definition 8.2, a document d is classified by $\mathcal{H}_c^2$ if (and only if) $|d \cap Pos| \geq p_2$ and $|d \cap Neg| \leq n_2$. Now, $\mathcal{H}_c^1 \succeq_\tau \mathcal{H}_c^2$ only if $p_1 \leq p_2$ and $n_1 \geq n_2$ (by Definition 8.10), which implies that a document d classified by $\mathcal{H}_c^2$ is classified by $\mathcal{H}_c^1$ as well, i.e., $\mathcal{D}(\mathcal{H}_c^1) \supseteq \mathcal{D}(\mathcal{H}_c^2)$. (*Inductive step*) $\mathcal{H}_c^1 \succeq_\tau \mathcal{H}_c^2$ only if, for each atom $\mathcal{H}_c^{2,i}$ appearing in $\mathcal{H}_c^2$ there exists an atom $\mathcal{H}_c^{1,j}$ appearing in $\mathcal{H}_c^1$ such that $\mathcal{H}_c^{1,j} \succeq_\tau \mathcal{H}_c^{2,i}$ (immediate from Definition 8.10). Since $\mathcal{H}_c^{1,j}$ classifies all documents classified by $\mathcal{H}_c^{2,i}$ (inductive hypothesis), it follows that $\mathcal{H}_c^1$ classifies all documents classified by $\mathcal{H}_c^2$. ∎

Unlike $\succeq_\phi$, the binary relation $\succeq_\tau$ is not a partial order.

Example 8.13 Classifiers $\mathcal{H}_c^1 = \{(1,2)\}$ and $\mathcal{H}_c^2 = \{(1,2),(2,2)\}$ are such that both $\mathcal{H}_c^1 \succeq_\tau \mathcal{H}_c^2$ and $\mathcal{H}_c^2 \succeq_\tau \mathcal{H}_c^1$ hold.

Definition 8.14 (Equivalence, minimality) Two ϕ-*homogeneous* classifiers $\mathcal{H}_c^1$ and $\mathcal{H}_c^2$ are *equivalent*, denoted $\mathcal{H}_c^1 \equiv \mathcal{H}_c^2$, if both $\mathcal{H}_c^1 \succeq_\tau \mathcal{H}_c^2$ and $\mathcal{H}_c^2 \succeq_\tau \mathcal{H}_c^1$. If a classifier $\mathcal{H}_c$ can be expressed as $\mathcal{H}_c^1 \vee \mathcal{H}_c^2$ such that either $\mathcal{H}_c^1 \succeq_\tau \mathcal{H}_c^2$ or $\mathcal{H}_c^2 \succeq_\tau \mathcal{H}_c^1$, then $\mathcal{H}_c$ is *redundant*. Otherwise $\mathcal{H}_c$ is *minimal*. If $\mathcal{H}_c = \langle Pos, Neg, \mathcal{T}\rangle$ is minimal, $\mathcal{T}$ is *minimal*. $\mathcal{H}_c^1$ *strictly* τ-subsumes $\mathcal{H}_c^2$, denoted $\mathcal{H}_c^1 >_\tau \mathcal{H}_c^2$, if $\mathcal{H}_c^1 \succeq_\tau \mathcal{H}_c^2$ and not $\mathcal{H}_c^1 \equiv \mathcal{H}_c^2$.

Example 8.15 Classifiers $\mathcal{H}_c^1$ and $\mathcal{H}_c^2$ of Example 8.13 are equivalent. Classifier $\mathcal{H}_c = \{(1,1),(3,1),(2,2)\}$ is redundant as $\mathcal{H}_c = \mathcal{H}_c^1 \vee \mathcal{H}_c^2$, where $\mathcal{H}_c^1 = \{(1,1),(2,2)\}$ and $\mathcal{H}_c^2 = \{(3,1)\}$ and, further, $\mathcal{H}_c^1 \succeq_\tau \mathcal{H}_c^2$ holds. On the contrary, $\mathcal{H}_c^1$ is minimal. It can be easily recognized that $\mathcal{H}_c \equiv \mathcal{H}_c^1$ holds.

The notion of equivalence encodes the intuition that equivalent hypotheses provide the same classification behavior. In fact, from Proposition 8.12 it immediately follows that equivalent classifiers do classify the same documents. The next lemma and proposition show that the viceversa holds as well, i.e., classifiers that classify the same documents are equivalent.

Lemma 8.16 Let $\mathcal{H}_c = \mathcal{H}_c^1 \vee \mathcal{H}_c^2$ and $\hat{\mathcal{H}}_c = \hat{\mathcal{H}}_c^1 \vee \hat{\mathcal{H}}_c^2$ be given. Then, $D(\mathcal{H}_c) \supseteq D(\hat{\mathcal{H}}_c)$ only if $(D(\mathcal{H}_c^1) \supseteq D(\hat{\mathcal{H}}_c^1)$ or $D(\mathcal{H}_c^2) \supseteq D(\hat{\mathcal{H}}_c^1))$ and $(D(\mathcal{H}_c^1) \supseteq D(\hat{\mathcal{H}}_c^2)$ or $D(\mathcal{H}_c^2) \supseteq D(\hat{\mathcal{H}}_c^2))$.

Proof Next we prove that, given $\mathcal{H}_c = \mathcal{H}_c^1 \vee \mathcal{H}_c^2$ and $\hat{\mathcal{H}}_c = \hat{\mathcal{H}}_c^1 \vee \hat{\mathcal{H}}_c^2$, the following holds: $D(\mathcal{H}_c) \supseteq D(\hat{\mathcal{H}}_c)$ only if $(D(\mathcal{H}_c^1) \supseteq D(\hat{\mathcal{H}}_c^1)$ or $D(\mathcal{H}_c^2) \supseteq D(\hat{\mathcal{H}}_c^1))$ and $(D(\mathcal{H}_c^1) \supseteq D(\hat{\mathcal{H}}_c^2)$ or $D(\mathcal{H}_c^2) \supseteq D(\hat{\mathcal{H}}_c^2))$. The proof proceeds by induction. (*Basis*) $\mathcal{H}_c^1 = \{(p_1, n_1)\}$, $\mathcal{H}_c^2 = \{(p_2, n_2)\}$ and $\hat{\mathcal{H}}_c = \{(\hat{p}, \hat{n})\}$ are atoms. By Definition 8.2, $\mathcal{D}(\mathcal{H}_c) = \{d \in \mathcal{D}\ s.t.\ |d \cap Pos| \geq p_1 \ \wedge\ |d \cap Neg| \leq n_1 \vee |d \cap Pos| \geq p_2 \wedge |d \cap Neg| \leq n_2\}$ and $\mathcal{D}(\hat{\mathcal{H}}_c) = \{d \in \mathcal{D}\ s.t.\ |d \cap Pos| \geq \hat{p} \wedge |d \cap Neg| \leq \hat{n}\}$. Thus, $D(\mathcal{H}) \supseteq D(\hat{\mathcal{H}}_c)$ only if either (1) $\hat{p} \geq p_1$ and $\hat{n} \leq n_1$ or (2) $\hat{p} \geq p2$ and $\hat{n} \leq n_2$. By Definition 8.10, condition (1) entails $\mathcal{H}_c^1 \succeq_\tau \hat{\mathcal{H}}_c$ and condition (2) $\mathcal{H}_c^2 \succeq_\tau \hat{\mathcal{H}}_c$, so as $D(\mathcal{H}) \supseteq D(\hat{\mathcal{H}}_c)$ only if $D(\mathcal{H}_c^1) \supseteq D(\hat{\mathcal{H}}_c)$ or $D(\mathcal{H}_c^2) \supseteq D(\hat{\mathcal{H}}_c)$. (*Inductive step*) $D(\mathcal{H}) \supseteq D(\hat{\mathcal{H}}_c)$ only if $D(\mathcal{H}_c^1) \cup D(\mathcal{H}_c^2) \supseteq D(\hat{\mathcal{H}}_c^1) \cup D(\hat{\mathcal{H}}_c^2)$ only if $D(\mathcal{H}_c^1) \cup D(\mathcal{H}_c^2) \supseteq D(\hat{\mathcal{H}}_c^1)$ and $D(\mathcal{H}_c^1) \cup D(\mathcal{H}_c^2) \supseteq D(\hat{\mathcal{H}}_c^2)$ only if (by inductive hypothesis) $D(\mathcal{H}_c^1) \supseteq D(\hat{\mathcal{H}}_c^1)$ or $D(\mathcal{H}_c^2) \supseteq D(\hat{\mathcal{H}}_c^1)$ and $D(\mathcal{H}_c^1) \supseteq D(\hat{\mathcal{H}}_c^2)$ or $D(\mathcal{H}_c^2) \supseteq D(\hat{\mathcal{H}}_c^2)$. ∎

Proposition 8.17 Let $\mathcal{H}_c$ and $\mathcal{H}'_c$ be two classifiers in $\mathbf{H}_\Phi(P, N)$. Then $\mathcal{D}(\mathcal{H}_c) \supseteq \mathcal{D}(\mathcal{H}'_c)$ implies $\mathcal{H}_c \succeq_\tau \mathcal{H}'_c$.

Proof Let $\mathcal{H}_c$ and $\mathcal{H}'_c$ be two classifiers in $\mathbf{H}_\Phi(P, N)$. We next show that $\mathcal{D}(\mathcal{H}_c) \supseteq \mathcal{D}(\mathcal{H}'_c)$ implies $\mathcal{H}_c \succeq_\tau \mathcal{H}'_c$. The proof proceeds by induction. (*Basis*) $\mathcal{H}_c = \{(p, n)\}$ and $\mathcal{H}'_c = \{(p', n')\}$ are atoms. By Definition 8.2, $\mathcal{D}(\mathcal{H}_c) = \{d \in \mathcal{D}\ s.t.\ d \cap Pos| \geq p \ \wedge\ |d \cap Neg| \leq n\}$ and $\mathcal{D}(\mathcal{H}'_c) = \{d \in \mathcal{D}\ s.t.\ d \cap Pos| \geq p' \ \wedge\ |d \cap Neg| \leq n'\}$. Clearly, $\mathcal{D}(\mathcal{H}_c) \supseteq \mathcal{D}(\mathcal{H}'_c)$ only if $p \leq p'$ and $n \geq n'$, that is, only if $\mathcal{H}_c \succeq_\tau \mathcal{H}'_c$. (*Inductive step*) Let $\mathcal{H}_c = \mathcal{H}_1 \vee \mathcal{H}_2$ and $\mathcal{H}'_c = \mathcal{H}'_1 \vee \mathcal{H}'_2$. Now, $\mathcal{D}(\mathcal{H}_c) \supseteq \mathcal{D}(\mathcal{H}'_c)$ only if $\mathcal{D}(\mathcal{H}_1) \cup \mathcal{D}(\mathcal{H}_2) \supseteq \mathcal{D}(\mathcal{H}'_1) \cup \mathcal{D}(H'_2)$ only if (by Lemma 8.16) $(\mathcal{D}(\mathcal{H}_1) \supseteq \mathcal{D}(\mathcal{H}'_1) \vee \mathcal{D}(\mathcal{H}_2) \supseteq \mathcal{D}(\mathcal{H}'_1)) \wedge (\mathcal{D}(\mathcal{H}_1) \supseteq \mathcal{D}(\mathcal{H}'_2) \vee \mathcal{D}(\mathcal{H}_2) \supseteq \mathcal{D}(H'_2))$ only if (by inductive hypothesis) $\mathcal{H}_1 \succeq_\tau \mathcal{H}'_1$ or $\mathcal{H}_2 \succeq_\tau \mathcal{H}'_1$ and $\mathcal{H}_1 \succeq_\tau \mathcal{H}'_2$ or $\mathcal{H}_2 \succeq_\tau \mathcal{H}'_2$ only if $\mathcal{H}_c \succeq_\tau \mathcal{H}'_c$. ∎
From the above proposition and proposition 8.12 it immediately follows the following statement.

Corollary 8.18 Given classifiers $\mathcal{H}_c^1$ and $\mathcal{H}_c^2$, $\mathcal{H}_c^1 \equiv \mathcal{H}_c^2$ iff $\mathcal{D}(\mathcal{H}_c^1) = \mathcal{D}(\mathcal{H}_c^2)$.

Proof We next prove that, given classifiers $\mathcal{H}_c^1$ and $\mathcal{H}_c^2$, $\mathcal{H}_c^1 \equiv \mathcal{H}_c^2$ iff $\mathcal{D}(\mathcal{H}_c^1) = \mathcal{D}(\mathcal{H}_c^2)$. Indeed, $\mathcal{H}_c^1 \equiv \mathcal{H}_c^2$ iff $\mathcal{H}_c^1 \succeq \mathcal{H}_c^2$ and $\mathcal{H}_c^2 \succeq \mathcal{H}_c^1$ iff $\mathcal{D}(\mathcal{H}_c^1) \supseteq \mathcal{D}(\mathcal{H}_c^2)$ and $\mathcal{D}(\mathcal{H}_c^2) \supseteq \mathcal{D}(\mathcal{H}_c^1)$ (by Proposition 8.12 and Proposition 8.17) iff $\mathcal{D}(\mathcal{H}_c^1) = \mathcal{D}(\mathcal{H}_c^2)$. ∎
Next we show a number of further interesting properties of classifiers.

Proposition 8.19 Let $\mathcal{H}_c = \langle Pos, Neg, \mathcal{T} \rangle$ be given. Then:

1. $\mathcal{H}_c$ is redundant iff there exist $(p_i, n_i),\ (p_j, n_j) \in \mathcal{T}$ such that $\{(p_i, n_i)\} \succeq_\tau \{(p_j, n_j)\}$.
2. $\mathcal{H}_c$ is minimal iff $\mathcal{T} = \{(p_1, n_1), \cdots, (p_r, n_r)\}$ is such that $p_i < p_j$ and $n_i < n_j$, or vice versa, for each $i, j \in [1, r]$.
3. If $\mathcal{H}_c = \mathcal{H}_c^1 \vee \mathcal{H}_c^2$ and $\mathcal{H}_c^1 \succeq_\tau \mathcal{H}_c^2$, then $\mathcal{H}_c \equiv \mathcal{H}_c^1$.

Proof Let $\mathcal{H}_c = \langle Pos, Neg, \mathcal{T} \rangle$ be given. We next show that the following properties hold:

1. $\mathcal{H}_c$ is redundant iff there exist $(p_i, n_i),\ (p_j, n_j) \in \mathcal{T}$ such that $\{(p_i, n_i)\} \succeq_\tau \{(p_j, n_j)\}$.
2. $\mathcal{H}_c$ is minimal iff $\mathcal{T} = \{(p_1, n_1), \cdots, (p_r, n_r)\}$ is such that $p_i < p_j$ and $n_i < n_j$, or vice versa, for each $i, j \in [1, r]$.
3. If $\mathcal{H}_c = \mathcal{H}_c^1 \vee \mathcal{H}_c^2$ and $\mathcal{H}_c^1 \succeq_\tau \mathcal{H}_c^2$, then $\mathcal{H}_c \equiv \mathcal{H}_c^1$.

(1) Let $\mathcal{H}_c = \mathcal{H}_c^1 \vee \cdots \vee \mathcal{H}_c^r$. By Definition 8.10, $\{(p_i, n_i)\} \succeq_\tau \{(p_j, n_j)\}$ iff $\mathcal{H}_c^i \succeq \mathcal{H}_c^j$, with $i, j \in [1, r]$, iff $\mathcal{H}_c = \mathcal{H}'_c \vee \mathcal{H}_c^j$ and $\mathcal{H}'_c \succeq \mathcal{H}_c^j$, where $\mathcal{H}'_c = \mathcal{H}_c^1 \vee \cdots \vee \mathcal{H}_c^{j-1} \vee \mathcal{H}_c^{j+1} \vee \cdots \vee \mathcal{H}_c^i \vee \cdots \vee \mathcal{H}_c^r$, iff $\mathcal{H}_c$ is redundant (by Definition 8.14).
(2) From point 1 above, $\mathcal{H}_c$ is minimal iff for each pair $(p_i, n_i),\ (p_j, n_j) \in \mathcal{T}$ neither $\{(p_i, n_i)\} \succeq_\tau \{(p_j, n_j)\}$ nor $\{(p_j, n_j)\} \succeq_\tau \{(p_i, n_i)\}$ iff neither ($p_i \leq p_j$ and $n_i \geq n_j$) nor ($p_j \leq p_i$ and $n_j \geq n_i$) (by Definition 8.10) iff $p_i < p_j$ and $n_i < n_j$, or viceversa.
(3) $\mathcal{H}_c^1 \succeq_\tau \mathcal{H}_c^2$ only if $D(\mathcal{H}_c^1) \supseteq D(\mathcal{H}_c^2)$ (by Proposition 8.12) only if $D(\mathcal{H}_c) = D(\mathcal{H}_c^1) \cup (\mathcal{H}_c^2) = D(\mathcal{H}_c^1)$ only if $\mathcal{H}_c \equiv \mathcal{H}_c^1$ (by Proposition 8.17). ∎

Example 8.20 According to Part 1 of Proposition 8.19, the classifier $\mathcal{H}_c = \{(1, 1), (2, 1), (2, 2)\}$ is redundant, as $\{(1, 1)\} \succeq_\tau \{(2, 1)\}$, while $\mathcal{H}_c^1 = \{(1, 1), (2, 2)\}$ is minimal. It is easily verified that $\mathcal{H}_c^1$ satisfies the condition $p_1 < p_2$ and $n_1 < n_2$ of Part 2 of Proposition 8.19. Since $\mathcal{H}_c = \mathcal{H}_c^1 \vee \mathcal{H}_c^2$, where $\mathcal{H}_c^2 = \{(2, 1)\}$, and $\mathcal{H}_c^1 \succeq_\tau \mathcal{H}_c^2$, $\mathcal{H}_c \equiv \mathcal{H}_c^1$ follows from Part 3 of Proposition 8.19.

Another interesting property of $\mathbf{H}_\Phi(P, N)$ is that, for any two classifiers $\mathcal{H}_c^1$ and $\mathcal{H}_c^2$ in it, there exists another classifier $\mathcal{H}_c$ in it (thus, $\mathcal{H}_c$ has the same sets of features of $\mathcal{H}_c^1$ and $\mathcal{H}_c^2$) which is a τ-specialization of both $\mathcal{H}_c^1$ and $\mathcal{H}_c^2$ that classifies exactly the documents classified by both $\mathcal{H}_c^1$ and $\mathcal{H}_c^2$ (i.e., $\mathcal{H}_c$ is equivalent to the logical AND of $\mathcal{H}_c^1$ and $\mathcal{H}_c^2$). We denote such a classifier by $and(\mathcal{H}_c^1, \mathcal{H}_c^2)$. Next we provide the (constructive) definition of $and(\mathcal{H}_c^1, \mathcal{H}_c^2)$. As we will see shortly after, this definition is a preliminary step for showing that $\succeq_\tau$ forms a complete lattice over the set of minimal classifiers.

Definition 8.21 (AND of classifiers) Given $\mathcal{H}_c^1, \mathcal{H}_c^2 \in \mathbf{H}_\Phi(P, N)$, $and(\mathcal{H}_c^1, \mathcal{H}_c^2)$ is the classifier inductively defined as follows:

- Basis: if $\mathcal{H}_c^1 = \{(p_1, n_1)\}$ and $\mathcal{H}_c^2 = \{(p_2, n_2)\}$ are atoms, then $and(\mathcal{H}_c^1, \mathcal{H}_c^2) = \{(p, n)\}$, where $p = Max\{p_1, p_2\}$ and $n = Min\{n_1, n_2\}$.
- Inductive step: if $\mathcal{H}_c^1 = \mathcal{H}_c^{1,1} \vee \mathcal{H}_c^{1,2}$ and $\mathcal{H}_c^2 = \mathcal{H}_c^{2,1} \vee \mathcal{H}_c^{2,2}$, then $and(\mathcal{H}_c^1, \mathcal{H}_c^2) = \mathcal{H}_1 \vee \mathcal{H}_2 \vee \mathcal{H}_3 \vee \mathcal{H}_4$, where $\mathcal{H}_1 = and(\mathcal{H}_c^{1,1}, \mathcal{H}_c^{2,1})$, $\mathcal{H}_2 = and(\mathcal{H}_c^{1,1}, \mathcal{H}_c^{2,2})$, $\mathcal{H}_3 = and(\mathcal{H}_c^{1,2}, \mathcal{H}_c^{2,1})$ and $\mathcal{H}_4 = and(\mathcal{H}_c^{1,2}, \mathcal{H}_c^{2,2})$.

Example 8.22 If $\mathcal{H}_c^1 = \{(1,2)\}$ and $\mathcal{H}_c^2 = \{(2,3)\}$, then $and(\mathcal{H}_c^1, \mathcal{H}_c^2) = \{max\{1,2\}, min\{2,3\}\} = \{2,2\}$ (base step of the definition). Intuitively, $and(\mathcal{H}_c^1, \mathcal{H}_c^2)$ is more specific than $\mathcal{H}_c^1$ and $\mathcal{H}_c^2$ as the higher the positive threshold, the lower the negative one, the more specific an atom is. As another example, if $\mathcal{H}_c^3 = \{(1,1),(2,3)\}$ and $\mathcal{H}_c^4 = \{(0,1),(2,2)\}$, then $and(\mathcal{H}_c^3, \mathcal{H}_c^4) = \{(1,1),(2,1),(2,2)\}$ (inductive step of the definition). Notice that $and(\mathcal{H}_c^3, \mathcal{H}_c^4)$ is not minimal.

Proposition 8.23 Given $\mathcal{H}_c, \hat{\mathcal{H}}_c \in \mathbf{H}_\Phi(P,N)$, the classifier $and(\mathcal{H}_c, \hat{\mathcal{H}}_c)$ is such that (1) $and(\mathcal{H}_c, \hat{\mathcal{H}}_c) \in \mathbf{H}_\Phi(P,N)$, (2) $\mathcal{D}(and(\mathcal{H}_c, \hat{\mathcal{H}}_c)) = \mathcal{D}(\mathcal{H}_c) \cap \mathcal{D}(\hat{\mathcal{H}}_c)$, and (3) $\mathcal{H}_c \succeq_\tau and(\mathcal{H}_c, \hat{\mathcal{H}}_c)$ and $\hat{\mathcal{H}}_c \succeq_\tau and(\mathcal{H}_c, \hat{\mathcal{H}}_c)$.

Proof Next we show that, given $\mathcal{H}_c, \hat{\mathcal{H}}_c \in \mathbf{H}_\Phi(P,N)$, the classifier $and(\mathcal{H}_c, \hat{\mathcal{H}}_c)$ is such that (1) $and(\mathcal{H}_c, \hat{\mathcal{H}}_c) \in \mathbf{H}_\Phi(P,N)$, (2) $\mathcal{D}(and(\mathcal{H}_c, \hat{\mathcal{H}}_c)) = \mathcal{D}(\mathcal{H}_c) \cap \mathcal{D}(\hat{\mathcal{H}}_c)$, and (3) $\mathcal{H}_c \succeq_\tau and(\mathcal{H}_c, \hat{\mathcal{H}}_c)$ and $\hat{\mathcal{H}}_c \succeq_\tau and(\mathcal{H}_c, \hat{\mathcal{H}}_c)$. The proof proceeds by induction. *Basis*. $\mathcal{H}_c = \{(p,n)\}$ and $\hat{\mathcal{H}}_c = \{(\hat{p},\hat{n})\}$ are atoms. *Statement* (1). To show that $and(\mathcal{H}_c, \hat{\mathcal{H}}_c)$ is in $\mathbf{H}_\Phi(P,N)$ it suffices to observe that both $p = Max\{p,\hat{p}\} \leq P$ and $n = Min\{n,\hat{n}\} \leq N$ hold. *Statement* (2). A document d is classified by $and(\mathcal{H}_c, \hat{\mathcal{H}}_c)$ iff d contains $x \geq Max\{p,\hat{p}\}$ positive features and $y < Min\{n_1,\hat{n}\}$ negative features, iff $x \geq p$, $x \geq \hat{p}$, $y < n$ and $y < \hat{n}$, iff d is classified by both $\mathcal{H}_c$ and $\hat{\mathcal{H}}_c$, i.e., $\mathcal{D}(\mathcal{H}_c) = \mathcal{D}(\mathcal{H}_c) \cap \mathcal{D}(\hat{\mathcal{H}}_c)$. *Statement* (3). Immediate from Statement 1 and Proposition 8.17.
Inductive step. Let $\mathcal{H}_c = \mathcal{H}_c^1 \vee \mathcal{H}_c^2$ and $\hat{\mathcal{H}}_c = \hat{\mathcal{H}}_c^1 \vee \hat{\mathcal{H}}_c^2$ be two classifiers. *Statement* (1). From Definition 8.21, $and(\mathcal{H}_c, \hat{\mathcal{H}}_c) = \mathcal{H}_1 \vee \mathcal{H}_2 \vee \mathcal{H}_3 \vee \mathcal{H}_4$, where $\mathcal{H}_1 = and(\mathcal{H}_c^1, \hat{\mathcal{H}}_c^1)$, $\mathcal{H}_2 = and(\mathcal{H}_c^1, \hat{\mathcal{H}}_c^2)$, $\mathcal{H}_3 = and(\mathcal{H}_c^2, \hat{\mathcal{H}}_c^1)$ and $\mathcal{H}_4 = and(\mathcal{H}_c^2, \hat{\mathcal{H}}_c^2)$. By the inductive hypothesis, $\mathcal{H}_1, \mathcal{H}_2, \mathcal{H}_3, \mathcal{H}_4$ are in $\mathbf{H}_\Phi(P,N)$ and thus, by Definition 8.2, $and(\mathcal{H}_c, \hat{\mathcal{H}}_c)$ is in $\mathbf{H}_\Phi(P,N)$. *Statement* (2). By using the inductive step of Definition 8.21, along with the inductive hypothesis of Statement 2, we get $\mathcal{D}(and(\mathcal{H}_c, \hat{\mathcal{H}}_c)) = D(\mathcal{H}_c^1) \cap D(\hat{\mathcal{H}}_c^1) \cup D(\mathcal{H}_c^1) \cap D(\hat{\mathcal{H}}_c^2) \cup D(\mathcal{H}_c^2) \cap D(\hat{\mathcal{H}}_c^1) \cup D(\mathcal{H}_c^2) \cap D(\hat{\mathcal{H}}_c^2)$, from which $\mathcal{D}(and(\mathcal{H}_c, \hat{\mathcal{H}}_c)) = D(\mathcal{H}_c^1) \cap D(\mathcal{H}_c^2)$ immediately follows. *Statement* (3). By using the inductive step of Definition 8.21, and applying the inductive hypothesis of Statement 3, we have that $\mathcal{H}_c^1 \succeq_\tau \mathcal{H}_1$ and $\hat{\mathcal{H}}_c^1 \succeq_\tau \mathcal{H}_1$ (as $\mathcal{H}_1 = and(\mathcal{H}_c^1, \hat{\mathcal{H}}_c^1)$), $\mathcal{H}_c^1 \succeq_\tau \mathcal{H}_2$ and $\hat{\mathcal{H}}_c^2 \succeq_\tau \mathcal{H}_2$ (as $\mathcal{H}_2 = and(\mathcal{H}_c^1, \hat{\mathcal{H}}_c^2)$), $\mathcal{H}_c^2 \succeq_\tau \mathcal{H}_3$ and $\hat{\mathcal{H}}_c^1 \succeq_\tau \mathcal{H}_3$ (as $\mathcal{H}_3 = and(\mathcal{H}_c^2, \hat{\mathcal{H}}_c^1)$), $\mathcal{H}_c^2 \succeq_\tau \mathcal{H}_4$ and $\hat{\mathcal{H}}_c^2 \succeq_\tau \mathcal{H}_4$ (as $\mathcal{H}_4 = and(\mathcal{H}_c^2, \hat{\mathcal{H}}_c^2)$). Thus, by Definition 8.10, it follows that both $\mathcal{H}_c \succeq_\tau and(\mathcal{H}_c, \hat{\mathcal{H}}_c)$ and $\hat{\mathcal{H}}_c \succeq_\tau and(\mathcal{H}_c, \hat{\mathcal{H}}_c)$ hold (that is, $and(\mathcal{H}_c, \hat{\mathcal{H}}_c)$ is more specific than both $\mathcal{H}_c$ and $\hat{\mathcal{H}}_c$). ∎

Example 8.24 In Example 8.22 we have seen that $and(\mathcal{H}_c^1, \mathcal{H}_c^2) = \{(2,2)\}$, for $\mathcal{H}_c^1 = \{(1,2)\}$ and $\mathcal{H}_c^2 = \{(2,3)\}$. Hence, $and(\mathcal{H}_c^1, \mathcal{H}_c^2)$ classifies a document d if d contains $x \geq 2$ positive features and $y < 3$ negative features; it is immediately recognized that a document satisfying such a condition is classified by both $\mathcal{H}_c^1$ and $\mathcal{H}_c^2$. On the other hand, d is classified by both $\mathcal{H}_c^1$ and $\mathcal{H}_c^2$ if it contains $x \geq max(1,2)$ positive features and $y < min(2,3)$ negative features, that is, if d is classified by $and(\mathcal{H}_c^1, \mathcal{H}_c^2)$.

The above result shows that the inclusion of the "$\wedge$" operator in the definition of classifier would not increase the expressivity of the language (i.e., it would be redundant).
Now we turn our attention to minimal classifiers. The following proposition shows a key result, that is, the uniqueness of the minimal classifier for an equivalence class.

Proposition 8.25 Any equivalence class into which is partitioned the hypothesis subspace $\mathbf{H}_\Phi(P, N)$ by the relation $\equiv$ has a unique minimal classifier.

Proof Next we show that any equivalence class into which is partitioned the hypothesis subspace $\mathbf{H}_\Phi(P, N)$ by the relation $\equiv$ has a unique minimal classifier. To this end, we prove that, if $\mathcal{H}_c$ and $\hat{\mathcal{H}}_c$ are two minimal classifiers such that $\mathcal{H}_c \equiv \hat{\mathcal{H}}_c$, then $\mathcal{H}_c = \hat{\mathcal{H}}_c$. From which the statement immediately follows. The proof proceeds by induction. *Base*: $\mathcal{H}_c$ and $\hat{\mathcal{H}}_c$ are atoms. Trivial. *Induction*: $\mathcal{H}_c = \mathcal{H}_c^1 \vee \mathcal{H}_c^2$ and $\hat{\mathcal{H}}_c = \hat{\mathcal{H}}_c^1 \vee \hat{\mathcal{H}}_c^2$. Note that, from the minimality of $\mathcal{H}_c$ and $\hat{\mathcal{H}}_c$, the minimality of $\mathcal{H}_c^1, \mathcal{H}_c^2, \hat{\mathcal{H}}_c^1$ and $\hat{\mathcal{H}}_c^2$ follows. By corollary 8.18, $\mathcal{H}_c \equiv \hat{\mathcal{H}}_c$ iff $D(\mathcal{H}_c) = D(\hat{\mathcal{H}}_c)$ iff $D(\mathcal{H}_c) \supseteq D(\hat{\mathcal{H}}_c)$ and $D(\mathcal{H}_c) \subseteq D(\hat{\mathcal{H}}_c)$. By Lemma 8.16, $D(\mathcal{H}_c) \supseteq D(\hat{\mathcal{H}}_c)$ only if $D(\mathcal{H}_c^1) \supseteq D(\hat{\mathcal{H}}_c^1)$ or $D(\mathcal{H}_c^2) \supseteq D(\hat{\mathcal{H}}_c^1)$ and $D(\mathcal{H}_c^1) \supseteq D(\hat{\mathcal{H}}_c^2)$ or $D(\mathcal{H}_c^2) \supseteq D(\hat{\mathcal{H}}_c^2)$. Likewise, $D(\mathcal{H}_c) \subseteq D(\hat{\mathcal{H}}_c)$ only if $D(\mathcal{H}_c^1) \subseteq D(\hat{\mathcal{H}}_c^1)$ or $D(\mathcal{H}_c^1) \subseteq D(\hat{\mathcal{H}}_c^2)$ and $D(\mathcal{H}_c^2) \subseteq D(\hat{\mathcal{H}}_c^1)$ or $D(\mathcal{H}_c^2) \subseteq D(\hat{\mathcal{H}}_c^2)$. It turns out that, because of the minimality of $\mathcal{H}_c^1, \mathcal{H}_c^2, \hat{\mathcal{H}}_c^1$ and $\hat{\mathcal{H}}_c^2$, either (1) $D(\mathcal{H}_c^1) = D(\hat{\mathcal{H}}_c^1)$ and $D(\mathcal{H}_c^2) = D(\hat{\mathcal{H}}_c^2)$ or (2) $D(\mathcal{H}_c^1) = D(\hat{\mathcal{H}}_c^2)$ and $D(\mathcal{H}_c^2) = D(\hat{\mathcal{H}}_c^1)$, only if (by Corollary 8.18) either (1) $\mathcal{H}_c^1 \equiv \hat{\mathcal{H}}_c^1$ and $\mathcal{H}_c^2 \equiv \hat{\mathcal{H}}_c^2$ or (2) $\mathcal{H}_c^1 \equiv \hat{\mathcal{H}}_c^2$ and $\mathcal{H}_c^2 \equiv \hat{\mathcal{H}}_c^1$. By the inductive hypothesis, $\mathcal{H}_c^i \equiv \hat{\mathcal{H}}_c^j$ only if $\mathcal{H}_c^i = \hat{\mathcal{H}}_c^j$, from which $\mathcal{H}_c = \hat{\mathcal{H}}_c$. ∎
We denote by $Min(\mathcal{H}_c)$ the minimal classifier of the equivalence class of $\mathcal{H}_c$. From now on, we will restrict our attention to the set of minimal classifiers $\mathbf{M}_\Phi(P, N) \subseteq \mathbf{H}_\Phi(P, N)$. It is immediate to recognize that the restriction of the binary relation $\succeq_\tau$ to $\mathbf{M}_\Phi(P, N)$ is a partial order. More precisely, it is a complete lattice.

Proposition 8.26 The poset $(\mathbf{M}_\Phi(P, N), \succeq_\tau)$, where $\mathbf{M}_\Phi(P, N)$ is the set of the minimal classifiers in $\mathbf{H}_\Phi(P, N)$, is a complete lattice. Indeed, for any two elements $\mathcal{H}_c^1$ and $\mathcal{H}_c^2$ of $\mathbf{M}_\Phi(P, N)$, there are both the greatest lower bound $glb_\tau(\mathcal{H}_c^1, \mathcal{H}_c^2)$ and the least upper bound $lub_\tau(\mathcal{H}_c^1, \mathcal{H}_c^2)$ as follows:

(a) $lub_\tau(\mathcal{H}_c^1, \mathcal{H}_c^2) = Min(\mathcal{H}_c^1 \vee \mathcal{H}_c^2)$, that is, the least upper bound of $\mathcal{H}_c^1, \mathcal{H}_c^2$ is the minimal classifier of the equivalence class of $\mathcal{H}_c^1 \vee \mathcal{H}_c^2$.

(b) $glb_\tau(\mathcal{H}_c^1, \mathcal{H}_c^2) = Min(and(\mathcal{H}_c^1, \mathcal{H}_c^2))$, that is, the greatest lower bound of $\mathcal{H}_c^1, \mathcal{H}_c^2$ is the minimal classifier of the equivalence class of $and(\mathcal{H}_c^1, \mathcal{H}_c^2)$.

Proof The statement we are going to prove is that the poset $(\mathbf{M}_\Phi(P, N), \succeq_\tau)$, where $\mathbf{M}_\Phi(P, N)$ is the set of the minimal classifiers in $\mathbf{H}_\Phi(P, N)$, is a complete lattice. To this end, let us consider two classifiers $\mathcal{H}_c^1$ and $\mathcal{H}_c^2$ in $\mathbf{M}_\Phi(P, N)$. We first show that $lub_\tau(\mathcal{H}_c^1, \mathcal{H}_c^2) = Min(\mathcal{H}_c^1 \vee \mathcal{H}_c^2)$.

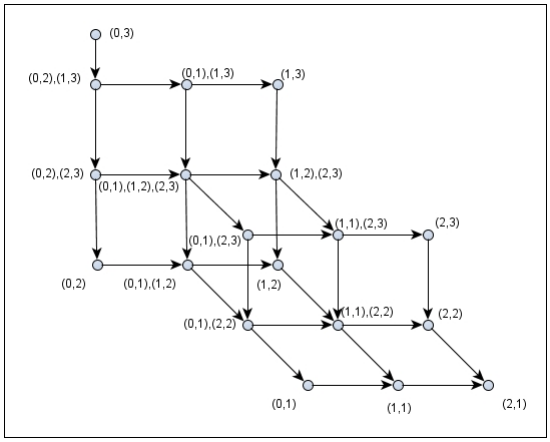

Figure 8.1: τ-subsumption lattice with threshold bounds $P = 2$ and $N = 3$

Let $\mathcal{H}_c \in \mathbf{M}_\Phi(P, N)$ be a τ-generalization of both $\mathcal{H}_c^1$ and $\mathcal{H}_c^2$, i.e., $\mathcal{H}_c \succeq_\tau \mathcal{H}_c^1$ and $\mathcal{H}_c \succeq_\tau \mathcal{H}_c^2$. Thus, by Proposition 8.12, both $\mathcal{D}(\mathcal{H}_c) \supseteq \mathcal{D}(\mathcal{H}_c^1)$ and $\mathcal{D}(\mathcal{H}_c) \supseteq \mathcal{D}(\mathcal{H}_c^2)$ hold. On the other hand, $\mathcal{D}(Min(\mathcal{H}_c^1 \vee \mathcal{H}_c^2)) = \mathcal{D}(\mathcal{H}_c^1) \cup \mathcal{D}(\mathcal{H}_c^2)$, so that $\mathcal{D}(\mathcal{H}_c) \supseteq \mathcal{D}(Min(\mathcal{H}_c^1 \vee \mathcal{H}_c^2))$. Therefore, by Proposition 8.17, $\mathcal{H}_c \succeq_\tau Min(\mathcal{H}_c^1 \vee \mathcal{H}_c^2)$, from which the statement $lub_\tau(\mathcal{H}_c^1, \mathcal{H}_c^2) = Min(\mathcal{H}_c^1 \vee \mathcal{H}_c^2)$ follows.
Now we prove that $glb_\tau(\mathcal{H}_c^1, \mathcal{H}_c^2) = Min(and(\mathcal{H}_c^1, \mathcal{H}_c^2))$. Let $\mathcal{H}_c$ be a τ-specialization of both $\mathcal{H}_c^1$ and $\mathcal{H}_c^2$, i.e., $\mathcal{H}_c^1 \succeq_\tau \mathcal{H}_c$ and $\mathcal{H}_c^2 \succeq_\tau \mathcal{H}_c$. Thus, by Proposition 8.12, both $\mathcal{D}(H_c^1) \supseteq \mathcal{D}(H_c)$ and $\mathcal{D}(\mathcal{H}_c^2) \supseteq \mathcal{D}(\mathcal{H}_c)$ hold. On the other hand, $\mathcal{D}(Min(and(\mathcal{H}_c^1, \mathcal{H}_c^2)) = \mathcal{D}(\mathcal{H}_c^1) \cap \mathcal{D}(\mathcal{H}_c^2)$ by Proposition 8.23, so that $\mathcal{D}(Min(and(\mathcal{H}_c^1, \mathcal{H}_c^2)) \supseteq \mathcal{D}(\mathcal{H}_c)$. Therefore, by Proposition 8.17, $Min(and(\mathcal{H}_c^1, \mathcal{H}_c^2)) \succeq_\tau \mathcal{H}_c$, from which the statement $glb_\tau(\mathcal{H}_c^1, \mathcal{H}_c^2) = Min(and(\mathcal{H}_c^1, \mathcal{H}_c^2))$ follows. ∎

Example 8.27 The τ-subsumption lattice, for threshold bounds $P = 2$ and $N = 3$, is depicted in Figure 8.1. How we can see, there are 19 classifiers; the most general one is $\{(0, 3)\}$ and the most specific one is $\{(2, 1)\}$. Further, the maximum order of a classifier is 3 (the order of $\{(0, 1), (1, 2), (2, 3)\}$).

We conclude this section by providing a constructive definition of both lub_τ and glb_τ. Let $\mathcal{H}_c^1 = \langle Pos, Neg, \mathcal{T}_1 \rangle$ and $\mathcal{H}_c^2 = \langle Pos, Neg, \mathcal{T}_2 \rangle$. Now, by Proposition 8.26, $lub_\tau(\mathcal{H}_c^1, \mathcal{H}_c^2) = Min(\mathcal{H}_c^1 \vee \mathcal{H}_c^2)$, that is, $lub_\tau(\mathcal{H}_c^1, \mathcal{H}_c^2) = Min(\langle Pos, Neg, \mathcal{T}_1 \cup \mathcal{T}_2 \rangle)$ (by Definition 8.2), that is, $lub_\tau(\mathcal{H}_c^1, \mathcal{H}_c^2) =$

$\langle Pos, Neg, Min(\mathcal{T}_1 \cup \mathcal{T}_2)\rangle$, where $Min(\mathcal{T}_1 \cup \mathcal{T}_2)$ is obtained from $\mathcal{T}_1 \cup \mathcal{T}_2$ simply by discarding every (p, n) such that there exists $(p', n') \in \mathcal{T}_1 \cup \mathcal{T}_2$ such that $\{(p', n')\} \succeq_\tau \{(p, n)\}$ (immediate from Proposition 8.19, Part 1). We denote $Min(\mathcal{T}_1 \cup \mathcal{T}_2)$ by $\sqcup(\mathcal{T}_1, \mathcal{T}_2)$, so that $lub_\tau(\mathcal{H}_c^1, \mathcal{H}_c^2) = \langle Pos, Neg, \sqcup(\mathcal{T}_1, \mathcal{T}_2)\rangle$. Likewise, by Proposition 8.26, we have that $glb_\tau(\mathcal{H}_c^1, \mathcal{H}_c^2) = Min(and(\mathcal{H}_c^1, \mathcal{H}_c^2))$. We denote by $\sqcap(\mathcal{T}_1, \mathcal{T}_2)$ the threshold set constructed by using Definition 8.21 and then minimized as shown above, so as $glb_\tau(\mathcal{H}_c^1, \mathcal{H}_c^2) = \langle Pos, Neg, \sqcap(\mathcal{T}_1, \mathcal{T}_2)\rangle$

Proposition 8.28 Let $\mathcal{H}_c^1 = \langle Pos, Neg, \mathcal{T}_1\rangle$ and $\mathcal{H}_c^2 = \langle Pos, Neg, \mathcal{T}_2\rangle$ be two (minimal) classifiers in $\mathbf{M}_\Phi(P, N)$. Then

$$lub_\tau(\mathcal{H}_c^1, \mathcal{H}_c^2) = \langle Pos, Neg, \sqcup(\mathcal{T}_1, \mathcal{T}_2)\rangle$$
$$glb_\tau(\mathcal{H}_c^1, \mathcal{H}_c^2) = \langle Pos, Neg, \sqcap(\mathcal{T}_1, \mathcal{T}_2)\rangle$$

where $\sqcup(\mathcal{T}_1, \mathcal{T}_2)$ and $\sqcap(\mathcal{T}_1, \mathcal{T}_2)$ are constructively defined as shown in Figure 8.2.

Proof We first prove $lub_\tau(\mathcal{H}_c^1, \mathcal{H}_c^2) = \langle Pos, Neg, \sqcup(\mathcal{T}_1, \mathcal{T}_2)\rangle$. By Proposition 8.26, $lub_\tau(\mathcal{H}_c^1, \mathcal{H}_c^2) = Min(\mathcal{H}_c^1 \vee \mathcal{H}_c^2)$, that is, $lub_\tau(\mathcal{H}_c^1, \mathcal{H}_c^2) = Min(\langle Pos, Neg, \mathcal{T}_1 \cup \mathcal{T}_2\rangle)$. It is immediate to recognize that this classifier is $\langle Pos, Neg, \sqcup(\mathcal{T}_1, \mathcal{T}_2)\rangle$ (see Figure 8.2), as function $\sqcup(\mathcal{T}_1, \mathcal{T}_2)$ simply minimizes $\mathcal{T}_1 \cup \mathcal{T}_2$ by discarding all thresholds (p_j, n_j) such that there exists (p_i, n_i) such that $\{(p_i, n_i)\} \succeq_\tau \{(p_j, n_j)\}$ holds (see Proposition 8.19 - Part 1).
Now we show that $glb_\tau(\mathcal{H}_c^1, \mathcal{H}_c^2) = \langle Pos, Neg, \sqcap(\mathcal{T}_1, \mathcal{T}_2)\rangle$. By Proposition 8.26, we have that $glb_\tau = Min(and(\mathcal{H}_c^1, \mathcal{H}_c^2))$. Next we show that $Min(and(\mathcal{H}_c^1, \mathcal{H}_c^2)) = \langle Pos, Neg, \sqcap(\mathcal{T}_1, \mathcal{T}_2)\rangle$. To this end, we observe that lines 9-12 of Figure 8.2 are the iterative version of the inductive definition of $and(\mathcal{H}_c^1, \mathcal{H}_c^2)$ (Definition 8.21). Indeed, if both $\mathcal{H}_c^1$ and $\mathcal{H}_c^2$ are atoms, then the function computes $\sqcap(\mathcal{T}_1, \mathcal{T}_2) = \{(Max(p_1, p_2), Min(n_1, n_2))\}$, which coincides with the base step of Definition 8.21 (of course, the classifier $\langle Pos, Neg, \{(Max(p_1, p_2), Min(n_1, n_2))\}\rangle$ is minimal, being an atom). Now, let us consider the general case. It is easy to see that the inductive step of Definition 8.21 generates, for each couple of pairs $(p_1, n_1) \in \mathcal{T}_1$ and $(p_2, n_2) \in \mathcal{T}_2$, a pair $\{(Max(p_1, p_2), Min(n_1, n_2))\}$. And this is exactly what *function* $\sqcap(\mathcal{T}_1, \mathcal{T}_2)$ does at lines 9-12. Thus, after the two nested "for" have been carried out (lines 10-12), the classifier $and(\mathcal{H}_c^1, \mathcal{H}_c^2))$ is generated. However, this classifier may not be minimal (see Example 8.22), so that function *Minimize* is invoked. So, we finally get $\langle Pos, Neg, \sqcap(\mathcal{T}_1, \mathcal{T}_2)\rangle = Min(and(\mathcal{H}_c^1, \mathcal{H}_c^2))$. ∎

8.4.4 The minimal hypothesis space

In the previous subsection we defined the notion of minimal classifier as the representative hypothesis of an equivalence class. Minimality is a desirable property of classifiers as, by guaranteeing the uniqueness of representation, imposes an ordered structure within the hypothesis space. For this reason, we restrict ourselves to *minimal* classifiers.

Functions $\sqcap(\mathcal{T}_1, \mathcal{T}_2)$ *and* $\sqcup(\mathcal{T}_1, \mathcal{T}_2)$

---

1. **function** Minimize($\mathcal{T}$)
2. drop from $\mathcal{T}$ each (p, n) s.t. $\exists\ (p', n') \in \mathcal{T}$ s.t. $(p', n') \succeq_\tau (p, n)$.
4. **return** $\mathcal{T}$.

---

5. **function** $\sqcup(\mathcal{T}_1, \mathcal{T}_2)$
6. **return** Minimize($\mathcal{T}_1 \cup \mathcal{T}_2$);

---

8. **function** $\sqcap(\mathcal{T}_1, \mathcal{T}_2)$
9. $\mathcal{T} = \emptyset$;
10. **for each** $(p, n) \in \mathcal{T}_1$
11. **for each** $(p', n') \in \mathcal{T}_2$
12. $\mathcal{T} = \mathcal{T} \cup \{(Max\{p, p'\}, Min\{n, n'\})\}$;
13. **return** Minimize($\mathcal{T}$);

Figure 8.2: Computation of $\sqcup(\mathcal{T}_1, \mathcal{T}_2)$ and $glb_\tau(\mathcal{T}_1, \mathcal{T}_2)$

Definition 8.29 Let the feature space $\mathcal{F}(k) = \langle Pos^*(k),\ Neg^*(k)\rangle$ and the threshold bounds P and N be given. The *minimal hypothesis space* constructible over $\mathcal{F}(k)$, for the given P and N values, is $(\mathbf{M}(\mathcal{F}(k), P, N), \succeq_\tau, \succeq_\phi)$, where

$$\mathbf{M}(\mathcal{F}(k), P, N) = \bigcup_{\Phi} \mathbf{M}_\Phi(P, N)\ \ s.t.\ \ \Phi \in \{\langle Pos, Neg\rangle |\ Pos \subseteq Pos^*(k), Neg \subseteq Neg^*(k)\}.$$

Thus, a minimal hypothesis space is uniquely determined by k, P and N.
Using the previously defined notational convention, in the following we will denote by $\mathbf{M}_\tau(\mathcal{F}(k))$ the set of (minimal) classifiers in $\mathbf{M}(\mathcal{F}(k), P, N)$ with threshold set $\mathcal{T}$. It is immediate to recognize that, given the minimal threshold set $\mathcal{T}$, $\mathbf{M}_\tau(\mathcal{F}(k))$ and $\mathbf{H}_\tau(\mathcal{F}(k))$ coincide, as both consist of all (minimal) classifiers with threshold set $\mathcal{T}$ constructible over $\mathcal{F}(k)$. It turns out that $(\mathbf{M}_\tau(\mathcal{F}(k)), \succeq_\phi)$ and $(\mathbf{H}_\tau(\mathcal{F}(k)), \succeq_\phi)$ coincide as well.
Next we discuss on the structure of $(\mathbf{M}(\mathcal{F}(k), P, N)\ \succeq_\tau, \succeq_\phi)$, as determined by the two subsumption relations. Since the ϕ-subsumption and the τ-subsumption lattices are the basic building blocks of a minimal hypothesis space, we start our discussion by preliminarily showing the size of such lattices.

The size of the two types of lattice

A ϕ-subsumption lattice $\mathbf{M}_{\mathcal{T}}(\mathcal{F}(k))$ consists, for a given $\mathcal{T}$, of all hypotheses that can be built over a given feature space $\mathcal{F}(k)$, each hypothesis corresponding to a particular choice of the sets Pos and Neg over $\mathcal{F}(k)$. It is immediate to recognize the following fact.

Fact 1 The size of $\mathbf{M}_{\mathcal{T}}(\mathcal{F}(k))$ is equal to the number of sets Pos and Neg constructible over the feature space $\mathcal{F}(k) = \langle Pos^*(k), Neg^*(k)\rangle$, that is, $|\mathbf{H}_{\mathcal{T}}(\mathcal{F}(k))| = 2^{2k}$.

A τ-subsumption lattice $\mathbf{M}_{\Phi}(P, N)$ consists, for a given $\Phi = \langle Pos, Neg\rangle$, of all hypotheses that can be built for the given threshold bounds P and N, each hypothesis corresponding to a particular threshold set satisfying P and N. The next lemma and proposition show both the size of $\mathbf{M}_{\Phi}(P, N)$ and the maximum order of a classifier.

Lemma 8.30 Given threshold bounds P and N, along with $k \leq Min(P + 1, N)$, let $T_k^+ = \{p_1, ..., p_k\}$ and $T_k^- = \{n_1, ..., n_k\}$ be sets of integers, where $\forall\ i \leq k$, $0 \leq\ p_i \leq P$ and $0 < n_i \leq N$. Then there exists a unique subset $S \subseteq T_k^+ \times T_k^-$ having size k which is a minimal threshold set.

Proof Given threshold bounds P and N, along with $k \leq Min(P + 1, N)$, let $T_k^+ = \{p_1, ..., p_k\}$ and $T_k^- = \{n_1, ..., n_k\}$ be sets of integers, where $\forall\ i \leq k$, $0 \leq\ p_i \leq P$ and $0 < n_i \leq N$. Next we show that there exists a unique subset $S \subseteq T_k^+ \times T_k^-$ having size k which is a minimal threshold set. Without loss of generality, we assume that $\forall p_i, p_j \in T_k^+$ and $\forall n_i, n_j \in T_k^-$, such that $i < j$, both $p_i < p_j$ and $n_i < n_j$ hold.
Existence. The set $S = \{(p_1, n_1), ..., (p_i, n_i), ...(p_k, n_k)\}$ is a subset of $T_k^+ \times T_k^-$ of size k where, for each pair of elements (p_i, n_i) and (p_j, n_j), with $i < j$, both $p_i < p_j$ and $n_i < n_j$ hold. Thus, by Proposition 8.19 - Part 2, S is a minimal threshold set.
Uniqueness. We show that any another subset $S' \neq S$ of $T_k^+ \times T_k^-$, which is a minimal threshold set, has size lower than k. Suppose that $(p_i, n_j) \in S'$ is such that $i < j$ (the case $j < i$ is likewise). Since S' is a minimal threshold set, by Proposition 8.19 - Part 2, for any $(p_s, n_t) \in S'$, either $p_s < p_i$ and $n_t < n_j$ or $p_i < p_s$ and $n_j < n_t$. Now, it is immediate to recognize that, by the above assumption on the ordering of the elements of T_k^+ and T_k^-, the elements $p_s \in T_k^+$ such that $s < i$ (i.e., smaller than p_i) are $i - 1$, while the elements $n_t \in T_k^-$ such that $j < t$ (i.e., greater than n_t) are $k - j$. It turns out that (1) there are at most $i - 1$ elements $(p_s, n_t) \in S'$ such that $s < i$, and (2) there are at most $k - j$ elements $(p_s, n_t) \in S'$ such that $j < t$. That is, the size of S' is at most $i + k - j < k$. ∎

Proposition 8.31 Given the threshold bounds P and N, (1) the maximum order of a classifier in $\mathbf{M}_{\Phi}(P, N)$ is $Min(P+1, N)$, and (2) the number of minimal threshold sets that can be constructed

for the given bounds is

$$\lambda(P,N) = \sum_{j=1}^{Min\{P+1,N\}} \left(\begin{array}{c} P+1 \\ j \end{array} \right) \left(\begin{array}{c} N \\ j \end{array} \right) \tag{8.1}$$

Proof *Part 1.* Every classifier in $\mathbf{M}_\Phi(P,N)$ is of order $r \leq Min\{P+1,N\}$. In fact, given $\mathcal{T} = \{(p_1,n_1),\ldots,(p_r,n_r)\}$, from Part 2 of Proposition 8.19 we have that $p_i \neq p_j$ and $n_i \neq n_j$ for all $i,j \in [1,r], i \neq j$. That is, in $\mathcal{T}$ there appear r different positive thresholds and r negative thresholds. Since $0 \leq p_i \leq P$ and $0 < n_i \leq N$, for each $i \in [1,r]$, it turns out that $r \leq P+1$ and $r \leq N$, i.e., $r \leq Min\{P+1,N\}$.
Part 2. Given P,N and $s = Min\{P+1,N\}$, let us consider the sets $T^+ = \{0,1,\ldots,P\}$ and $T^- = \{1,\ldots,N\}$. T^+ (resp. T^-) is the set of possible values for the positive (resp. negative) thresholds appearing in the classifiers of $\mathbf{M}_\Phi(P,N)$. Now, $\forall r \in [1,s]$, there exist $binom(P+1,r)$ subsets $T_r^+ \subseteq T^+$ (made of r elements from T^+) and $binom(N,r)$ subsets $T_r^- \subseteq T^-$. Also, from Lemma 8.30 we know that, for each pair of sets T_r^+, T_r^- , there is a unique minimal threshold set $S \subset T_r^+ \times T_r^-$ or order r constructible from T^+ and T^-. It turns out that there are $binom(P+1,r) \times binom(N,r)$ minimal threshold sets of order r. Therefore, since $r \leq Min\{P+1,N\}$, the total number of threshold sets in $\mathbf{M}_\Phi(P,N)$ is that given by equation 8.1. ∎

The landscape from the τ-subsumption perspective

Given threshold bounds P and N, let us consider two lattices $(\mathbf{M}_\Phi(P,N),\succeq_\tau)$ and $(\mathbf{M}_{\Phi'}(P,N),\succeq_\tau)$, where $\Phi = \langle Pos, Neg\rangle$ and $\Phi' = \langle Pos', Neg'\rangle$. By Proposition 8.31, they have the same number $\lambda(P,N)$ of classifiers, i.e., all those constructible for the given P and N. Hence, there is a one to one correspondence g between the classifiers of $\mathbf{M}_\Phi(P,N)$ and those of $\mathbf{M}_{\Phi'}(P,N)$, two related classifiers having the same threshold sets. Since the τ-subsumption relation among classifiers is determined by the τ-subsumption relation among the respective threshold sets (see Definition 8.10), clearly $\mathcal{H}_c^1 \succeq_\tau \mathcal{H}_c^2$ holds in $(\mathbf{M}_\Phi(P,N),\succeq_\tau)$ iff $g(\mathcal{H}_c^1) \succeq_\tau g(\mathcal{H}_c^2)$ holds in $(\mathbf{M}_{\Phi'}(P,N),\succeq_\tau)$. That is, the two lattices are isomorphic. Further, all classifiers in $\mathbf{M}_\Phi(P,N)$ share the feature sets $\langle Pos, Neg\rangle$, while those in $\mathbf{M}_{\Phi'}(P,N)$ share the feature sets $\langle Pos', Neg'\rangle$, so as $\mathbf{M}_\Phi(P,N)$ and $\mathbf{M}_{\Phi'}(P,N)$ are disjoint. Since the number of different Φs (i.e., pairs of sets Pos and Neg) constructible over a given feature space $\mathcal{F}(k)$ is 2^{2k}, we may conclude that $(\mathbf{M}(\mathcal{F}(k),P,N),\succeq_\tau)$ has a structure made of 2^{2k} isomorphic, disjoint lattices
$(\mathbf{M}_\Phi(P,N),\succeq_\tau)$, each of size $\lambda(P,N)$. For an instance, given $P=2$ and $N=3$, $(\mathbf{M}(\mathcal{F}(k),2,3),\succeq_\tau)$ will consists of 2^{2k} lattices whose structure is that depicted in Figure 8.1.

Fact 2 The partial order $(\mathbf{M}(\mathcal{F}(k),P,N),\succeq_\tau)$ consists of 2^{2k} isomorphic, disjoint lattices $(\mathbf{M}_\Phi(P,N),\succeq_\tau)$, each of size $\lambda(P,N)$.

The landscape from the ϕ-subsumption perspective

The ϕ-subsumption perspective is of course dual to the τ-subsumption one. Consider two lattices $(\mathbf{M}_{\mathcal{T}}(\mathcal{F}(k)), \succeq_\phi)$ and $\mathbf{M}_{\mathcal{T}'}(\mathcal{F}(k), \succeq_\phi)$, for any $\mathcal{T},\ \mathcal{T}'$. As stated by Fact 1, their size is 2^{2k}. Since the relationship $\succeq_\phi$ among classifiers is determined only by the inclusion relationship among the respective sets of features (see Definition 8.5), the structure of the above lattices does not depend on $\mathcal{T}$. Hence, $\mathbf{M}_{\mathcal{T}}(\mathcal{F}(k))$ and $\mathbf{M}_{\mathcal{T}'}(\mathcal{F}(k))$ are isomorphic under $\succeq_\phi$. Since any hypothesis in $\mathbf{M}_{\mathcal{T}}(\mathcal{F}(k))$ has threshold set $\mathcal{T}$ and any hypothesis in $\mathbf{M}_{\mathcal{T}'}(\mathcal{F}(k))$ has threshold set $\mathcal{T}'$, $\mathbf{M}_{\mathcal{T}}(\mathcal{F}(k))$ and $\mathbf{M}_{\mathcal{T}'}(\mathcal{F}(k))$ are disjoint. Therefore, in the hypothesis space $(\mathbf{M}(\mathcal{F}(k), P, N), \succeq_\phi)$ there exist $\lambda(P, N)$ isomorphic, disjoint lattices $(\mathbf{M}_{\mathcal{T}}(\mathcal{F}(k)), \succeq_\phi)$, each of size 2^{2k}.

Fact 3 The partial order $(\mathbf{M}(\mathcal{F}(k), P, N), \succeq_\phi)$ consists of $\lambda(P, N)$ isomorphic, disjoint lattices $(\mathbf{M}_{\tau}(\mathcal{F}(k)), \succeq_\tau)$, each of size 2^{2k}.

8.4.5 Decision Boundaries

There is an interesting graphical representation of a classifier $\mathcal{H}_c = \langle Pos, Neg, \mathcal{T}\rangle$ on the 2-dimensional space N^2 (see Figure 8.3). Here, each point (x, y), with x and y non-negative integers, is labeled by a pair of integers $\langle \pi(x, y), \nu(x, y)\rangle$, where $\pi(x, y)$ is the number of *positive* examples (documents) and $\nu(x, y)$ the number of *negative* ones containing exactly x features from Pos and y features from Neg. Intuitively, we may think of a point (x, y) as identifying the set of (both positive and negative) examples with x positive features and y negative ones. Hence, the region of the plane

$$\mathrm{R}_{\mathcal{H}_c} = \{(x, y) \mid x \leq |Pos|,\ y \leq |Neg|,\ \exists (p_i, n_i) \in \mathcal{T}\ s.t.\ x \geq p_i,\ y < n_i\},$$

whose points satisfy the threshold conditions, identifies the documents that are classified by $\mathcal{H}_c$ (we call $\mathrm{R}_{\mathcal{H}_c}$ *classification region*). It turns out that the number of documents classified by $\mathcal{H}_c$ is $\sum_{(x,y)\in \mathrm{R}_{\mathcal{H}_c}} (\pi(x, y) + \nu(x, y))$. The border of the region $\mathrm{R}_{\mathcal{H}_c}$ is the *decision boundary* of $\mathcal{H}_c$.
As an example, the classification regions of the (ϕ-homogeneous) classifiers $\mathcal{H}_c = \{(1, 2)\}$ and $\mathcal{H}'_c = \{(1, 2), (2, 3)\}$ are those depicted in Figure 8.3. Here, the following should be noted:

1. the decision boundary of the atom $\mathcal{H}_c$ is a rectangle (left side of Figure 8.3), while that of the 2-order classifier $\mathcal{H}'_c$ is the overlapping of two rectangles, one for each atom (right side of Figure 8.3), and

2. the classification region of $\mathcal{H}_c$, which is a τ-specialization of $\mathcal{H}'_c$, is contained in the classification region of $\mathcal{H}_c$.

The above two statements can be generalized. In particular, concerning point (2), it can be easily verified that, for any two classifiers $\mathcal{H}'_c,\ \mathcal{H}_c$ such that $\mathcal{H}'_c \succeq_\tau \mathcal{H}_c$, the condition $\mathrm{R}_{\mathcal{H}'_c} \supseteq \mathrm{R}_{\mathcal{H}_c}$ holds,

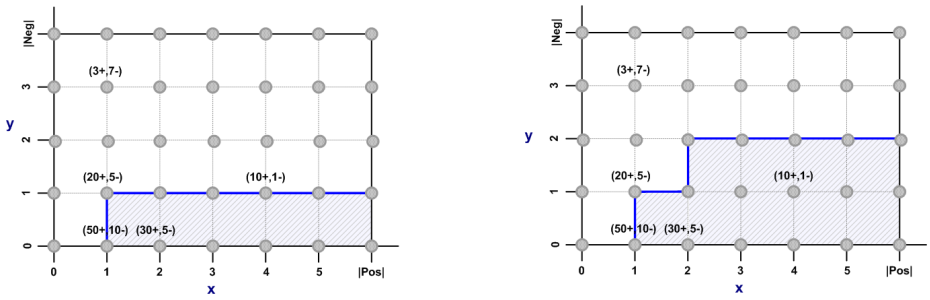

Figure 8.3: Decision boundaries of $\{(1,2)\}$ (left side) and $\{(1,2),(2,3)\}$ (right side)

and vice versa (it suffices to use the above definition of classification region along with Definition 8.10). As for point (1), we can state that the decision boundary of a classifier $\mathcal{H}_c^1 \vee \cdots \vee \mathcal{H}_c^r$ is a step-wise non-decreasing polyline in N^2 consisting alternately of vertical and horizontal segments. To see why, it suffices to observe the following:

a. the decision boundary of each single atom $\mathcal{H}_c^i = \{(p_i, n_i)\}$, $1 \leq i \leq r$, is a rectangle subtending the points (x, y) which satisfy the test conditions $p_i \leq x \leq |Pos|$ and $0 \leq y < n_i$, and

b. the r atoms $\mathcal{H}_c^1 \cdots \mathcal{H}_c^r$ are such that $p_{i-1} < p_i$ and $n_{i-1} < n_i$, for each $i \in [1, r]$ (see Proposition 8.19, Part 2).

Intuitively, the non-decreasingness of decision boundaries implies that documents which are less likely to belong to a category c (that is, documents with few positive features and many negative ones) are also less likely to be classified by $\mathcal{H}_c$. For an instance, consider two documents $d(x, y)$ and $d'(x', y)$, having x and x' positive features, respectively, with $x' \geq x$, and both containing the same number y of negative features. Intuitively, $d(x, y)$ is less likely to be a positive example for c than $d'(x', y)$ (as it holds less positive features, which are indicative of membership, for the same number of negative ones). On the other hand, since the boundary is non-decreasing, it also happens that $d(x, y)$ is less likely to fall within $\mathrm{R}_{\mathcal{H}_c}$ than $d'(x', y)$, that is, $d(x, y)$ is less likely to be classified by $\mathcal{H}_c$.

8.4.6 Remarks on the proposed language

The "family resemblance" metaphor. In a binary classification task there are two families (classes): the positive, call it P, and the negative, call it N. Let us assume that the atom p-of-$Pos \wedge \neg n$-of-Neg is used to characterize the members of P. Here, Pos is the set of features that such members

share, while the threshold p states how many of such features each member must hold. Symmetrically, Neg is the set of features shared by the members of the other family N. Actually, not all members, but more specifically only those that are most similar to the members of P (recall that, by Definition 8.1, the features in Neg are those that characterize members of N holding some features of the family P). Thus, an example that exhibits p positive features is a member of P provided that it holds less than n negative features. To use an analogy, imagine that the members of the Brown family hold at least two of the following features: green eyes, black hair and tallness. However, also in the White family there are members that are tall and have green eyes, but they also hold at least two of the following features: fair hair, long nose and high forehead. Thus, an individual that is tall and has green eyes belongs to the Brown family provided that he possesses less than two of such features (that would play the role of negative features for the Brown family with threshold $n = 2$).

On the expressivity of M-of-$N^{\{\neg,\vee\}}$. M-of-N hypotheses can be regarded as M-of-$N^{\{\neg,\vee\}}$ atoms with *only* positive features, i.e., atoms of the form p-of-Pos. Simple M-of-N hypotheses are often sufficient for the classification of new and unseen data, but it is well known that there are cases where the need for negative features cannot be avoided. A simple example is the following: document $\{t_0, t_1\}$ belongs to c, document $\{t_0, t_1, t_2\}$ belongs to c' and document $\{t_1, t_2\}$ belongs to c''. It is easy to recognize that this scenario can be modeled by the atoms $\mathcal{H}_c = \langle\{t_0\}, \{t_2\}, \{(1,1)\}\rangle$, $\mathcal{H}_{c'} = \langle\{t_0, t_2\}, \emptyset, \{(2,1)\}\rangle$, and $\mathcal{H}_{c''} = \langle\{t_1\}, \{t_0\}, \{(1,1)\}\rangle$, where the negative features are needed to discriminate among classes.

Although M-of-$N^{\{\neg,\vee\}}$ atoms surpass classical M-of-N hypotheses in expressive power, there are data sets that cannot be represented simply by atoms. As an example, assume that documents $d_1 = \{t_0\}$, $d_2 = \{t_0, t_1\}$ and $d_3 = \{t_0, t_1, t_2\}$ are associated with category c, while $d_4 = \{t_0, t_2\}$ is not. Intuitively, to correctly classify such data we need a hypothesis $\mathcal{H}_c$ stating the following: the occurrence of either t_0 or t_1 is sufficient in order for a document d be classified under c, provided that t_2 does not appear in d; but, if t_2 does appear in d, a stronger condition is needed, that is, both t_0 and t_1 must occur in d. We can easily recognize that $\mathcal{H}_c$ is the 2-order classifier $\langle\{t_0, t_1\}, \{t_2\}, \{(1,1), (2,2)\}$, and that no atomic equivalent classifier there exists.

However, though the proposed language improves the expressive power of M-of-N concepts, M-of-$N^{\{\neg,\vee\}}$ does not actually reach the full expressiveness of DNF. For an instance, there is no M-of-$N^{\{\neg,\vee\}}$ hypothesis capable of explaining the following data: $d_1 = \{t_0\}$, $d_2 = \{t_1, t_2\}$ and $d_3 = \{t_0, t_2\}$, with d_1 and d_2 belonging to class c and d_3 to its complement. It is easy to recognize that the reason for this limitation is that all atoms forming a hypothesis share the same sets of positive and negative features. As we will see in the next sections, the rationale for this choice is that it drastically restricts the search space. That is, effectiveness is traded-of against efficiency.

Why subsumption relations are important. As we have seen, the two relations $\succeq_\tau$ and $\succeq_\phi$ codify the intuitive notion of "more-general-than" between hypotheses. That is, given $\mathcal{H}_c^1$ and $\mathcal{H}_c^2$ such

that $\mathcal{H}_c^1 \succeq_\tau \mathcal{H}_c^2$, any example covered by H_c^2 is covered by $\mathcal{H}_c^1$ as well.
The idea of ordering the concept space by a "more-general-than" relation is not new in Inductive Logic Programming (see, e.g., [123]). What is actually original in our approach is the ordering along two dimensions, the feature and the threshold dimensions.
The ordering relations are important because they provide the learning algorithm with a means to *selectively* search the hypothesis space. For an instance, the search strategy can move towards a more general hypothesis whenever too few positive examples are covered by the current one or, viceversa, towards a more specific hypothesis if too negative examples are covered.
The implementation of a selective search requires the definition of suitable operators which, by exploiting the subsumption relations, enable the generalization/specialization of a hypothesis. This is what we will do in the next section.

8.5 Refinement operators

Informally, a refinement operator is a function which enables to "navigate" the space of the minimal classifiers through the partial order relations. We next provide two classes of refinement operators: *unary* and *binary* refinement operators.
Notation. For the sake of simplicity, in the following definitions we will often denote the set of minimal hypotheses $\mathbf{M}(\mathcal{F}(k), P, N,)$ simply by $\mathbf{M}$.

8.5.1 Unary refinement operators

A *unary refinement operator* is a non-deterministic function which returns a "neighbor" of $\mathcal{H}_c$ either in the ϕ-subsumption or in the τ-subsumption relationship. It is used to move a classifier "one step" upward or downward in either one of the two hierarchies.

Definition 8.32 (Unary Refinement Operators) A *unary refinement operator* is a non-deterministic function from $\mathbf{M}$ to $\mathbf{M}$. In particular, the *unary x-generalization* operator, denoted $\uparrow^x$, with $x \in \{\phi, \tau\}$, is a function such that: $\uparrow^x (\mathcal{H}_c) = \mathcal{H}_c$ if $\nexists \mathcal{H}'_c \in \mathbf{M}$ such that $\mathcal{H}'_c >_x \mathcal{H}_c$ (i.e., $\mathcal{H}_c$ is the top element); otherwise, $\uparrow^x (\mathcal{H}_c) = \mathcal{H}'_c$ where $\mathcal{H}'_c >_x \mathcal{H}_c$ and $\nexists \mathcal{H}''_c$ such that $\mathcal{H}'_c >_x \mathcal{H}''_c >_x \mathcal{H}_c$. The *unary x-specialization* operator $\downarrow^x$ is defined accordingly.

We first provide a constructive definition of both $\uparrow^\phi (\mathcal{H}_c)$ and $\downarrow^\phi (\mathcal{H}_c)$ in the ϕ-subsumption lattice. Informally, a direct ancestor of $\mathcal{H}_c$ in the ϕ-subsumption hierarchy is obtained from $\mathcal{H}_c$ either by adding to Pos a candidate positive term or by removing any term from Neg (a direct descendent is obtained in a dual way).

$\diamond$ `COMPUTATION of` $\uparrow^\phi (\mathcal{H}_c)$ `and` $\downarrow^\phi (\mathcal{H}_c)$. Given $\mathcal{H}_c = \langle Pos, Neg, \mathcal{T} \rangle$, compute:

$\uparrow^{\phi}(\mathcal{H}_c) = \mathcal{H}_c$ if $Pos = Pos^*(k)$ and $Neg = \emptyset$ (i.e., if $\mathcal{H}_c$ is the top element in the ϕ-subsumption lattice), otherwise

$$\uparrow^{\phi}(\mathcal{H}_c) = \begin{cases} \langle Pos \cup \{t\}, Neg, \mathcal{T}\rangle & \text{where } t \in Pos^*(k),\ or \\ \langle Pos, Neg \setminus \{t\}, \mathcal{T}\rangle & \text{where } t \in Neg \end{cases}$$

$\downarrow^{\phi}(\mathcal{H}_c) = \mathcal{H}_c$ if $Pos = \emptyset$ and $Neg = Neg^*(k)$, otherwise

$$\downarrow^{\phi}(\mathcal{H}_c) = \begin{cases} \langle Pos \setminus \{t\}, Neg, \mathcal{T}\rangle & \text{where } t \in Pos,\ or \\ \langle Pos, Neg \cup \{t\}, \mathcal{T}\rangle & \text{where } t \in Neg^*(k) \end{cases}$$

∎

Correctness of the computation of $\uparrow^{\phi}(\mathcal{H}_c)$ We restrict the proof to the correctness of the computation of $\uparrow^{\phi}(\mathcal{H}_c)$. The proof concerning $\downarrow^{\phi}(\mathcal{H}_c)$ follows a similar framework.
Let us start by proving $\uparrow^{\phi}(\mathcal{H}_c) = \langle Pos', Neg', \mathcal{T}\rangle$, where $Pos' = Pos \cup \{t\}$, $t \in Pos^*(k)$ and $Neg' = Neg$. First of all we note that, since both $Pos' \supseteq Pos$ and $Neg' = Neg$, $\uparrow^{\phi}(\mathcal{H}_c) \succeq_{\phi} \mathcal{H}_c$ holds - see Definition 8.5. Now assume by absurd the existence of $\mathcal{H}''_c = \langle Pos'', Neg'', \mathcal{T}\rangle$ such that $\uparrow^{\phi}(\mathcal{H}_c) \succeq_{\tau} H''_c \succeq_{\tau} H_c$. Thus, both $Pos' \supseteq Pos'' \supseteq Pos$ and $Neg' \subseteq Neg'' \subseteq Neg$. However, since Pos' and Pos differ for exactly one term, either $Pos' = Pos''$ or $Pos'' = Pos$ holds. Further, since $Neg = Neg'$, $Neg' = Neg'' = Neg$ holds as well. That is, either $\mathcal{H}''_c = \mathcal{H}_c$ or $\mathcal{H}''_c = \uparrow^{\phi}(\mathcal{H}_c)$, a contradiction.
Now let us prove $\uparrow^{\phi}(\mathcal{H}_c) = \langle Pos', Neg', \mathcal{T}\rangle$, where $Pos' = Pos$, $Neg' = Neg \setminus \{t\}$ and $t \in Neg$. Since both $Pos' = Pos$ and $Neg' \subseteq Neg$, $\uparrow^{\phi}(\mathcal{H}_c) \succeq_{\phi} \mathcal{H}_c$ holds - see Definition 8.5. Now, by absurd, assume that there exists $\mathcal{H}''_c = \langle Pos'', Neg'', \mathcal{T}\rangle$ such that $\uparrow^{\phi}(\mathcal{H}_c) \succeq_{\tau} H''_c \succeq_{\tau} H_c$. Since both $Pos' \supseteq Pos'' \supseteq Pos$ and $Pos = Pos'$, it follows that $Pos'' = Pos$. Moreover, since $Neg' \subseteq Neg'' \subseteq Neg$ and, further, Neg and Neg' differ for exactly one feature, either $Neg'' = Neg$ or $Neg'' = Neg'$. That is, either $\mathcal{H}''_c = \mathcal{H}_c$ or $\mathcal{H}''_c = \uparrow^{\phi}(\mathcal{H}_c)$, a contradiction.

Example 8.33 Given $\mathcal{H}_c = \langle \{t_0, t_1\}, \{t_2\}, \mathcal{T}\rangle$, let $t \in Pos^*(k)$ and $t' \in Neg^*(k)$ be two candidate features. Then, the following hypotheses are "neighbors" of $\mathcal{H}_c$ in the ϕ-subsumption hierarchy:

$$\uparrow^{\phi}(\mathcal{H}_c) = \langle \{t_0, t_1, t\}, \{t_2\}, \mathcal{T}\rangle,\ \uparrow^{\phi}(\mathcal{H}_c) = \langle \{t_0, t_1\}, \emptyset, \mathcal{T}\rangle$$

$$\downarrow^{\phi}(\mathcal{H}_c) = \langle \{t_1\}, \{t_2\}, \mathcal{T}\rangle,\ \downarrow^{\phi}(\mathcal{H}_c) = \langle \{t_0, t_1\}, \{t_2, t'\}, \mathcal{T}\rangle.$$

Let us now see how a neighbor $\uparrow^{\tau}(\mathcal{H}_c)$ or $\downarrow^{\tau}(\mathcal{H}_c)$ of $\mathcal{H}_c$ in the τ-subsumption lattice is computed. Clearly, to obtain, say, $\uparrow^{\tau}(\mathcal{H}_c)$, we have to replace in $\mathcal{H}_c$ the threshold set $\mathcal{T}$ by an immediate ancestor $\uparrow \mathcal{T}$ in the τ-subsumption lattice. So as the problem reduces to the computation of $\uparrow \mathcal{T}$.

Non-deterministic function $\uparrow \mathcal{T}$

Input: threshold bounds P and N; a minimal threshold set $\mathcal{T} = \{\tau_1, ..., \tau_k\}$, where $\tau_i = (p_i, n_i)$ for each $i \in [1, k]$ and $p_i < p_{i+1},\ n_i < n_{i+1}$, for each $i \in [1, k)$ (see Proposition 8.19)
Output: a direct ancestor $\uparrow \mathcal{T}$ of $\mathcal{T}$;

function $NewElement(X, Y)$
1. **if** $(p_x > p_y)$ **then** swap $X = (p_x, n_x)$ and $Y = (p_y, n_y)$
2. $\delta^+ = |p_y - p_x|,\ \delta^- = |n_y - n_x|$;
3. **if** $(\delta^+ > 1\ and\ \delta^- > 1)$ **then** compute the most specific threshold pair (p, n) such that
4. $p_x < p < p_y$ and $n_x < n < n_y$, i.e., $(p, n) = (p_y - 1, n_x + 1)$
5. **else** compute the most specific threshold pair (p, n) such that $p_x \leq p \leq p_y$
6. and $n_x \leq n \leq n_y$ and:
7. **if** $\delta^+ > 1$ **then** $(p, n) \succeq_\tau Y$; set $(p, n) = (p_y - 1, n_y)$;
8. **else if** $\delta^- > 1$ **then** $(p, n) \succeq_\tau X$; set $(p, n) = (p_x, n_x + 1)$
9. **else** $(p, n) \succeq_\tau X$ and $(p, n) \succeq_\tau Y$; set $(p, n) = (p_x, n_y)$
10. **return** $\{(p, n)\}$.

11. **begin**
12. **if** $\mathcal{T} = \{(0, N)\}$ (i.e., $\mathcal{T}$ is the top of the lattice) **return** $\emptyset$;
13. $\tau_0 = (p_0, n_0) = (-1, 0)$; $\tau_{k+1} = (p_{k+1}, n_{k+1}) = (P + 1, N + 1)$;
14. randomly select $i \in [1, k]$;
15. **if** $i = 1$ and $p_i = 0$ **then** $adj = \tau_{i+1}$ // right adjacent
16. **else if** $i = k$ and $n_i = N$ **then** $adj = \tau_{i-1}$; // left adjacent
17. **else** randomly select $adj \in \{\tau_{i-1}, \tau_{i+1}\}$; ;
18. **return** $\uparrow \mathcal{T} = Minimize(\mathcal{T} \cup NewElement(\tau_i, adj))$;

Figure 8.4: Pseudo code for the random selection of a direct ancestor of a threshold set in the τ-subsumption lattice

◇ `COMPUTATION of` $\uparrow^\tau (\mathcal{H}_c)$ `and` $\downarrow^\tau (\mathcal{H}_c)$. Given $\mathcal{H}_c = \langle Pos, Neg, \mathcal{T} \rangle$, compute $\uparrow^\tau (\mathcal{H}_c)$ and $\downarrow^\tau (\mathcal{H}_c)$ as follows

$$\uparrow^\tau (\mathcal{H}_c) = \langle Pos, Neg, \uparrow \mathcal{T} \rangle \text{ and } \downarrow^\tau (\mathcal{H}_c) = \langle Pos, Neg, \downarrow \mathcal{T} \rangle$$

where the non-deterministic operator $\uparrow$ (resp. $\downarrow$) applied to $\mathcal{T}$ returns an immediate ancestor (resp. descendant) of $\mathcal{T}$ in the τ-subsumption hierarchy. $\uparrow \mathcal{T}$ is constructed from $\mathcal{T}$ by the algorithm of Figure 8.4 (we do not report the dual algorithm for $\downarrow \mathcal{T}$ for space reason). ∎

For a description of the algorithm of Figure 8.4 the reader is referred to 8.5.2.

8.5.2 The non-deterministic function $\uparrow \mathcal{T}$

Next we give a description of the algorithm of Figure 8.4. It creates a direct ancestor $\uparrow \mathcal{T}$ of $\mathcal{T} = \{\tau_1, \ldots, \tau_k\}$, where $\tau_i = (p_i, n_i)$, $1 \leq i \leq k$, by applying to $\mathcal{T}$ only local changes. We preliminarily recall that, by Proposition 8.19, the minimality of $\mathcal{T}$ requires that $p_i < p_j,\ n_i < n_j$, or vice versa, for each $i, j \in [1, k]$. In the following discussion we assume $p_i < p_j,\ n_i < n_j$ (read "τ_i *smaller than* τ_j") if $i < j$.

The algorithm starts by checking the condition $\mathcal{T} = \{0, N\}$, that is, if $\mathcal{T}$ is the top element. Clearly, in such a case no direct ancestor exists and the algorithm returns the empty set (line 12). Since the algorithm works on the "distance" between two elements of $\mathcal{T}$, in order not to exclude the first element τ_1 and the last one τ_k, two fictitious elements are defined, namely, $\tau_0 = (-1, 0)$ and $\tau_{k+1} = (P + 1, N + 1)$ (line 2). Then, an element τ_i of $\mathcal{T}$, along with one adjacent (left or right), are randomly selected at lines 14-17. Of course, if $i = 1$, i.e., $\tau_i = (p_i, n_i)$ is the smallest element of $\mathcal{T}$, and $p_i = 0$, then the adjacent of τ_i will be the right one, i.e., τ_{i+1} (line 15). Symmetrically, if $i = k$, i.e., $\tau_k = (p_k, n_k)$ is the greatest element of $\mathcal{T}$, and $n_i = N$ (recall that N is the negative threshold bound), then the adjacent of τ_i will be the left one, i.e., τ_{i-1} (line 16). Then, the function $NewElement$ is invoked by passing τ_i and the selected adjacent adj (line 18). This function works as follows. First, it orders the element $X = (p_x, n_x)$ and its adjacent $Y = (p_y, n_y)$ in such a way that X is the smallest one (line 1). Then, the distances δ^+ and δ^- between X and Y are computed (line 2). Now, there are two ways for constructing an immediate ancestor of $\mathcal{T}$: either (1) by adding a suitable element τ to $\mathcal{T}$, or (2) by replacing an element τ_i of $\mathcal{T}$ by the most specific τ which generalizes τ_i. Which one of the two alternatives is applied depends on the distances δ^+ and δ^-. In particular, if both distances are greater than one (line 3 - intuitively, this means that there is "enough room" in between X and Y to accommodate a new element in $\mathcal{T}$), then the most specific threshold pair (p, n) such that $p_x < p < p_y$ and $n_x < n < n_y$ is computed, i.e., $p = p_y - 1$ and $n = n_x + 1$ (the most specific (p, n) is the one with the highest possible p value and the lowest possible n value). Then, $\uparrow \mathcal{T}$ is set to $Minimize(\mathcal{T} \cup \{(p, n)\})$ (line 18), where $Minimize$ is the function sketched in Figure 8.2.

As an example, let us consider the classifier $\mathcal{H}_c = \langle Pos, Neg, \mathcal{T} \rangle$, where $\mathcal{T} = \{\tau_1, \tau_2\}$, $\tau_1 = \{(0, 1)\}$ and $\tau_2 = \{(2, 3)\}$, and assume that the threshold bounds are $P = 2$ and $N = 3$. Note that τ_1 is smaller than τ_2. Now, suppose that the algorithm at line 13 selects $i = 2$ (i.e., $\tau_2 = (2, 3)$). Since τ_2 is the greatest element of $\mathcal{T}$ and $n_2 = N$, the algorithm choses the left adjacent, i.e., τ_1 (line 15). Then, the function is invoked (line 18) and the distances $\delta^+ = p_2 - p_1 = 2$ and $\delta^- = n_2 - n_1 = 2$ are computed (line 3). Since $(\delta^+ > 1\ and\ \delta^- > 1)$ holds (line 3), the function sets $(p, n) = (p_2 - 1, n_1 + 1) = (1, 2)$ (line 4) and returns it to the main. The resulting threshold set is $\uparrow \mathcal{T} = Minimize(\mathcal{T} \cup \{(1, 2)\})$, that is, $\{(0, 1), (1, 2), (2, 3)\}$, which is an immediate ancestor of $\mathcal{T}$ (see Figure 8.1).

If the condition on the distances at line 3 does not apply, then the most specific threshold pair (p, n) which generalizes either X or Y is computed. Again, this is done depending on the values of two distances δ^+ and δ^-. In particular, if $\delta^+ > 1$, then the algorithm generates the most specific element (p, n) which generalizes Y, i.e., $(p, n) = (p_y - 1, n_y)$. On the contrary, if $\delta^- > 1$, then the algorithm generates the most specific element (p, n) which generalizes X, i.e., $(p, n) = (p_x, n_x + 1)$. Finally, if none of the above conditions hold (line 9), the algorithm generates the most specific element (p, n) which generalizes both X and Y, i.e., $(p, n) = (p_x, n_y)$.
As an example, assume that $\mathcal{T} = \{\tau_1, \tau_2\}$, with $\tau_1 = \{(1, 1)\}$ and $\tau_2 = \{(2, 2)\}$, and let $i = 2$ (i.e., $\tau_2 = (2, 2)$). Suppose that the chosen adjacent is the left one, i.e., τ_1. Since $\delta^+ = \delta^- = 1$, none of the conditions at lines 3, 7 and 8 applies. Thus $(p, n) = (p_1, n_2) = (1, 2)$ is computed at line 9 and returned to the main. This element is then added to $\mathcal{T}$ (line 18) and, after minimization, the algorithm returns $\uparrow \mathcal{T} = \{(1, 2)\}$, which is an immediate ancestor of $\mathcal{T}$ (see Figure 8.1).

8.5.3 Binary refinement operators

Binary refinement operators are aimed at exploiting the lattice structure of both the ϕ-subsumption and the τ-subsumption hierarchies. In particular, given two classifiers, they return a classifier which is either the lub or the glb of the two classifiers in any of the two subsumption lattices, depending on whether a generalization or a specialization is needed, respectively.

Definition 8.34 (Binary refinement operators) A binary refinement operator is a function from $\mathbf{M} \times \mathbf{M}$ to $\mathbf{M}$. Let classifiers $\mathcal{H}_c^1 = \langle Pos_1, Neg_1, \mathcal{T}_1 \rangle$ and $\mathcal{H}_c^2 = \langle Pos_2, Neg_2, \mathcal{T}_2 \rangle$ be given. There are two *binary generalization operators*, the τ-generalization $\bigvee_\tau$ and the ϕ-generalization $\bigvee_\phi$, defined as follows:

$$\bigvee_\tau(\mathcal{H}_c^1, \mathcal{H}_c^2) = \langle Pos_1, Neg_1, \sqcup(\mathcal{T}_1, \mathcal{T}_2) \rangle$$

$$\bigvee_\phi(\mathcal{H}_c^1, \mathcal{H}_c^2) = \langle Pos_1 \cup Pos_2, Neg_1 \cap Neg_2, \mathcal{T}_1 \rangle$$

and two *binary specialization operators* $\bigwedge_\tau$ and $\bigwedge_\phi$ defined as follows:

$$\bigwedge_\tau(\mathcal{H}_c^1, \mathcal{H}_c^2) = \langle Pos_1, Neg_1, \sqcap(\mathcal{T}_1, \mathcal{T}_2) \rangle$$

$$\bigwedge_\phi(\mathcal{H}_c^1, \mathcal{H}_c^2) = \langle Pos_1 \cap Pos_2, Neg_1 \cup Neg_2, \mathcal{T}_1 \rangle.$$

It should be noted that all the above operators are not commutative. In fact, $\bigvee_x(\mathcal{H}_c^1, \mathcal{H}_c^2)$, with $x \in \{\tau, \phi\}$, yields a generalization of $\mathcal{H}_c^1$ (through $\mathcal{H}_c^2$), and $\bigwedge_x(\mathcal{H}_c^1, \mathcal{H}_c^2)$ yields a specialization of $\mathcal{H}_c^1$ (through $\mathcal{H}_c^2$).

Intuitively, $\bigvee_\tau(\mathcal{H}_c^1, \mathcal{H}_c^2)$ is the least upper bound of $\mathcal{H}_c^1$ and the ϕ-homogeneous classifier having the same threshold set of $\mathcal{H}_c^2$ (see Figure 8.5). Dually, $\bigvee_\phi(\mathcal{H}_c^1, \mathcal{H}_c^2)$ is the least upper bound of $\mathcal{H}_c^1$ and the τ-homogeneous classifier having the same feature sets of $\mathcal{H}_c^2$ (the specialization operators are defined accordingly).

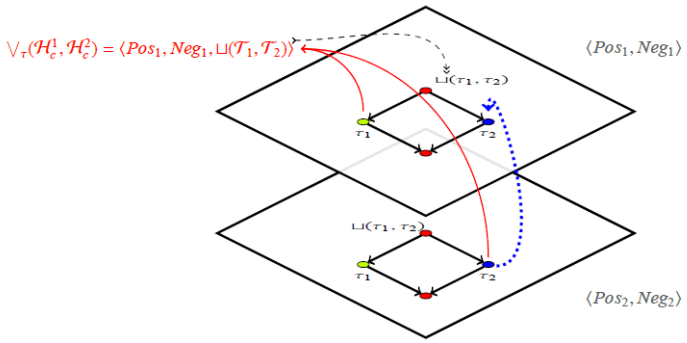

Figure 8.5: Given $\mathcal{H}_c^1 = \langle Pos_1, Neg_1, \mathcal{T}_1\rangle$ and $\mathcal{H}_c^2 = \langle Pos_2, Neg_2, \mathcal{T}_2\rangle$, the hypothesis $\bigvee_\tau(\mathcal{H}_c^1, \mathcal{H}_c^2)$ is the least upper bound of $\mathcal{H}_c^1$ and the ϕ-homogeneous classifier $\langle Pos_1, Neg_1, \mathcal{T}_2\rangle$, i.e., $\bigvee_\tau(\mathcal{H}_c^1, \mathcal{H}_c^2) = \langle Pos_1, Neg_1, \sqcup(\mathcal{T}_1, \mathcal{T}_2)\rangle$

Example 8.35 Consider $\mathcal{H}_c^1 = \langle Pos_1, Neg_1, \mathcal{T}_1\rangle$ and $\mathcal{H}_c^2 = \langle Pos_2, Neg_2, \mathcal{T}_2\rangle$, where $\mathcal{T}_1 = \{(2,2)\}$ and $\mathcal{T}_2 = \{(1,1)\}$. According to Definition 8.34, we have that

$$\bigvee_\tau(\mathcal{H}_c^1, \mathcal{H}_c^2) = \langle Pos_1, Neg_1, \sqcup(\mathcal{T}_1, \mathcal{T}_2)\rangle) = \langle Pos_1, Neg_1, \{(1,1),(2,2)\}\rangle$$

$$\bigwedge_\tau(\mathcal{H}_c^1, \mathcal{H}_c^2) = \langle Pos_1, Neg_1, \sqcap(\mathcal{T}_1, \mathcal{T}_2)\rangle) = \langle Pos_1, Neg_1, \{(2,1)\}\rangle.$$

It is easily verified that $\bigvee_\tau(\mathcal{H}_c^1, \mathcal{H}_c^2)$ is a generalization of $\mathcal{H}_c^1$, while $\bigwedge_\tau$ is a specialization of $\mathcal{H}_c^1$. Now, assume that $Pos_1 = \{t_1, t_2\}$, $Neg_1 = \{t_3, t_4\}$, $Pos_2 = \{t_2, t_5\}$, $Neg_2 = \{t_3\}$. We generalize $\mathcal{H}_c^1$ (through $\mathcal{H}_c^2$) by using the $\bigvee_\phi$ operator as follows:

$$\bigvee_\phi(\mathcal{H}_c^1, \mathcal{H}_c^2) = \langle Pos_1 \cup Pos_2, Neg_1 \cap Neg_2, \mathcal{T}_1\rangle) = \langle \{t_1, t_2, t_5\}, \{t_3\}, \mathcal{T}_1\rangle$$

and specialize $\mathcal{H}_c^1$ (through $\mathcal{H}_c^2$) by $\bigwedge_\phi$ as follows:

$$\bigwedge_\phi(\mathcal{H}_c^1, \mathcal{H}_c^2) = \langle Pos_1 \cap Pos_2, Neg_1 \cup Neg_2, \mathcal{T}_1\rangle) = \langle \{t_2\}, \{t_3, t_4\}, \mathcal{T}_1\rangle.$$

It is easy to recognize that both $\bigvee_\phi(\mathcal{H}_c^1, \mathcal{H}_c^2) \succeq_\phi \mathcal{H}_c^1$ and $\mathcal{H}_c^1 \succeq_\phi \bigwedge_\phi(\mathcal{H}_c^1, \mathcal{H}_c^2)$ hold.

Proposition 8.36 Given classifiers $\mathcal{H}_c^1$ and $\mathcal{H}_c^2$, the following holds:

- $\bigvee_x(\mathcal{H}_c^1, \mathcal{H}_c^2) \succeq_x \mathcal{H}_c^1$.
- $\mathcal{H}_c^1 \succeq_x \bigwedge_x(\mathcal{H}_c^1, \mathcal{H}_c^2)$.

where $x \in \{\tau, \phi\}$.

Proof Next we show that, given classifiers $\mathcal{H}_c^1$ and $\mathcal{H}_c^2$, the following holds: $\bigvee_x(\mathcal{H}_c^1, \mathcal{H}_c^2) \succeq_x \mathcal{H}_c^1$ and $\mathcal{H}_c^1 \succeq_x \bigwedge_x(\mathcal{H}_c^1, \mathcal{H}_c^2)$, where $x \in \{\tau, \phi\}$. By Definition 8.34, we have that $\bigvee_x(\mathcal{H}_c^1, \mathcal{H}_c^2) = lub_x(\mathcal{H}_c^1, \mathcal{H}_c^{1,2})$. Since $lub_x(\mathcal{H}_c^1, \mathcal{H}_c^{1,2}) \succeq_x \mathcal{H}_c^1$, it turns out that $\bigvee_x(\mathcal{H}_c^1, \mathcal{H}_c^2) \succeq_x \mathcal{H}_c^1$. Dually, from $\bigwedge_x(\mathcal{H}_c^1, \mathcal{H}_c^2) = glb_x(\mathcal{H}_c^1, \mathcal{H}_c^{1,2})$ and $\mathcal{H}_c^1 \succeq_x glb_x(\mathcal{H}_c^1, \mathcal{H}_c^{1,2})$, $\mathcal{H}_c^1 \succeq_x \bigwedge_x(\mathcal{H}_c^1, \mathcal{H}_c^2)$ follows.

8.6 Learning Problem and Complexity

Before providing an effective algorithm for the learning of classifiers, in this section we give a definition of the learning problem and show its complexity.
The goal is to find, for each category $c \in \mathcal{C}$, a (minimal) hypothesis $\mathcal{H}_c \in \mathbf{M}(\mathcal{F}(k), P, N)$ that best fits the training data. To this end, we assume that categories in $\mathcal{C}$ are mutually independent, so as the whole learning task consists of $|\mathcal{C}|$ independent binary sub-tasks, one for each category.
To assess $\mathcal{H}_c$ we use the F-measure. This is a measure that trades off precision Pr versus recall Re and is defined as the harmonic mean of Pr and Re as follows[2]:

$$F = \frac{2PrRe}{Pr + Re}. \tag{8.2}$$

Let us denote by $F(\mathcal{H}_c, T)$ the F-measure obtained by $\mathcal{H}_c$ when it is applied to the documents of the training set T. Now, the learning problem can be formulated as the following optimization problem.

Definition 8.37 (Learning problem) Let the feature space $\mathcal{F}(k)$ and the threshold bounds P, N be given. The learning problem is to find a (minimal) classifier $\mathcal{H}_c \in \mathbf{M}(\mathcal{F}(k), P, N)$ that maximizes the F-measure $F(\mathcal{H}_c, T)$ of $\mathcal{H}_c$ over the training set T.

The above learning problem is essentially an instance of Inductive Logic Programming (ILP) [98], which deals with the general problem of inducing logic programs from examples in the presence of background knowledge. It is well known that ILP problems are computationally intractable.

Proposition 8.38 The decision version of the learning problem is NP-complete.

[2]This is also known as the F1 measure, because recall and precision are evenly weighted.

Proof We have to prove that the decision version of the GAMoN learning problem is NP-complete. The proof is by a reduction from the Knapsack problem. Given an atom $\mathcal{H}_c = \langle Pos, Neg, \{(p,n)\}\rangle$, let $S \subseteq T$ be the set of training documents classified by $\mathcal{H}_c$ under c, i.e., $S = \{d \in \mathcal{D}\ s.t.\ |d \cap Pos| \geq p\ \wedge\ |d \cap Neg| < n\}$. *Precision* is defined as the probability that a document in S is also in the training set T_c of c, i.e.,

$$Pr(\mathcal{H}_c, T) = \frac{|S \cap T_c|}{|S|} \tag{8.3}$$

and *Recall* is defined as the probability that a document in T_c is also in S, i.e.,

$$Re(\mathcal{H}_c, T) = \frac{|S \cap T_c|}{|T_c|}. \tag{8.4}$$

By replacing equations (8.3) and (8.4) into equation (8.2), after some algebra, we get the following formulation of the objective function

$$F(\mathcal{H}_c, T) = \frac{2 \cdot a}{b + |T_c|}$$

where $a = |S \cap T_c|$ and $b = |S \setminus T_c|$. Hence, to maximize $F(\mathcal{H}_c, T)$ we want a to be as large as possible, while keeping b bound to some given value (note that $|T_c|$ is a constant). Thus, the problem of learning an atomic classifier, in its recognition version, can be formulated as follows:
LEARN-ATOM-DECISION (LAD): Given the training set T, the feature space $\langle Pos_c^*(k), Neg_c^*(k)\rangle$ and two positive integers U and V, does there exist a hypothesis $\mathcal{H}_c = \langle Pos, Neg, \{(p,n)\}\rangle$ over $\langle Pos_c^*(k), Neg_c^*(k)\rangle$ such that $a \geq U$ and $b \leq V$? That is, does there exist a hypothesis which is consistent with at least a positive examples and is not consistent with at most b negative examples?
Now KNAPSACK is the following NP-complete problem: Given $2n+2$ positive integers $w_1, \cdots, w_n$, $v_1, \cdots, v_n$, W and Z, does there exist $X \subseteq \{1, \cdots n\}$ such that $\sum_{i \in X} w_i \leq W$ and $\sum_{i \in X} v_i \geq Z$?
We claim KNAPSACK polynomially reduces to LAD. To see this, suppose $I = (w_1, \cdots, w_n, v_1, \cdots, v_n, W, Z)$ is an instance of KNAPSACK. Make the following instance for LAD: (a) $U = Z$ and $V = W$; (b) $\langle Pos_c^*(k), Neg_c^*(k)\rangle = \langle \{t_1, \cdots, t_n\}, \emptyset\rangle$, i.e., the feature space consists of n positive candidate features and no negative candidate feature; (c) the training set T is such that:

c.1 $\Theta(t_i) \cap \Theta(t_j) = \emptyset$, for each $t_i, t_j \in \{t_1, \cdots, t_n\}$, and

c.2 $v_i = |\Theta(t_i) \cap T_c|$ and $w_i = |\Theta(t_i) \setminus T_c|$, for each $i \in [1, n]$

where $\Theta(t_i)$ denotes the set of examples (documents) in T where term t_i occurs. From point (c.1) above it follows that each document contains at most one positive candidate term. Further, from points (c.1) and (c.2) it turns out that, for a given $Pos \subseteq \{t_1, \cdots, t_n\}$, the following holds: $a = \sum_{t \in Pos} v_i$ and $b = \sum_{t \in Pos} w_i$. Thus, LAD turns out to be the following problem: "does there

exist $\mathcal{H}_c = \langle Pos, \emptyset, \{(1, *)\}\rangle$ such that $\sum_{t\in Pos} v_i \geq V$ and $\sum_{t\in Pos} w_i \leq C$ (the symbol "*" stands for "immaterial", as $Neg = \emptyset$)?" Or, equivalently: "does there exist $X \subseteq \{1, \cdots n\}$ such that $\sum_{i\in X} v_i \geq V$ and $\sum_{i\in X} w_i \leq C$?" Clearly, the answer to this LAD is "yes" iff I is an instance of KNAPSACK, then proving our claim. To conclude the proof it suffices to notice that verifying a YES instance of LAD requires polynomial time. Hence, problem LAD, i.e., the problem of deciding whether there exists an atom satisfying the constraints $a \geq U$ and $b \leq V$, is NP-complete. It is immediate to realize that the decision version of the learning problem (see Definition 8.37) is NP-complete as well.

The theory of *PAC-learnability*, first proposed by Valiant in [126], provides a model of *approximated* polynomial learning where the polynomially bound amount of resources (both number of examples and computational time) is traded-off against the accuracy of the induced hypothesis. However, as shown by the above proposition, there is no algorithm that produces a consistent M-of-$N^{\{\neg,\vee\}}$ hypothesis on p examples in time polynomial in p, so as M-of-$N^{\{\neg,\vee\}}$ hypotheses are not PAC-learnable (this should not be surprising, given that M-of-N concepts are not PAC-learnable - see Pitt and Valiant [102]).

8.7 Learning a Classifier: a GA-based approach

So far, we have seen the structural properties of the M-of-$N^{\{\neg,\vee\}}$ hypothesis space and designed a set of refinement operators that are the search abstract tools. Further, we have defined the learning problem and showed that it is computationally difficult. In this section we provide an effective algorithm for learning classifiers in the M-of-$N^{\{\neg,\vee\}}$ hypothesis space. In particular, we propose a heuristic approach based on a Genetic Algorithm (GA).

A GA represents a well known and powerful domain-independent search technique based on natural evolutionary operators. A standard GA can be regarded as composed of three basic elements: (1) A *population*, i.e., a set of candidate solutions (classifiers), called *individuals* or *chromosomes*, that will evolve during a number of iterations (*generations*); (2) a *fitness function* used to assign a score to each individual of the population; (3) an evolution mechanism based on operators such as *elitism*, *selection*, *crossover* and *mutation*. A comprehensive description of GAs can be found in [10].

GAs showed to be well suited for learning classification rules (see, e.g., [19, 129, 132]) as well as M-of-N hypotheses [82], as they perform a thorough search of the hypothesis space, not limited by any greedy search bias. However, GAs also have some disadvantages for rule discovery. For instance, conventional genetic operators, such as crossover and mutation, are normally applied without directly trying to optimize the quality of the new candidate solution by exploiting the structure of the hypothesis space. A recent research trend aimed at overcoming this drawback is that of combining the standard search strategy of GAs with that of *task-specific* genetic operators which

incorporate the knowledge about the specific application [28, 56, 57, 90] (here, by "application" we mean the task of inducing classification rules).
Next we present GAMoN, the task-specific GA designed to induce M-of-$N^{\{\neg,\vee\}}$ hypotheses. As we will see, GAMoN relies on a search strategy where *ad hoc*, selective reproduction operators, aimed at exploiting the structure of the hypothesis space, are combined with standard ones.
Detecting the "best" hypothesis space $\mathbf{M}(\mathcal{F}(k), P, N)$ to be explored by GAMoN is a fundamental task which strongly affects the quality of the learning process. In principle, we might either (1) manage the model parameters k, P and N (which uniquely determines the hypothesis space) as parameters to be manually tuned, or (2) embed them in the evolutive dynamics of the GA, letting it to adaptively evolve the best values. GAMoN incorporates this latter approach. To this end, evolution relies on a number of competing sub-populations $S(k_1, P_1, N_1), \cdots, S(k_n, P_n, N_n)$, where each $S(k_i, P_i, N_i)$ consists of individuals encoding classifiers in the same hypothesis space $\mathbf{M}_i(\mathcal{F}(k_i), P_i, N_i)$, $1 \leq i \leq n$.
A preliminary step for the creation of the sub-populations $S(k_1, P_1, N_1), \cdots, S(k_n, P_n, N_n)$ is the detection of a suitable range $[k_{min}, k_{max}]$ for the feature space dimensionality k_i of each subpopulation. This is the subject of the next subsection. Afterward, we will discuss on individual encoding and reproduction operators. Then, we report a detailed description of the genetic algorithm GAMoN and, finally, we provide some remarks on the proposed GA.

8.7.1 Detecting the feature space dimensionality

The feature space $\mathcal{F}(k)$ provides the basic symbols from which the classifiers of a given hypothesis space $\mathbf{M}(\mathcal{F}(k), P, N)$ are constructed. Behind its definition there is the implicit assumption that only the selected terms are representative of the category being learned, while the rest are redundant. Thus, predicting the right value of the dimensionality k is a crucial step. On one hand, a reduced feature space is desirable as redundant or noisy features may "deceive" the learning algorithm and have detrimental effect on classification results (this is particularly true in the text classification task, where data sets are usually noisy and ambiguous). Further, reducing the number of features makes the learning process more efficient (especially in the evolutionary approach, where large feature spaces may entail large individuals and, thus, more match operations). On the other hand, an aggressive feature selection might discard features that carry essential information.
Next we provide a criterion, inspired to the one proposed in [53], for detecting a range of dimensionality values based on the statistical characteristics of the data set at hand.

Definition 8.39 (Dimensionality range) We are given the vocabulary V_c of category c and a scoring function σ. We define the *dimensionality range* $[k_{min}, k_{max}]$ for category c as follows:

$$k_{min} = |\{t \in V_c \mid \sigma(t, c) \geq m + s\}|$$

$$k_{max} = |\{t \in V_c \mid \sigma(t,c) \geq m + 3s\}|$$

where $\sigma(t,c)$ is the score of feature $t \in V_c$ w.r.t. category c, and m and s are the average and standard deviation of the scoring values, respectively.

We notice that the above definition is essentially aimed at selecting a good set $Pos_c^*(k)$ of candidate positive features (recall that $Neg_c^*(k)$ consists of terms *co-occurring* with terms in $Pos_c^*(k)$ - see Definition 8.1). Indeed, to determine k_{min} (resp. k_{max}) we compute the scoring function σ for all features in V_c, and then count the number of features whose score is higher than 1 (resp. 3) standard deviations above the average, i.e., features with high discriminating power.

8.7.2 Individual Encoding

Given a hypothesis space $\mathbf{M}(\mathcal{F}(k), P, N)$, a candidate (minimal) classifier $\mathcal{H}_c = \langle Pos, Neg, \mathcal{T} \rangle \in \mathbf{M}(\mathcal{F}(k), P, N)$ is encoded by a bit string $I = \langle I^+, I^-, I^{\succeq_\tau} \rangle$, where:

1. the positive component I^+ is used to encode $Pos \subseteq Pos_c^*(k)$. It is made of k bits, each associated with a candidate feature $t_i \in Pos_c^*(k)$. A '1' or '0' in the gene $I^+[t_i]$, $1 \leq i \leq k$, indicates whether or not $t_i \in Pos_c^*(k)$ belongs to Pos.

2. The negative component I^- is used to encode $Neg \subseteq Neg_c^*(k)$. It is made of k bits, each associated with a candidate feature $t_i \in Neg_c^*(k)$. A '1' or '0' in the gene $I^-[t_i]$, $1 \leq i \leq k$, indicates whether or not the i-th candidate feature t_i belongs to Neg.

3. The threshold component $I^{\succeq_\tau}$ is used to encode the threshold set $\mathcal{T}$. The encoding of $\mathcal{T}$ relies on a straightforward binary representation of all pairs $(p,n) \in \mathcal{T}$, with $0 \leq p \leq P$ and $0 < n \leq N$. One additional bit for each element (p,n) is used to represent presence/absence of that element. Thus, the length of $I^{\succeq_\tau}$ is $Min(P+1, N)(\lceil log(N(P+1)) \rceil + 1)$, where $Min(P+1, N)$ is the maximum order of a classifier with threshold bounds P and N (see Proposition 8.31). In the following we will denote by enc the encoding function, i.e., $I^{\succeq_\tau} = enc(\mathcal{T})$.

It turns out that the length $L(I)$ of I is the following function of k, P and N

$$L(I) = L(I^+) + L(I^-) + L(I^{\succeq_\tau}) = 2k + Min(P+1, N)(\lceil log(N(P+1)) \rceil + 1).$$

Clearly, individuals encoding classifiers in the same hypothesis space $\mathbf{M}(\mathcal{F}(k), P, N)$ are of equal length.

Example 8.40 Let the hypothesis space $\mathbf{M}(\mathcal{F}(k), P, N)$ be given, where $k = 50$, $P = 2$ and $N = 3$. According to Proposition 8.31, the maximum order of a classifier is $Min(P+1, N) =$

$Min(3,3) = 3$ (see also Example 8.27). Thus, an individual encoding a classifier $\langle Pos, Neg, \mathcal{T} \rangle \in \mathbf{M}(\mathcal{F}(k), P, N)$ consists of $2k = 100$ bits needed to represent sets Pos and Neg, and further $Min(P+1, N)(\lceil log(N(P+1)) \rceil + 1) = 15$ bits to encode the threshold set $\mathcal{T}$.

8.7.3 Fitness

The performance measure used for evaluating the fitness of an individual is the objective function of the learning problem (see Definition 8.37).

Definition 8.41 (Fitness) We are given a chromosome I, encoding classifier $\mathcal{H}_c$, and the training set T. The fitness of I is $F(\mathcal{H}_c, T)$.

8.7.4 Task-specific GA operators and stochastic refinement

Next we propose some application-specific reproduction operators as an implementation of the refinement operators defined in Section 8.5. Such operators provide a concrete means whereby the learning algorithm selectively searches the hypothesis space. In particular, we next define two classes of Generalizing/Specializing (GS) operators: GS Crossover and GS Mutation.

Generalizing/Specializing Crossover

Crossover is the operation of swapping genetical material between two individuals (parents). GS crossover (GSX) is a special kind of crossover aimed at making a classifier more general or more specific.
The GSX operators we are defining are an application of the binary refinement operators given by Definition 8.34. As we have seen, they combine two classifiers of the same hypothesis space and provide a new classifier in the same space. Thus, GSX operators combine two parents belonging to the same sub-population (i.e., encoding classifiers in the same hypothesis space) and yields an individual in the same sub-population. Therefore, they operate on individuals of equal length (and isomorphic).

Notation. With a small abuse of notation, in the following we will denote by $\bigvee_x(I_1, I_2)$ and $\bigwedge_x(I_1, I_2)$ the individuals encoding the classifiers $\bigvee_x(\mathcal{H}_c^1, \mathcal{H}_c^2)$ and $\bigwedge_x(\mathcal{H}_c^1, \mathcal{H}_c^2)$, respectively, where $x \in \{\tau, \phi\}$ and I_i is the binary encoding of $\mathcal{H}_c^i$ $(1 \leq i \leq 2)$. Further, we write $I_1 \succeq_x I_2$ if $\mathcal{H}_c^1 \succeq_x \mathcal{H}_c^2$.

Definition 8.42 (GSX operators) We are given individuals I_1 and I_2 encoding classifiers $\mathcal{H}_c^1, \mathcal{H}_c^2 \in \mathbf{M}(\mathcal{F}(k), P, N)$, respectively. The *generalization* crossover $GX(I_1, I_2)$ of I_1 and I_2 is the individual encoding either the binary ϕ–generalization $\bigvee_\phi(\mathcal{H}_c^1, \mathcal{H}_c^2)$ or the binary τ-generalization

$\bigvee_\tau(\mathcal{H}_c^1, \mathcal{H}_c^2)$ of $\mathcal{H}_c^1$ and $\mathcal{H}_c^2$. More precisely, using the above agreed notation

$$GX(I_1, I_2) = \begin{cases} \bigvee_\tau(I_1, I_2) & \text{with probability } p = 0.5 \\ \bigvee_\phi(I_1, I_2) & \text{otherwise.} \end{cases}$$

The *specialization* crossover operator $SX(I_1, I_2)$ is defined accordingly (i.e., using $\bigwedge_x$ in place of $\bigvee_x$, with $x \in \{\tau, \phi\}$).

Based on Definition 8.34, the implementation of $\bigvee_\phi(I_1, I_2)$ and $\bigwedge_\phi(I_1, I_2)$ can be achieved by simple bitwise logical operations (OR and AND) on I_1 and I_2 as follows:

- $\bigvee_\phi(I_1, I_2) = I$ s.t. $I^+ = OR(I_1^+, I_2^+)$, $I^- = AND(I_1^-, I_2^-)$, $I^{\succeq_\tau} = I_1^{\succeq_\tau}$.
- $\bigwedge_\phi(I_1, I_2) = I$ s.t. $I^+ = AND(I_1^+, I_2^+)$, $I^- = OR(I_1^-, I_2^-)$, $I^{\succeq_\tau} = I_1^{\succeq_\tau}$.

where $I^+ = OR(I_1^+, I_2^+)$ stands for $\forall i \in [1, k],\ I^+[t_i] = OR(I_1^+[t_i], I_2^+[t_i])$ (AND is defined accordingly).
However, there is a problem with the above implementation. In fact, by performing a "blind" OR or AND, the two individuals exchange 0's and 1's with no regard for the relevance of the features they represent. This may be detrimental, as a surplus of low-quality features may increase the risk of overfitting the training data. To overcome this drawback, we introduce the *probabilistic OR* (pOR) and the *probabilistic AND* (pAND), which are logical operators biased towards high-quality features. They both rely on the notion of *relevance* of a candidate feature.

Definition 8.43 Given the feature space $\langle Pos_c^*(k), Neg_c^*(k)\rangle$, let σ and η be the scoring functions for the positive and the negative candidate terms, respectively (see Definition 8.1). With each $t \in Pos^*(k) \cup Neg^*(k)$ we assign the *relevance* measure $\rho(t)$ as follows:

$$\rho(t) = \frac{f(t)}{Max\{f(t_i)|t_i \in S\}},$$

where $f(t) = \sigma(t)$ and $S = Pos_c^*(k)$ if t is candidate positive feature[3], or $f(t) = \eta(t)$ and $S = Neg^*(t)$ otherwise. Dually, we define the *irrelevance* $\bar{\rho}(t)$ of t as

$$\bar{\rho}(t) = \frac{Min\{f(t_i)|t_i \in S\}}{f(t)}$$

where f and S are as above. Clearly, $0 < \rho(t) \leq 1$ takes on the value 1 for the highest scoring term t, while $0 < \bar{\rho}(t) \leq 1$ takes on the value 1 for the lowest scoring term t (see Figure 8.6).

[3]We assume that $\sigma(t) > 0$ for any $t \in Pos^*(t)$

It can be easily verified that $\approx\bigvee_{\phi}(I_1, I_2)$ is a generalization of I_1 and $\approx\bigwedge_{\phi}(I_1, I_2)$ a specialization of I_1.

Unlike the implementation of the ϕ-subsumption primitives $\bigvee_{\phi}$ and $\bigwedge_{\phi}$, which relies on bit-wise operations performed at the genotype level, the implementation of the τ-subsumption primitives $\bigvee_{\tau}$ and $\bigwedge_{\tau}$ is performed at phenotype level. To see this point, we preliminarily recall that $\bigvee_{\tau}(\mathcal{H}_c^1, \mathcal{H}_c^2) = \langle Pos_1, Neg_1, \sqcup(\mathcal{T}_1, \mathcal{T}_2)\rangle$ (see Definition 8.34). Thus, to implement $\bigvee_{\tau}(I_1, I_2)$, we first extract from the individuals I_1 and I_2 the threshold sets $\mathcal{T}_1$ and $\mathcal{T}_2$ by using the inverse of the encoding function, i.e., $\mathcal{T}_i = enc^{-1}(I_i^{\succeq_\tau})$ $(1 \leq i \leq 2)$. Then, we compute $\sqcup(\mathcal{T}_1, \mathcal{T}_2)$ by using the algorithm of Figure 8.2 and, finally, we apply the encoding function $enc(\sqcup(\mathcal{T}_1, \mathcal{T}_2))$. Therefore, $\bigvee_{\tau}(I_1, I_2)$ is the individual having the same positive and negative components of I_1 and threshold component $enc(\sqcup(\mathcal{T}_1, \mathcal{T}_2))$.

◇ COMPUTATION of $\bigvee_{\tau}(I_1, I_2)$ and $\bigwedge_{\tau}(I_1, I_2)$. Let I_1 and I_2 be two individuals, and let $\mathcal{T}_i = enc^{-1}(I_i^{\succeq_\tau})$ be the threshold set of the classifier encoded by I_i $(1 \leq i \leq 2)$. Then compute

- $\bigvee_{\tau}(I_1, I_2) = I$ such that $I^+ = I_1^+,\ \ I^- = I_1^-,\ \ I^{\succeq_\tau} = enc(\sqcup(\mathcal{T}_1, \mathcal{T}_2))$
- $\bigwedge_{\tau}(I_1, I_2) = I$ such that $I^+ = I_1^+,\ \ I^- = I_1^-,\ \ I^{\succeq_\tau} = enc(\sqcap(\mathcal{T}_1, \mathcal{T}_2))$

where $\sqcup(\mathcal{T}_1, \mathcal{T}_2)$ and $\sqcap(\mathcal{T}_1, \mathcal{T}_2)$ are constructed by the algorithm of Figure 8.2. ∎

The correctness of the above computation directly follows from Definition 8.34.

Generalizing/Specializing (GS) Mutation

The GS mutation (GSM) operators are an implementation of the unary refinement operators defined in Definition 8.32. Therefore, the GSM applied to an individual encoding $\mathcal{H}_c$ returns another individual encoding a neighbor (generalization or specialization) of $\mathcal{H}_c$ in either one of the two hierarchies.

Notation. In the following we will denote, with a small abuse of notation, by $\uparrow^x (I)$ (resp. $\downarrow^x (I)$) the individual encoding the classifiers in $\uparrow^x (\mathcal{H}_c)$ (resp. $\downarrow^x (\mathcal{H}_c)$), where $x \in \{\phi, \tau\}$ and I is the encoding of $\mathcal{H}_c$ (see Definition 8.32).

Definition 8.45 (GSM operators) Let I be an individual encoding $\mathcal{H}_c \in \mathbf{M}(\mathcal{F}(k), P, N)$. The *generalization mutation* $GM(I)$ of I is an individual encoding a direct ancestor of $\mathcal{H}_c$ in $\mathbf{M}(\mathcal{F}(k), P, N)$ either in the τ- or in the ϕ-generalization hierarchy, i.e., either $GM(I) = \uparrow^\tau (I)$ or $GM(I) = \uparrow^\phi (I)$. More precisely,

$$GM(I) = \begin{cases} \uparrow^\phi (I) & \text{with probability } p = 0.5 \\ \uparrow^\tau (I) & \text{otherwise.} \end{cases}$$

The *specialization mutation* $SM(I)$ is defined accordingly.

That is, the GS mutation of I yields an individual encoding, with equal probability, a neighbor of the classifier encoded by I either in the ϕ- or the τ-hierarchy (this is exactly what in our model is a "small" change in a hypothesis).
The computation of the non-deterministic primitives $\uparrow^{\phi}(I)$ and $\downarrow^{\phi}(I)$ is clearly the transposition at genotype level of the computation of the unary refinement operators $\uparrow^{\phi}(\mathcal{H}_c)$ and $\downarrow^{\phi}(\mathcal{H}_c)$ shown in Subsection 8.5.1. Hence, we obtain the binary encoding $\uparrow^{\phi}(I)$ of a classifier $\uparrow^{\phi}(\mathcal{H}_c)$ simply by flipping either one 0 into 1 in I^+ (i.e., add a positive feature to $\mathcal{H}_c$) or a 1 into 0 in I^- (i.e., remove a negative feature from $\mathcal{H}_c$). Dually, we get the binary encoding $\downarrow^{\phi}(I)$ of a classifier $\downarrow^{\phi}(\mathcal{H}_c)$ by flipping a 1 into 0 in I^+ or a 0 into 1 in I^-.
However, like in the case of the GS crossover, we bias the GM mutation towards high-relevance features. To this end, we introduce the notions of *insertion* probability $ip(t)$ and *removal* probability $rp(t)$ of a candidate feature t as follows

$$ip(t) = \frac{\rho(t)}{\sum_{i=1,k}\rho(t_i)},\ \ rp(t) = \frac{\bar{\rho}(t)}{\sum_{i=1,k}\bar{\rho}(t_i)}$$

where $\rho(t)$ and $\bar{\rho}(t)$ are the relevance and the irrelevance measures of t, respectively (see Definition 8.43). Intuitively, the probability $ip(t)$ represents the chance that $I[t]$ is flipped from 0 to 1, that is, the chance that the candidate feature t is selected as a term (either positive or negative) for the classifier encoded by individual I. The meaning of $rp(t)$ is dual. We notice that, since $\rho(t_i) \leq 1$, the condition $\sum_{i=1,k}\rho(t_i) \leq k$ holds (recall that k is the number of both positive and negative features). Therefore, for the highest scoring feature t (for which $\rho(t) = 1$) we have that $ip(t) = 1/\sum_{i=1,k}\rho(t_i) \geq 1/k$, i.e., the maximum insertion probability is not smaller than $1/k$ (defined in [22] as the lower bound of the optimal mutation rate). Dually, the removal probability $rp(t)$ is maximum for the lowest scoring feature t, for which the relation $rp(t) \geq 1/k$ holds as well. We are now ready to provide the computation of $\uparrow^{\phi}(I)$ and $\downarrow^{\phi}(I)$.

◇ `COMPUTATION of` $\uparrow^{\phi}(I)$ `and` $\downarrow^{\phi}(I)$. Let the individual I be given. Compute $\uparrow^{\phi}(I)$ and $\downarrow^{\phi}(I)$ as follows:

a) $\uparrow^{\phi}(I)$: select randomly (with probability 0.5) either one of the options below:

1. probabilistically select a bit $I^+[t] = 0$ according to the insertion probability distribution $ip(t)$; mutate it from 0 to 1, or
2. probabilistically select a bit $I^-[t] = 1$ according to the removal probability distribution $rp(t)$; mutate it from 1 to 0.

b) $\downarrow^{\phi}(I)$: select randomly (with probability 0.5) either one of the options below:

1. probabilistically select a bit $I^+[t] = 1$ according to the removal probability distribution $rp(t)$; mutate it from 1 to 0, or
2. probabilistically select a bit $I^-[t] = 0$ according to the insertion probability distribution $ip(t)$; mutate it from 0 to 1. ∎

Now let us consider the τ-subsumption primitives $\uparrow^\tau$ and $\downarrow^\tau$. Like in the case previously seen of the τ-subsumption primitives $\bigvee_\tau$ and $\bigwedge_\tau$, also the implementation of $\uparrow^\tau$ and $\downarrow^\tau$ is performed at phenotype level. To this end, we first extract from the individual I the threshold set $\mathcal{T}$ by using the inverse of the encoding function, i.e., $\mathcal{T} = enc^{-1}(I^{\succeq_\tau})$. Then, we compute $\uparrow \mathcal{T}$ by using the algorithm of Figure 8.4 and, finally, we apply the encoding function $enc(\uparrow \mathcal{T})$. Therefore, $\uparrow \mathcal{T}$ is the individual having the same positive and negative components of I and threshold component $enc(\uparrow \mathcal{T})$.

◇ `COMPUTATION of` $\uparrow^\tau (I)$ `and` $\downarrow^\tau (I)$. Let the individual I be given, and let $\mathcal{T} = enc^{-1}(I^{\succeq_\tau})$ be the threshold set encoded by $I^{\succeq_\tau}$. Now, $\uparrow^\tau (I)$ is implemented simply by replacing the encoding $I^{\succeq_\tau}$ of $\mathcal{T}$ by the encoding $enc(\uparrow \mathcal{T})$ of any direct ancestor $\uparrow \mathcal{T}$ computed by the algorithm of Figure 8.4. $\downarrow^\tau (I)$ is implemented accordingly. That is:

- $\uparrow^\tau (I) = I'$, where $I'^+ = I^+$, $I'^- = I^-$, and $I'^{\succeq_\tau} = enc(\uparrow \mathcal{T})$
- $\downarrow^\tau (I) = I'$, where $I'^+ = I^+$, $I'^- = I^-$, and $I'^{\succeq_\tau} = enc(\downarrow \mathcal{T})$. ∎

8.7.5 The Genetic Algorithm

First, the dimensionality range $[k_{min}, k_{max}]$ is computed from the input vocabulary by applying Definition 8.39. Then, given the (user-defined) input values P_{max} and N_{max}, for each randomly generated triple (k, P, N), with $k \in [k_{min}, k_{max}]$, $0 \leq P \leq P_{max}$ and $0 < N \leq N_{max}$, a random number (> 1) of individuals of length $2k + Min(P + 1, N)(\lceil log(N(P + 1)) \rceil + 1)$ is created. Each of such individuals encodes a classifier in the hypothesis space $\mathbf{M}(\mathcal{F}(k), P, N)$. The set of individuals created for the same triple (k, P, N) form a subpopulation $S(k, P, N)$. Each individual $I \in S(k, P, N)$ is initialized as follows: the k bits in I^+ and I^- are set to 1 with probability 0.5, while $I^{\succeq_\tau}$ is randomly set to a (minimal) threshold set $\{(p_1, n_1), \cdots, (p_r, n_r)\}$ such that $0 \leq p_i \leq P$ and $0 < n_i \leq N$, for each $i = 1, r$ (see Definition 8.2). Afterwards, evolution takes place by iterating elitism, selection, crossover and mutation, until a pre-defined number of generations is created. Finally, the phenotype of the best generated chromosome is returned.

Next we give some details about selection, crossover and mutation.

Selection. We want to be able to preserve sub-populations under the pressure of selection, in order to guarantee a certain degree of population diversity (niching methods are often used for this purpose [17, 60]). At the same time, we want to avoid premature convergence within sub-populations that consists of a small number of individuals. At these aims, we maintain a set of

mating pools, each being the union of all subpopulations with the same threshold bounds P and N, i.e., $M(P,N) = \cup_i S(k_i, P, N)$. So, individuals in $M(P,N)$ may have different length, while belonging to isomorphic τ-subsumption lattices. In particular, the lengths of two individuals in $M(P,N)$ may differ only as far as the feature components are concerned, the threshold components being of equal length (as they have the same threshold bounds P and N - see Sub-section 8.7.2). Now, selection is performed as follows: a mating pool is randomly selected, and tournament selection is then applied over its individuals.

Crossover. We are given individuals $I_1 \in S(k_1, P, N)$ and $I_2 \in S(k_2, P, N)$, thus belonging to the same mating pool $M(P,N)$. The GAMoN crossover of I_1, I_2 combines a slightly modified version of the uniform crossover (called MUX) with the GS crossover operators defined in the previous sections. A sketch of the proposed method is shown in Figure 8.7. It basically relies on two steps:

Step 1: decide probabilistically whether or not $MUX(I_1, I_2)$ takes place (line 18). This decision is made positively with (user-defined) probability p_x. MUX is an adaptation of the uniform crossover UX to deal with (i) the different lengths of the feature components of two mating individuals, and (ii) the presence of threshold sets. Informally, $MUX(I_1, I_2)$ can be regarded as $UX(I_1, I_2)$ where (1) only the first $min(k_1, k_2)$ bits of the positive and negative components of I_1 and I_2 are probabilistically exchanged (lines 2-4), and (2) $I_1^{\succeq}$ and $I_2^{\succeq}$ are swapped as if they were single bits (line 5). Note that the offspring J_1 and J_2 are such that $J_1 \in S(k_1, P, N)$ and $J_2 \in S(k_2, P, N)$ (i.e., they belong to the same subpopulations of their parents).

Step 2: If the decision for MUX has not been made positively in Step 1, then perform the GS crossover by invoking function $GSX(I_1, I_2)$ (line 20). This is executed with a probability equal to the F-measure of the classifier $\mathcal{H}_c$ encoded by I_1 (line 13). This way, we give fitter individuals a higher chance to generalize or specialize so as to allow them for further refinement - see discussion in Sub-section 8.7.7. Whether a generalization or a specialization of I_1 is to be performed, depends on whether $\mathcal{H}_c$ is too specific or too general (line 14). However, to carry out $GSX(I_1, I_2)$, individual I_2 has preliminarily to be "promoted" citizen of the subpopulation $S(k_1, P, N)$ of I_1 as, by Definition 8.42, the GS crossover can be applied only to members of the same sub-population. To this end, I_2 is made of the same length of I_1 by invoking function *promote* (line 15). The following two cases may arise (recall that the threshold set components of I_1 and I_2 are of equal length):

- $k_1 \leq k_2$. Only the first k_1 bits of I_2^+ and I_2^- are picked up (lines 7-8).
- $k_1 > k_2$. Both I_2^+ and I_2^- are extended with further $n = k_1 - k_2$ bits. In particular, I_2^+ is padded with n 1's (resp. 0's) and I_2^- with n 0's (resp. 1's) if a generalization (resp. specialization) is to be performed (lines 9-12).

At this point, either $GX(I_1, I_2)$ or $SX(I_1, I_2)$ is computed according to Definition 8.42 (lines 16-17). Once $GSX(I_1, I_2)$ has been carried out, $GSX(I_2, I_1)$ is performed likewise (line 20). Again, the offspring J_1 and J_2 belong to sub-populations $S(k_1, P, N)$ and $S(k_2, P, N)$, respectively.

Mutation. Mutation is performed by using a similar framework. This is a combination of a modified version of standard mutation (denoted MSM), which takes into account threshold sets, with the GS mutation (GSM) operators previously defined. In particular, MSM(I) works as follows: first, it randomly decides (with probability 0.5) whether to operate over the feature or the threshold component. In the former case, MSM(I) randomly flips the bits of I^+ and I^- with probability $1/(2k)$. In the latter case, MSM(I) replaces $I^{\succeq}$ by the encoding of a randomly chosen neighbor of the threshold set encoded by $I^{\succeq}$ (contrary to GSM which selectively chooses either a direct ancestor or a direct descendant depending on whether generalization or specialization is to be performed, respectively). Note that, in all cases, MSM(I) causes small changes of position in the subsumption lattices. Now, GAMoN mutation works as follows (for each offspring):

- *Step* 1: decide probabilistically whether or not GSM(I) takes place. This decision is made positively with probability $F\text{-}measure(\mathcal{H}_c)$, $\mathcal{H}_c$ being the phenotype of I.
- *Step* 2: If the decision for GSM(I) has not been made positively in step 1, execute MSM(I).

8.7.6 GAMoN time complexity

It is immediate to recognize that the cost of the task-specific reproduction operators is $O(k)$, while the cost of the fitness computation is $O(km)$, where k is the size of the feature space and m the number of examples in the training set (in fact, the evaluation of the fitness of an individual requires the evaluation of the number of candidate (both positive and negative) features occurring in each document of the training set). Now, since the number of different features (words) occurring in the training set is asymptotically independent of m (as the lexicon is finite), irrespective of feature selection, k is (asymptotically) independent of m. Thus, technically, we have that $O(km) = O(m)$, that is, the asymptotic behavior of GAMoN is linear in the size of the training set. Quite obviously, for relatively small values of m (like those that characterize real-life data sets), the practical complexity is $O(km)$.

8.7.7 Remarks on the proposed GA

Individual Encoding. There are two basic approaches, according to whether a chromosome of the population is used to represent *a single rule* or a *rule set* [65]. Within the former approach (i.e., "chromosome = one rule") there are rule induction GAs like XCS [132], SIA [129], COGIN [61]. In the second approach (i.e., the "chromosome = set of rules"), called Pittsburgh approach, a rule is used to code an entire classifier. GAssist [19], OlexGA [101] and BioHEL [18, 48] fall in this category.

From one side, the "chromosome = one rule" approach makes the individual encoding simpler, but the fitness of a genotype may not be a meaningful indicator of the quality of the phenotype [50].

Further, under the competitive style of GA, there may be a conflict between individual and collective interests of the rules forming a classifier [65]. On the other side, the "chromosome = set of rules" approach requires a more sophisticated encoding of individuals, but the fitness provides a more reliable indicator [50]. Moreover, no conflict of interests can happen in this case, as competition occurs among classifiers (and not single rules).
In our approach, an individual encodes a candidate classifier - so it falls in the class of Pittsburgh methods. Despite this, individual encoding is very simple and compact - $2k$ bits for the encoding of Pos and Neg (a few tens of bits altogether) and a handful of bits to encode the threshold set. Thus, GAMoN combines the advantages of both the above mentioned approaches, i.e., the individual simplicity and compactness of the "chromosome = one rule" approach, along with the effectiveness of both the reproductive competition and the fitness function of the "chromosome = set of rules" approach.

Search Strategy. It is well known that standard reproduction operators are rather disruptive, in the sense that the offspring may be very different from the parents. On one hand, this has the advantage of making unlikely the GA getting stuck into local optima but, on the other hand, the high degree of unpredictability in the generation of new candidate classifiers may make the GA converge very slowly. In contrast, GS operators move hypotheses from one position to another in either one of the two hierarchies in a controlled way, depending on the "state" of the current hypothesis. Such a search bias, however, forces a search strategy which may quickly converge to local optima.
As we have seen, GAMoN combines the space search of a standard GA with that based on GS operators. The rationale behind this choice is that of exploiting the latter to perform a selective search, and to compensate the selectiveness of this search by introducing a certain degree of diversity through the standard operators. In particular, GAMoN runs the GS operators (both crossover and mutation) with increasing probability, this being defined as the F-measure achieved by an individual I over the training set. This way, as the generations pass and the algorithm more and more approaches the optimal solution, a more controlled search of the space is performed.

$GAMoN\ Xover(I_1, I_2)$
Input: individuals $I_1 \in S(k_1, P, N)$ and $I_2 \in S(k_2, P, N)$; MUX probability p_x;
Output: offspring $J_1 \in S(k_1, P, N)$ and $J_2 \in S(k_2, P, N)$
function $MUX(I_1, I_2)$
1. $J_1 = I_1$, $J_2 = I_2$;
2. **for** $i = 1\ to\ min(k_1, k_2)$ **do**
3. Swap($J_1^+[i], J_2^+[i]$) with probability 0.5;
4. Swap($J_1^-[i], J_2^-[i]$) with probability 0.5;
5. Swap($J_1^{\succeq_\tau}, J_2^{\succeq_\tau}$) with probability 0.5;
return J_1, J_2.
function $promote(I_2, generalize)$
6. $J^{\succeq_\tau} = I_2^{\succeq_\tau}$; /* copy the threshold component of I_2 into J */
7. **for** $i = 1, min(k_1, k_2)$ **do** /* copy the first $min(k_1, k_2)$ positive and negative features from I_2 to J */
8. $J^+[i] = I_2^+[i],\ J^-[i] = I_2^-[i]$;
9. **for** $i = k_2 + 1, k_1$ **do** /* when $k_1 > k_2$ add further $k_1 - k_2$ bits to both J^+ and J^- so that their length becomes k_1 */
10. **if** $generalize$ **then**
11. $J^+[i] = 0,\ J^-[i] = 1$ /* pad J^+, J^- with $k_1 - k_2$ 0's and 1's, resp., */
12. **else** $J^+[i] = 1,\ J^-[i] = 0$; /* pad J^+, J^- with $k_1 - k_2$ 1's and 0's, resp. */
return J.
function $GSX(I_1, I_2)$
13. with probability equal to $Fmeasure(\mathcal{H}_c)$ do /* $\mathcal{H}_c$ is the classifier encoded by I_1 */
14. $generalize = (precision(\mathcal{H}_c) > recall(\mathcal{H}_c))$;
15. $\widehat{I}_2 = promote(I_2, generalize)$; /* I_2 becomes a citizen of $S(k_1, P, N)$ */
16. **if** $generalize$ **then** $J = GX(I_1, \widehat{I}_2)$
17. **else** $J = SX(I_1, \widehat{I}_2)$ /* GX or SX are performed according to Definition 8.42 */
return j.
begin
18. with probability p_x set $< J_1, J_2 >= MUX(I_1, I_2)$;
19. **if** MUX has not been performed **then**
20. $J_1 = GSX(I_1, I_2)$; $J_2 = GSX(I_2, I_1)$
return J_1, J_2.

Figure 8.7: Pseudocode for the GAMoN Xover

Chapter 9

Empirical Investigation Framework

9.1 Machine learning algorithms

To evaluate the GAMoN approach proposed in this paper, we focused on comparisons with other rule learning algorithms. To this end, we selected two rule induction GAs, namely, BioHEL and OlexGA, and two non-evolutionary algorithms, namely, C4.5 and Ripper. Further, we included in our study Platt's Sequential Minimal Optimization (SMO) method for linear SVM training [76], as it is reported to be one of the best methods for text categorization.
Our interest for OlexGA was that of assessing to what extent GAMoN is an effective extension (see Section 8.1). BioHEL was chosen as it is one of best performing GA-based methods. We are not aware of experimental results of BioHEL on textual data sets. Finally, C4.5 and Ripper were selected as they are standard decision tree/rule learners widely used for text classification.
All the selected learning algorithms are implemented in Java, but not all are available on the same platform. In particular, GAMoN runs only on the Weka platform, while BioHEL is available only on the KEEL platform (KEEL - Knowledge Extraction based on Evolutionary Learning - is a suite of machine learning software tools - [11]). C4.5, Ripper, SMO and OlexGA run on both platforms. For the purpose of our work, we used Weka (version 3.5.8) for all algorithms, but BioHEL.

9.2 Benchmark Corpora

We carried out our empirical work on 13 real-world data sets whose properties are summarized in Table 9.1. As we can see, they span over a wide range of sizes, from a minimum of around 900 (Oh15) to a maximum of nearly 204,000 (market) documents. The rarest category has 51 documents (Oh0), while the most frequent has 85,440 documents (Market). Most of these datasets have been widely used in large scale text classification tasks, and are publicly available.

Market is a data set of 203,926 documents extracted from Reuters Corpus Volume I (RCV1),

Table 9.1: Data set description

Name	Source	Original Format	#Doc	#Feat	#Cat	Cat Size Min	Cat Size Max
Oh15	Ohsumed-233445	arff	913	3,100	10	53	157
Oh5	Ohsumed-233445	arff	918	3,012	10	59	149
Oh0	Ohsumed-233445	arff	1,003	3,182	10	51	194
Oh10	Ohsumed-233445	arff	1,050	3,238	10	52	165
BlogsGender	Blog author gender	text	3,232	15,026	2	1,548	1,684
Ohscale	Ohsumed-233445	arff	11,162	11,465	10	709	1,621
R10	Reuters-21578	text	12,897	21,363	10	237	3,964
20NG	20 newsgroups	csv	18,846	59,903	20	628	999
Ohsumed	Ohsumed-233445	text	34,389	34,359	23	427	9,611
Cade12	Gerindo Proj.	csv	40,983	69,470	12	625	8,473
SRAA	UseNet	text	73,218	63,966	4	4,796	41,351
ODP-S22	ODP	text	107,262	25,068	22	88	28,286
Market	Rcv1	text	203,926	68,604	4	26,036	85,440

(Lewis et al., 2004). *R10* is the standard subset of the Reuters-21578 Distribution 1.0 which consists of 12,897 documents and uses the 10 most frequent Topics categories [39]. *Ohsumed* is from the Ohsumed-233445 collection subset of MEDLINE database [67] and is made of all 34,389 cardiovascular diseases abstracts out of 50,216 medical abstracts contained in the year 1991. The classification scheme consists of the 23 cardiovascular diseases MeSH categories. *Ohscale*, *Oh0*, *Oh5*, *Oh10* and *Oh15* are other subsets of Ohsumed-233445 [68]. Data sets SRAA and 20NG (20-newsgroups) are articles from newsgroups. In particular, *20NG* is a collection of 18,846 newsgroup documents organized into 20 different categories. We used the version sorted by date, which does not include newsgroup-identifying headers. The SRAA [1] data set contains 73,218 articles from four discussion groups on simulated auto racing, simulated aviation, real autos, and real aviation. *BlogsGender* is a binary data set of 3,232 blogs used for author gender classification [2]. *Cade12* is a subset of the CADE Web Directory consisting of 40,983 web pages classified across 12 categories [3]. *ODPS-22* is a subset of ODP (Open Directory Project) [4] whose documents are stored as RDF files. For our experimentation, we used the subset of 107,262 documents classified under the categories of the *Top/Science* subtree, which has 25 first-level categories. We first collapsed each of the 25 subtrees into the respective root, thus obtaining a flat structure made of 25 categories. Then, we grouped together into one category "Misc" the 4 smallest categories, namely, Search Engines (7 documents), Charts-and-forums (16 documents), Directories (27 documents) and Events (38 documents), thus getting a set of 22 categories. From each document (web page), we extracted the title and the description (thus, discarding the URL).

9.3 Experimental Setup

We preliminarily pre-processed all data sets downloaded in textual format, by performing tokenization (word unigrams) and stopword removal. We used the bag-of-words representation with binary word weighting. Each feature was represented as a numerical attribute.
Experiments were performed in a binary classification setting. To this end, we binarized all data sets by performing multi-class to two-class conversion. This way, the m-class learning problem is decomposed into m independent two-class sub-problems, one for each class, with the i-th classifier separating class i from all the remaining ones.
Finally, for each category, feature scoring by CHI square [46] was performed (on the training set).
Following are two major issues arose during the design of experiments.

1. Dimensionality of the feature space.

   (a) Unlike the other systems, GAMoN automatically detects the appropriate dimensionality of the feature space. That is, no manual feature selection is preliminarily needed. As we will see later on this section, the feature spaces selected by GAMoN usually consist of a few tens of features.

   (b) Previous works show that systems like Ripper, C4.5 and SVM require relatively large vocabularies (usually a few thousands of features) to learn good prediction functions.

   (c) The efficiency of OlexGA, like that of most evolutionary methods, strongly depends on the feature space dimensionality, as many features imply long individuals and, thus, low efficiency.

   (d) BioHel represents an exception in the evolutionary landscape, as it was designed to efficiently deal with high dimensional data sets. However, the memory space limitations of the KEEL platform severely limits the number of attributes that can actually be used in case of large data sets.

   The above observations demonstrate the difficulty of applying a single feature selection policy to all systems. In fact, it would be unfair using for the non-evolutionary methods the feature dimensionalities detected by GAMoN (too small for their characteristics - see points (a) and (b)). On the other hand, running OlexGA and BioHEL over the same number of features used for the non-evolutionary methods would practically be unfeasible - see points (c) and (d).

2. Time efficiency. As we have seen, our empirical study involves very large data sets (e.g., ODP-S22 and Market), on which most of the experimented systems perform quite inefficiently. In particular, Ripper and C4.5 showed to be extremely slow on such data sets, especially when we tried to use large vocabularies (for an instance, on vocabularies of 10.000

features, we had to stop C4.5 as it was overly inefficient). This prevented us from performing optimization over more vocabularies.

Given the above premises, the following experimental design choices were finally taken:

1. GAMoN was run with 500 individuals, 200 generations, elitism rate 0.2, MUX probability 0.6 (see Section 8.7.5). The maximum threshold bounds were $P_{max} = N_{max} = 4$.
2. OlexGA and BioHEL were executed over the same vocabularies made of 100 features. The former was run with the default parameters shown at `http://www.mat.unical.it/OlexGA`), while the latter with those provided by KEEL.
3. The remaining (non-evolutionary) systems were all executed over vocabularies made of 2000 terms, with the default settings provided by Weka. The SMO normalization option was turned off to improve the training time.

Due to efficiency reasons, we performed 5-fold cross validation (80% training, 20% test) only on small data sets (from Oh15 up to R10), while holdout (70% training, 30% test) was applied on the remaining data sets.

9.4 Predictive performance measure and Statistical Tests

Performance was measured, as is common in text classification, by the arithmetic mean of Precision and Recall - denoted PRavg (an approximation of the Precision/ Recall Break-Even Point). To obtain global estimates over more categories, the standard definitions of micro-averaged Precision and Recall were used, notably:

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

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

where TP_c is the set of documents correctly assigned by the classifier to category c, FP_c is the set of documents incorrectly assigned by the classifier to category c, and FN_c is the set of documents incorrectly not assigned by the classifier to category c. We note that micro-averaging gives equal weight to every document (it is called a document-pivoted measure) and is largely dependent on the most common categories.

Each run of the evolutionary algorithms was repeated 3 times, and the average PRavg was taken.

In order to make comparisons statistically significant, we performed the Iman-Davenport test, with the Holm's *post-hoc* test, recommended for comparison of more classifiers on multiple data sets in [40].

Chapter 10

Experimental Results

10.1 A glimpse to M-of-$N^{\{\neg,\vee\}}$ hypotheses

The experimental results show that GAMoN has a bias towards learning compact and readable hypotheses. The following are examples of classifiers induced for categories "corn", "wheat" and "grain" from R10:

$$\mathcal{H}_{wheat} = \langle\{wheat\}, \{deficit, investment, net, treasury, york\}, \{(1,1)\}\rangle.$$

$$\mathcal{H}_{corn} = \langle\{corn, maize\}, \{london, money, quarter\}, \{(1,1)\}\rangle$$

$$\mathcal{H}_{grain} = \langle\{barley, cereals, corn, grain, maize, rice, sorghum, wheat\},$$
$$\{acquisition, bank, earning, pay, profit, tax, york\}, \{(1,1),(2,2)\}\rangle$$

As we can see, the former two classifiers are atoms, while the latter is a 2-order classifier (for a description see Section 8.3). It must be emphasized the high semantic correlation between the positive features and the respective categories.

10.2 Automatic selection of the feature space dimensionality

Table 10.1 shows, for the categories from R10, the values of k_{min}, k_{max} and k_{opt} given by one execution of GAMoN, where k_{min} and k_{max} define the dimensionality range of the feature space, and k_{opt} ($k_{min} \leq k_{opt} \leq k_{max}$) is the size of the feature space on which the "optimal" classifier has been found. As it can be seen, the learning of all categories generally relies on small sets of candidate features. As an example, for "corn" we have $k_{min} = 19$ and $k_{max} = 43$ (and PRavg equal to 90.20), meaning that the positive terms for the "best" classifier are to be found among the first k higher scoring features, with $19 \leq k \leq 43$. This is clearly indicative of an aggressive feature selection. To see why, let us have a look at Figure 10.1 - left side, where the distribution of features

Table 10.1: Feature space dimensionality range $[k_{min}, k_{max}]$, size of the feature space on which the "optimal" classifier has been found k_{opt} and F-measure for categories of R10

	acq	corn	crud	earn	grain	int	mon	ship	trad	wheat
k_{min}	105	16	58	37	42	59	86	53	78	18
k_{max}	172	43	106	60	77	102	148	97	141	41
k_{opt}	132	21	100	41	55	68	99	61	86	21
PRavg	86.66	90.20	87.50	95.18	92.09	56.05	68.12	80.41	69.15	88.66

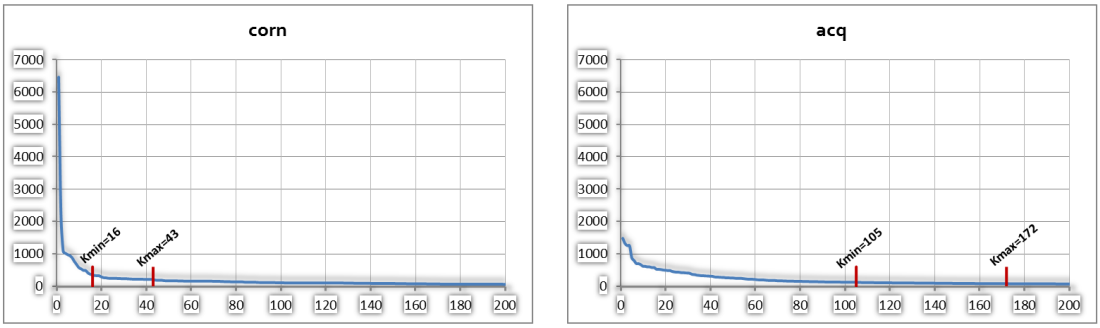

Figure 10.1: Distribution of features by CHI square for two categories from R10 - "corn" (left side) and "acq" (right side). Only first 200 features are shown.

by CHI square is reported. As we can see, "corn" has a few features scoring very high, while the remaining ones rapidly approach near-zero values. The sharply declining shape of this graph is indicative of an "easy" category, i.e., a category for which a high performance can be achieved with only a few discriminative words.

In contrast, "acq" is a more "difficult" category. As we can see from Figure 10.1- right side, it has lower initial CHI square values, and the graph has a smooth (decreasing) trend. That is, no features with highly discriminative power there exist. As a consequence, the dimensionality range is shifted rightwards on the x-axis ($k_{min} = 105$ and $k_{max} = 172$), this being indicative of a less aggressive reduction of the feature space.

10.3 Decision Boundaries

Figure 10.2 shows the decision boundary $DB_{\mathcal{H}_{earn}}$ of the 3-order classifier $\mathcal{H}_{earn} = \langle Pos, Neg, \{(1,1),(2,2),(3,3)\}\rangle$ for category "earn" from R10. Here, Pos is made of 14 positive features and Neg consists of 16 negative features. As we can see, $DB_{\mathcal{H}_{earn}}$ is a three-step polyline. From

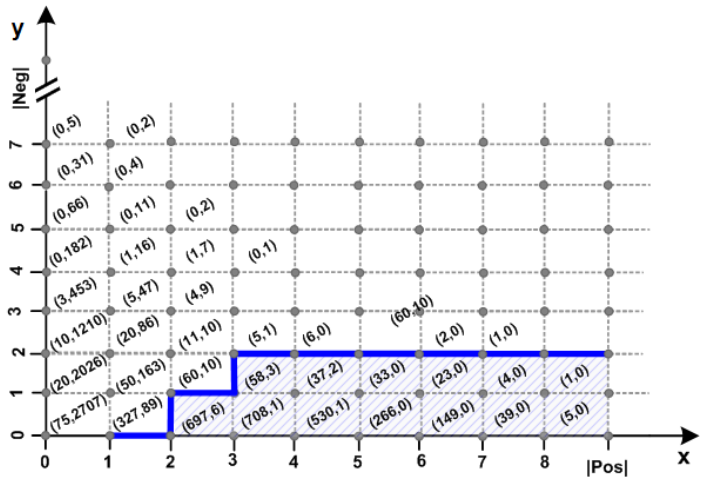

Figure 10.2: Decision boundary of the classifier $\langle Pos,\ Neg, \{(1,1),(2,2),(3,3)\}\rangle$ for category "earn" from R10. Each label $(\pi(x,y), \nu(x,y))$ represents the number $\pi(x,y)$ of positive examples and the number $\nu(x,y)$ of negative ones with x positive features and y negative ones. Labels $(0,0)$ are omitted from the figure.

the data reported in Figure 10.2, it results that the subset of documents classified by $\mathcal{H}_{earn}$ consists of the 2954 positive examples and the 113 negative ones lying in the classification region $\mathrm{R}_{\mathcal{H}_{earn}}$ delimited by $DB_{\mathcal{H}_{earn}}$. The generalization error and the PRavg value of $\mathcal{H}_{earn}$ are

$$err = \frac{b+c}{a+b+c+d} = 0.031; \quad PRavg = \frac{2a^2+a+ac}{2(a+b)(a+c)} = 0.94$$

where a, b, c and d are computed as follows (see Subsection 8.4.5):

$$a = \sum_{(x,y)\in \mathrm{R}_{\mathcal{H}_{hearn}}} \pi(x,y) = 2954, \quad b = \sum_{(x,y)\in \mathrm{R}_{\mathcal{H}_{hearn}}} \nu(x,y) = 113,$$

$$c = \sum_{(x,y)\notin \mathrm{R}_{\mathcal{H}_{hearn}}} \pi(x,y) = 205, \quad d = \sum_{(x,y)\notin \mathrm{R}_{\mathcal{H}_{hearn}}} \nu(x,y) = 6988$$

10.4 Effect of GS operators

To see the effect of the GS reproduction operators defined in this paper, we compared the accuracy results of GAMoN over the data sets previously seen with those obtained by running a version of GAMoN where the GS operators (GS Xover and GS mutation) were disabled. The experimental

results, reported in Table 10.2, show that the GS operators improve the accuracy on every single data set, even though on some they induce only trivial improvements (e.g., Ohsu and BG), while on other the gain is remarkable (e.g., oh15, oh10, R10, cade and Mkt). On average, the enhancement over all data sets is of around 1.5 points. As discussed earlier in this paper, the aim of GS operators is that of further refining fitter individuals by exploiting the structure of the hypothesis space. This explains why often significant improvements are obtained.

Table 10.2: Micro-averaged PRavg values on each data set obtained by GAMoN and GAMoN* (a version of GAMoN with no GS reproduction operators). Legend - BG: BlogsGender, OhS: OhScale, Ohsu: Ohsumed, Mkt: Market

	oh15	oh5	oh0	oh10	BG	OhS	R10	20NG	Ohsu	cade	SRAA	ODP	Mkt
GAMoN	80.46	84.85	84.27	78.62	68.82	75.03	86.50	75.72	67.25	48.42	85.25	71.14	92.42
GAMoN*	77.59	82.55	82.90	75.54	68.33	74.28	84.56	75.21	67.09	46.43	84.27	70.20	90.59

10.5 Comparison with other systems

Table 10.3 shows, for each algorithm and data set, the micro-averaged PRavg. The average values over all data sets along with the average ranking of each algorithm, are also included (bottom of the table). The best results are stressed in bold-face. The ranking is obtained by assigning a position to each algorithm depending on its performance on each data set. The algorithm showing the best accuracy on a given data set is assigned rank 1.
As we can see, SMO is the best performer (PRavg=78.00), followed by GAMoN (PRavg=76.83). The two algorithms, however, show the same average rank (2.08). We note that GAMoN outperforms all the other rule induction methods. In particular, it behaves uniformly better than OlexGA on all individual data sets, and compares favorably with the other rule learners on most of the data sets.

In order to establish whether the above differences in performance are statistically significant, the Iman-Davenport's test is applied. This is a non-parametric statistical test recommended in [40] for comparing two or more classifiers on multiple data sets. In brief, with 13 data sets, 6 algorithms and confidence $\alpha = 0.05$, the Iman-Davenport statistics is 9.54, greater than the critical value $CV = 2.37$. Thus, the null hypothesis (which states that all the algorithms are equivalent) is rejected. Hence, we apply the Holm's *post-hoc* test [40], with GAMoN as control algorithm, for controlling the family-wise error in multiple hypothesis testing. The results of this test are summarized in Table 10.4. Based on them, we can reject the null hypothesis of equivalence only for BioHEL and OlexGA (as $p\text{-}value < \alpha/i$ holds). That is, with confidence 95%, we can state that

Table 10.3: Micro-averaged PRavg results obtained by 5-fold cross-validation on Oh0, Oh5, Oh10, Oh15, BlogsGender, Ohscale and R10 (80/20 split) and by holdout on the remaining data sets (70/30 split)

Data set	**Ripper**	**SMO**	**C4.5**	**OlexGA**	**BioHEL**	**GAMoN**
OH15	79.55	79.95	76.75	74.33	66.03	**80.46**
OH5	83.74	84.29	82.30	80.76	76.72	**84.85**
OH0	84.37	**84.80**	79.24	81.29	73.20	84.27
OH10	**78.82**	74.70	74.78	74.46	67.39	78.62
BlogsGender	60.96	60.95	58.33	66.75	62.82	**68.82**
Ohscale	72.96	69.52	70.77	74.36	68.26	**75.03**
R10	85.21	**88.94**	84.67	84.07	83.59	86.50
20NG	72.66	**83.64**	74.86	72.97	70.65	75.72
Ohsumed	60.35	66.94	63.25	65.58	63.79	**67.25**
cade	44.31	**54.06**	48.10	44.10	42.96	48.42
SRAA	81.04	**90.06**	86.85	79.60	81.34	85.25
ODP	66.73	**80.65**	74.76	69.46	71.21	71.14
Market	94.63	**95.56**	95.37	88.95	74.75	92.42
avg microPRavg	74.26	78.00	74.62	73.59	69,44	76.83
avg rank	3.96	**2.08**	3.62	4.31	5.23	**2.08**

GAMoN performs better than such algorithms, while it is statistically equivalent to SMO, Ripper and C4.5.

Table 10.4: Holm's test with GAMoN as control algorithm. The null hypothesis is rejected when $p\text{-}value < \alpha/i$

i	Method	$z = \frac{(R_0 - R_i)}{SE}$	$p-$value	α/i
5	BioHEL	-4.2980	0.0005	0.01
4	OlexGA	-3.0400	0.0024	0.0125
3	Ripper	-2.2014	0.0278	0.0167
2	C4.5	-2.0966	0.0366	0.025
1	SMO	0.0000	1.0000	0.05

Table 10.5: Avg size of the rule-based classifiers on R10

Algorithm	Avg size of classifiers
GAMoN	#Pos = 20, #Neg=10, order=1.8
Ripper	#Rules= 16
C4.5	#Rules= 78
BioHEL	#Rules= 14, #literals/rule = 19
OlexGA	#Pos = 16, #Neg=15

10.6 Size of the classifiers

Apart from SMO, the other classifiers yield models as sets of rules. Although we do not have a unique formal definition of size of a classifier, being either the number of rules, number of features, etc., in Table 10.5 we provide some statistical data (averaged over the five folds) giving an insight into the quantitative characteristics of the classifiers induced on R10. As we can see, Ripper, C4.5 and BioHEL induce classifiers consisting on average of 16 rules, 78 rules and 14 rules, respectively, each BioHEL rule having 19 literals on average. In turn, GAMoN induces classifiers of 20 positive features and 10 negative ones on avearge, against the 16 positive features and 15 negative ones of OlexGA classifiers. Going beyond the results given in the table, nearly 44% of the classifiers induced by GAMoN are atoms, 38% are of order 2, 18% of order 3 and 2% of order 4 (note that with $P = N = 4$, the maximum order of a classifier is $Min(P+1, N) = 4$ - see Proposition 8.31).

10.7 Time Efficiency

The experiments previously described were performed on an Intel Xeon 2.33 GHz machine with 4 Gb RAM.

The learning times needed to achieve the accuracy results previously seen are reported for each method in Table 10.6 - first row. As we can see, OlexGa (46 hours) is the best performers, followed by SMO (71), GAMoN (156), BioHEL (185), Ripper (445) and C4.5 (488) (recall that each run of GAMoN, OlexGA and BioHEL was repeated 3 times). Table 10.6 also reports the average learning times per category. Again, OlexGA is the fastest algorithm (0.04 h/category), followed by GAMoN (0.12), BioHEL (0.16) and SMO (0.18). Ripper and C4.5 are ten times slower than GAMoN (1.13 and 1.23, respectively).

To see the effect of the training set size over learning times, in Figure 10.3 we plotted the average learning times per category over each data set (data sets are ordered by increasing size). The graph provides an empirical picture of the progression of learning times with the number of training documents. As we can see, GAMoN asymptotically behaves similarly to OlexGA, BioHEL and

Table 10.6: Learning times expressed in hours. Each run of GAMoN, OlexGA and BioHEL was repeated 3 times

	Ripper	**C4.5**	**SMO**	**OlexGA**	**BioHEL**	**Gamon**
Overall learning time (h)	445	488	71	46	185	156
# runs	395	395	395	1185	1185	1185
Avg learning time per category (h)	1.13	1.23	0.18	0.04	0.16	0.12

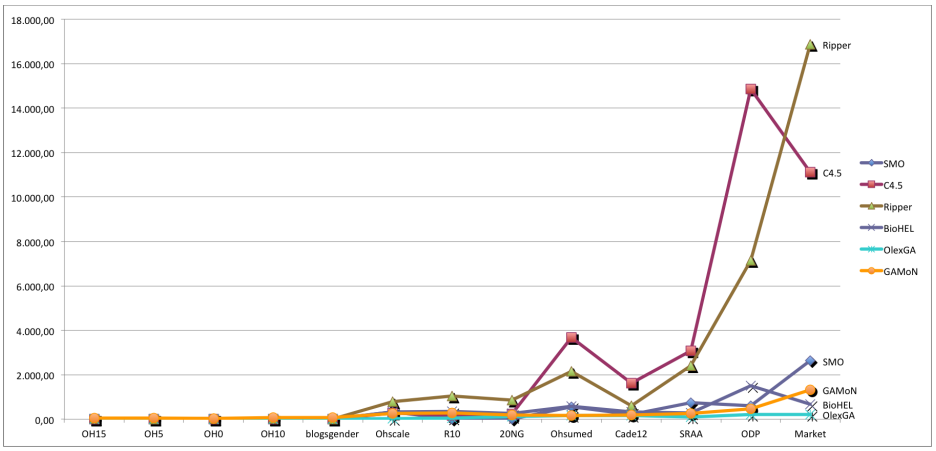

Figure 10.3: Comparison of the average learning time per category over each data set (data sets are ordered by increasing size)

SMO, while its has a significantly smoother trend than both Ripper and C4.5. That is, GAMoN scales better than the two non-evolutionary rule induction methods.

Chapter 11

Discussion and Related Work

The experimental study described in the previous sections shows that GAMoN induces classifiers that are both accurate and compact. Interestingly, these properties have consistently been observed over all 13 data sets, on which GAMoN showed a uniform behavior. Given the very different application domains the corpora refer to, this is a clear proof of robustness. Further, GAMoN showed to perform efficiently on large data sets.

M-of-$N^{\{\neg,\vee\}}$ Representation. As discussed in Sub-section 8.4.6, the "family resemblance" metaphor provides us with a qualitative understanding on the basic reason why the M-of-N paradigm is well suited for the purpose of text categorization. M-of-$N^{\{\neg,\vee\}}$ extends M-of-N through negation and disjunction, two constructs that enables to express hypotheses capable of best fitting the true structure of the data (a discussion on the expressivity of M-of-$N^{\{\neg,\vee\}}$ has been reported in Sub-section 8.4.6). Unlike most of the existing classifiers which focus on features that positively discriminate a class, in our approach negation is used as a "first class citizen" allowing us to explicitly model the interactions between positive and negative features within a given example. In turn, disjunction enables to "modulate" such interactions, by capturing the positive correlation (that simple atoms would miss) existing between positive and negative features (see Sub-section 8.4.5). Negation takes precision under control, while disjunction improves recall.

One advantage of the proposed language over a DNF-type representation is conciseness. Indeed, although a M-of-$N^{\{\neg,\vee\}}$ hypothesis can be represented in terms of disjunctions of conjunctions (see Section 8.4.2), a DNF-type representation of a M-of-$N^{\{\neg,\vee\}}$ concept would be prohibitively long (the number of disjuncts is exponential in the size of Pos and Neg). We believe that the proposed language is one main contribution of this paper. One interesting direction for future work could be that of using a mathematical relationship (e.g., a linear function) between thresholds p and n, instead of the current threshold multiple pairs.

Feature Space. M-of-$N^{\{\neg,\vee\}}$ hypotheses are built over a set of pre-selected candidate features. While there has been a long history of applying dimensionality reduction methods, one contri-

bution of this paper is represented by an original definition of the feature space, as consisting of both terms indicative of membership and terms indicative of non-membership for a category. Unlike in the traditional feature selection approach, where only positive terms are selected, using our definition enables the learner to focus on negative information in the same way as it does with positive one (a similar approach distinguishing between positive and negative features was proposed in [140]). A criterion for the automatic detection of a suitable dimension has also been provided (see Definition 8.39). This criterion proved to be very effective in practice. Experimental results showed indeed that a few tens of well-selected features are sufficient to build accurate prediction functions, irrespective of the data set.

GAMoN biases. Apart from the language bias, we can characterize GAMoN in terms of both a search bias and an overfitting avoidance bias [**?**]. We have extensively discussed about the former, which refers to the way the hypothesis space is searched through the subsumption relationships by means of the task-dependent genetic operators. The proposed approach, actually not new in inductive learning (see, e.g., [20, 28, 41, 56, 57, 90]), overcomes a major problem in the use of conventional GAs which do not take into account the structure of the search space. The overfitting avoidance bias is a preference for simpler classifiers. GAMoN includes such a bias in the induction mechanisms by using suitable feature probability distributions (see Sub-section 8.7.4) which enable the reproduction operators to select few, high-quality features. In combination with the proposed feature selection technique, which provides GAMoN with an effective lexicon, capable of expressing the essential patterns, the overfitting avoidance bias guarantees the induction of classifiers that are parsimonious, made of a handful of well selected features. This makes them effective on the unseen data, as few high-quality features drastically reduce the risk of overfitting the training data.

GAMoN, C4.5 and Ripper. Unlike GAMoN, the two rule-based non-evolutionary classifiers used in this work, notably, C4.5 and Ripper, achieve the full expressive power of DNF. Despite this, the conducted experimental study showed that they do not outperform GAMoN. This is a clear proof of the effectiveness of the proposed algorithm.
In addition, GAMoN performs significantly more efficiently than both C4.5 and Ripper on large data sets, as the graphs of Figure 10.3 show. This should not be surprising, as the time complexity of Ripper is $O(mlog^2m)$ and that of C4.5 $O(m^3)$, while the complexity of GAMoN is $O(m)$, where m the number of examples in the training set (see Sub-section 8.7.6). That is, GAMoN can scale up to large and realistic real-world domains better than the other two rule-based classifiers. We point out that further improvements of the learning times may be obtained by, e.g., a more efficient implementation of the task-specific reproduction operators and, in a real application environment, by distributed approaches. Current research is likely to further improve efficiency of GAMoN.

GAMoN, OlexGA and BioHEL. GAMoN is a substantial extension of OlexGA from two respects. The first is the language. An OlexGA hypothesis is the special case of a M-of-$N^{\{\neg,\vee\}}$ atom where both thresholds are equal to 1, i.e., $\langle Pos, Neg, (1,1)\rangle$ (thus, M-of-$N^{\{\neg,\vee\}}$ is strictly

more expressive than the language of OlexGA). The second is the genetic algorithm. Since the hypothesis space of OlexGA does not provide any structure, OlexGA relies on a simple, standard GA, where the population is made of fixed-length individuals, and the reproduction operators are the standard uniform crossover and mutation. That is, OlexGA is a special case of GAMoN. As shown in Table 10.3, the proposed extension results in a statistically significant improvement over OlexGA. Needless to say, the price for that is a slower learning procedure.

BioHEL (Bioinformatics-oriented Hierarchical Evolutionary Learning) is a state-of-the-art GA which showed to perform very effectively on non-textual data sets [21]. To the best of our knowledge, this is the first study where BioHEL has been tested on text classification problems. BioHEL inherits several features from GAssist. It relies on the Pittsburg representation approach and applies the iterative rule learning approach [129]. BioHEL was explicitly designed to handle large-scale datasets. To this end, a rule, instead of coding all the domain attributes, keeps only a subset of them, thus avoiding hundreds of irrelevant computations. Using such an approach, BioHEL is able to handle problems with hundreds of attributes (in datasets with large sets of instances [48]) or even tens of thousands of attributes (but with few instances). In addition, in order to further reduce the computational cost, BioHEL uses a windowing scheme called ILAS (incremental learning with alternating strata). The experimental results of this paper confirm that BioHEL behaves quite efficiently, with a learning time similar to that of GAMoN (see Table 10.6). On the contrary, in terms of predictive accuracy, it showed to be statistically inferior to GAMoN. However, we feel that better results could be obtained by a finer tuning of the system. For an instance, a recent publication [**?**] shows that BioHEL has a parameter which is highly problem sensitive, the coverage breakpoint. Also, an appropriate use of the ILAS windowing scheme, as well as the usage of the C++ implementation[1] (in place of the KEEL implementation), could further improve efficiency.

Other systems learning with negation. As already mentioned, using negative evidence is deemed important in the text classification task. However, apart from OlexGA and GAMoN, none of the experimented systems focuses on the exploitation of negative information. In general, examples of IRL (Inductive Rule Learning) approaches that involve the direct generation of negation are very rare (see, e.g., [**?**, 111]). Outside the realm of rule learners, Complement Naive Bayes (CNB) is among the few text classifiers that leverage negative features [109]. Its peculiarity is that of learning the weights for a class using all training data not in that class. CNB works in a multi-class setting (i.e., it needs at least 3 classes). In [109], the authors claim that CNB approaches the state-of-the-art accuracy of SVMs. Unfortunately, we could not compare GAMoN with CNB in our empirical study, as the (binary) one-versus-all technique was used to deal with multi-label classification (basically, the problem was that Weka does not provide support for a multi-label data set representation that would be necessary in order to provide CNB with the *same* input of the other systems.)

Other MofN approaches. Several research works have recently been done to develop methods

[1] Available at http://icos.cs.nott.ac.uk/software/biohel.html

for inducing M-of-N concepts but, to the best of our knowledge, none for text categorization. For an instance, in [119] a technique for extracting M-of-N hypotheses from neural networks is reported. However, most work in this field has been carried out for constructive induction. ID-2-of-3 [96] is a M-of-N induction algorithm which incorporates M-of-N tests in decision-tree learning. It is based on a (greedy) hill-climbing approach to get the best M-of-N hypotheses at each node of a decision tree. XofN [141] is another greedy constructive induction algorithm that learns X-of-N nominal attributes. Both ID-2-of-3 and XofN, when building a decision tree, construct a new attribute for each decision node using the local training set. More recently, a Genetic Algorithm for constructive induction has been proposed in [82]. It relies on a variable length individual representation encoding the set of N attribute-value pairs composing a X-of-N attribute. The fitness is defined as the information gain ratio of the constructed attribute. The genetic operators are the standard uniform crossover along with a mutation which is a simple variant of the standard one. A conventional niching method to foster population diversity is also used.

11.1 Conclusions

In this paper we proposed a new language, called M-of-$N^{\{\neg,\vee\}}$, for text classification, along with a GA-based approach for constructing M-of-$N^{\{\neg,\vee\}}$ hypotheses from training data.
The M-of-$N^{\{\neg,\vee\}}$ representation generalizes the classical notion of M-of-N concepts by allowing negation and disjunction. We conjectured that it is well-suited to express text classification conditions, as it complies with the so-called "family resemblance" metaphor. We have shown that the space of M-of-$N^{\{\neg,\vee\}}$ hypotheses has a structure determined by two kinds of subsumption relationships - the *feature* and the *threshold* relationships, that form complete lattices. Based on that, suitable refinement operators for an effective exploration of the hypothesis space were designed.
To induce M-of-$N^{\{\neg,\vee\}}$ hypotheses, the task-specific genetic algorithm GAMoN was proposed. It is based on the Pittsburg approach, where an individual encodes a candidate classifier, as well as on *ad hoc* GS reproduction operators which are a stochastic implementation of the refinement operators. GAMoN dynamically adapts the probability of selecting the GS operators. The population is partitioned into a number of competing sub-populations, each consisting of individuals belonging to the same hypothesis subspace. To this end, a statistical criterion for automatically detecting the dimensionality range of the feature space has been proposed.
This paper also presented empirical results obtained by extensive experiments on 13 real-world test collections in a wide spectrum of sizes - from a few hundreds to a few hundreds thousands of documents. We found that GAMoN is competitive with a large collection of state-of-the-art learning techniques belonging to different classes, and that it provides hypotheses that are compact and easily interpretable. In particular, though there are small differences in predictive accuracy between GAMoN and SMO (the latter being a bit more performant), and between GAMoN and

both Ripper and C4.5 (the latter two being a bit less performant), all such systems showed to be statistically equivalent. Whereas, GAMoN proved to be superior to the other evolutionary algorithms. In particular, it showed statistically significant improvements over its predecessor OlexGA, thus confirming the effectiveness of the proposed extension. Finally, we observed that, as we scale up the size of the data set, GAMoN performs much more efficiently then both Ripper and C4.5.

Part IV

Bibliography

Bibliography

[1] http://www.cs.umass.edu/~mccallum/data.html.

[2] http://www.cs.uic.edu/~liub/FBS/blog-gender-dataset.rar.

[3] http://web.ist.utl.pt/~acardoso/datasets/.

[4] http://www.dmoz.org/rdf.html(content.rdf.u8.gz).

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

[6] S Guan A Homaifar and G E. Liepins. Schema analysis of the traveling salesman problem using genetic algorithms. *Complex Systems*, 1992.

[7] 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.

[8] 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.

[9] 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.

[10] Ahn and Chang Wook. *Advances in Evolutionary Algorithms: Theory, Design and Practice (Studies in Computational Intelligence)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.

[11] Jesús Alcalá-Fdez, Luciano Sánchez, Salvador García, María José del Jesús, Sebastián Ventura, Josep Maria Garrell i Guiu, José Otero, Cristóbal Romero, Jaume Bacardit, Víctor M. Rivas, Juan Carlos Fernández, and Francisco Herrera. Keel: a software tool to assess evolutionary algorithms for data mining problems. *Soft Comput.*, 13(3):307–318, 2009.

[12] Riquelme JC. Alvarez JL, Mata J. Cg03: An oblique classification system using an evolutionary algorithm and c4.5. *International Journal of Computer, Systems and Signals*, 2001.

[13] 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.

[14] 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.

[15] 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.

[16] 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.

[17] Jaume Bacardit. *Pittsburgh Genetics-Based Machine Learning in the Data Mining era: Representations, generalization, and run-time*. PhD thesis, Ramon Llull University, Barcelona, Spain, 2004.

[18] Jaume Bacardit, Edmund K. Burke, and Natalio Krasnogor. Improving the scalability of rule-based evolutionary learning. *Memetic Computing*, 1(1):55–67, 2009.

[19] Jaume Bacardit, David E. Goldberg, and Martin V. Butz. Improving the performance of a pittsburgh learning classifier system using a default rule. In *Proceedings of the 2003-2005 international conference on Learning classifier systems*, IWLCS'03-05, pages 291–307, Berlin, Heidelberg, 2007. Springer-Verlag.

[20] Jaume Bacardit and Natalio Krasnogor. Performance and efficiency of memetic pittsburgh learning classifier systems. *Evolutionary Computation*, 17(3):307–342, 2009.

[21] Jaume Bacardit, Michael Stout, Jonathan D. Hirst, Kumara Sastry, Xavier Llorà, and Natalio Krasnogor. Automated alphabet reduction method with evolutionary algorithms for protein structure prediction. In *GECCO '07: Proceedings of the 9th annual conference on Genetic and evolutionary computation*, pages 346–353. ACM Press, 2007.

[22] T. Baick. Optimal mutation rates in genetic search. In *Proc. Fifth International Conference on Genetic Algorithms, Morgan Kaufmann, San Mateo, CA*, pages 2–9, 1993.

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

[24] 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.

[25] 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.

[26] 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.

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

[28] Deborah R. Carvalho and Alex A. Freitas. A hybrid decision tree/genetic algorithm method for data mining. *Inf. Sci.*, 163:13–35, June 2004.

[29] 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.

[30] 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.

[31] Frans Coenen and Paul Leng. The effect of threshold values on association rule based classification accuracy. *Data Knowl. Eng.*, 60:345–360, February 2007.

[32] 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.

[33] 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.

[34] 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.

[35] 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.

[36] 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 [37].

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

[38] 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.

[39] 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.

[40] Janez Demšar. Statistical Comparison of Classifiers over Multiple Data Sets. *Journal of Machine Learning Research*, 7(1):1–30, 2006.

[41] Federico Divina, Maarten Keijzer, and Elena Marchiori. A method for handling numerical attributes in ga-based inductive concept learners. In *GECCO*, pages 898–908, 2003.

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

[43] 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.

[44] 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.

[45] Alberto Fernández, Salvador García, Julián Luengo, Ester Bernadó-Mansilla, and Francisco Herrera. Genetics-based machine learning for rule induction: state of the art, taxonomy, and comparative study. *Trans. Evol. Comp*, 14:913–941, December 2010.

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

[47] 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.

[48] M.A. Franco, N. Krasnogor, and J. Bacardit. Speeding up the evaluation of evolutionary learning systems using gpgpus. In *Proceedings of the 12th annual conference on Genetic and evolutionary computation - GECCO '10*, pages 1039– 1046, 2010.

[49] AA Freitas. *In Springer-Verlag, editor,* Advances in Soft Computing-Engineering Design and Manufacturing*, title = A genetic algorithm for generalized rule induction, year = 1999*.

[50] Alex A. Freitas. *Data Mining and Knowledge Discovery with Evolutionary Algorithms*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2002.

[51] 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.

[52] 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.

[53] Evgeniy Gabrilovich and Shaul Markovitch. Text categorization with many redundant features: Using aggressive feature selection to make svms competitive with c4.5. In *In ICML04*, pages 321–328, 2004.

[54] 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.

[55] 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.

[56] Anglano Giordana, C. Anglano, A. Giordana, G. Lo Bello, and L. Saitta. A network genetic algorithm for concept learning. In *Proceedings of the Sixth International Conference on Genetic Algorithms*, pages 436–443. Morgan Kaufmann, 1997.

[57] Attilio Giordana, Lorenza Saitta, and Floriano Zini. Learning disjunctive concept definitions using a genetic algorithm. In *ECAI*, pages 483–486, 1994.

[58] 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.

[59] David E. Goldberg. Genetic algorithms in search, optimization and machine learning. *Addison-Wesley*, 1989.

[60] David E. Goldberg and Jon Richardson. Genetic algorithms with sharing for multimodal-function optimization. In *ICGA*, pages 41–49, 1987.

[61] David Perry Greene and Stephen F. Smith. Competition-based induction of decision models from examples. *Machine Learning*, 13:229–257, 1993.

[62] Eui-Hong Han, George Karypis, and Vipin Kumar. Text categorization using weight-adjusted k-nearest neighbor classification. In David Cheung, Qing Li, and Graham Williams, editors, *Proceedings of PAKDD-01, 5th Pacific-Asia Conferenece on Knowledge Discovery*

and Data Mining, pages 53–65, Hong Kong, CN, 2001. Springer Verlag, Heidelberg, DE. Published in the "Lecture Notes in Computer Science" series, number 2035.

[63] 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.

[64] 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.

[65] Francisco Herrera. Genetic fuzzy systems: Status, critical considerations and future directions. *International Journal of Computational Intelligence Research*, 1, 2005.

[66] 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.

[67] 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.

[68] Eui hong Han and George Karypis. Centroid-based document classification: Analysis and experimental results. In *Principles of Data Mining and Knowledge Discovery*, pages 424–431, 2000.

[69] M Hristakeva and D Shrestha. Solving the 0/1 knapsack problem with genetic algorithms. *Midwest Instruction and Computing Symposium*, 2004.

[70] 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.

[71] 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.

[72] Thorsten Joachims. Text categorization with support vector machines: learning with many relevant features. In *Proceedings of ECML-98, 10th European Conference on Machine Learning*, number 1398. Springer Verlag, Heidelberg, DE, 1998.

[73] 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.

[74] Thorsten Joachims. *Learning to Classify Text using Support Vector Machines*. Kluwer, 2002.

[75] Jung Y. Suh Prasanna Jog and Dirk Van Gucht. Parallel genetic algorithms applied to the traveling salesman problem. *SIAM Journal of Optimization*, 1991.

[76] J.Platt. Fast training of support vector machines using sequential minimal optimization. In *In B. Scholkopf, C. Burges, A. Smola (Eds.), Advances in Kernel methods support vector learning, Cambridge, MA: MIT Press*, 1998.

[77] 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.

[78] 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.

[79] 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.

[80] 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.

[81] 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.

[82] Otavio Larsen, Alex A. Freitas, and Julio C. Nievola. Constructing x-of-n attributes with a genetic algorithm. In *In Proc. of the Genetic and Evolutionary Computation Conference*, page 1268. Morgan Kaufmann, 2002.

[83] 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.

[84] 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.

[85] 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.

[86] 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.

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

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

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

[90] Juliet Juan Liu and James Tin yau Kwok. An extended genetic rule induction algorithm. In *Proceedings of the 2000 Congress on Evolutionary Computation (CEC00)*, pages 458–463, 2000.

[91] 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.

[92] 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.

[93] Andrew McCallum and Kamal Nigam. A comparison of event models for naive bayes text classification. In *in AAAI-98 Workshop on Learning for Text Categorization*, pages 41–48. AAAI Press, 1998.

[94] Z. Michalewicz. Genetic algorithms+data structures=evolution programs. *3th Edition, Springer-Verlag*, 1999.

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

[96] Patrick M. Murphy and Michael J. Pazzani. Id2-of-3: Constructive induction of m-of-n concepts for discriminators in decision trees. In *Proc. Of the Eighth Int. Workshop on Machine Learning, pp 183-187, Evanston, IL, 1991*, 1991.

[97] 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.

[98] Shan-Hwei Nienhuys-Cheng and Ronald de Wolf. *Foundations of Inductive Logic Programming*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 1997.

[99] A.A.; Lopes H.S. Noda, E.; Freitas. *Proc. Congress on Evolutionary Computation (CEC-99)*.

[100] E.D.; Punch W.F. Pei, M.; Goodman. Pattern discovery from data using genetic algorithms. *Proc. 1st Pacific-Asia Conf. Knowledge Discovery and Data Mining*, 1997.

[101] A. Pietramala, V.L. Policicchio, P. Rullo, and I. Sidhu. A genetic algorithm for text classification rule induction. In *Proceedings of the European conference on Machine Learning and Knowledge Discovery in Databases - Part II*, ECML PKDD '08, pages 188–203, Berlin, Heidelberg, 2008. Springer-Verlag.

[102] Leonard Pitt and Leslie G. Valiant. Computational limitations on learning from examples. *Journal of the ACM*, 35:965–984, 1988.

[103] V. L. Policicchio, A. Pietramala, and P. Rullo. A ga-based learning algorithm for inducing m-of-n-like text classifiers. In *Proceedings of the 10th International Conference on Machine Learning and Applications and Workshops, ICMLA, vol. 1, pp.269-274*, 2011.

[104] Christoph Schommer Qin Sun and Alexander Lang.

[105] J. R. Quinlan. Generating production rules from decision trees. In *Proceedings of the 10th international joint conference on Artificial intelligence - Volume 1*, pages 304–307, San Francisco, CA, USA, 1987. Morgan Kaufmann Publishers Inc.

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

[107] J. Ross Quinlan. Learning logical definitions from relations. *Machine Learning*, 5:239–266, 1990.

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

[109] Jason D. Rennie, Lawrence Shih, Jaime Teevan, and David R. Karger. Tackling the poor assumptions of naive bayes text classifiers. In *ICML*, pages 616–623, 2003.

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

[111] P. Rullo, L.V. Policicchio, C. Cumbo, and S. Iiritano. Olex: effective rule learning for text categorization. *IEEE Transactions on Knowledge and Data Engineering*, 21:1118–1132, 2009.

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

[113] 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.

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

[115] 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.

[116] 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 [117].

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

[118] Fabrizio Sebastiani. Machine learning in automated text categorization. *ACM Comput. Surv.*, 34:1–47, March 2002.

[119] R. Setiono. Extracting m-of-n rules from trained neural networks. *IEEE Transactions on Neural Networks*, 11:512–519, 2001.

[120] R. Setiono, S.L. Pan, M.H. Hsieh, and A. Azcarraga. Automatic knowledge extraction from survey data: learning m-of-n constructs using a hybrid approach. *Journal of the Operational Research Society*, page 314, 2005.

[121] 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.

[122] 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.

[123] Alireza Tamaddoni-Nezhad and Stephen Muggleton. A genetic algorithms approach to ilp. In *Proceedings of the 12th international conference on Inductive logic programming*, ILP'02, pages 285–300, Berlin, Heidelberg, 2003. Springer-Verlag.

[124] Geoffrey G. Towell and Jude W. Shavlik. Extracting refined rules from knowledge-based neural networks. *Machine Learning*, 13:71–101, 1993.

[125] 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.

[126] 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.

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

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

[129] Gilles Venturini. SIA: A supervised inductive algorithm with genetic search for learning attributes based concepts. In *Machine Learning: ECML-93*, pages 280–296. 1993.

[130] 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.

[131] 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.

[132] Stewart W. Wilson. Classifier fitness based on accuracy. *Evol. Comput.*, 3:149–175, June 1995.

[133] 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.

[134] 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.

[135] 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.

[136] 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.

[137] 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.

[138] X. Yin and J Han. Cpar: Classification based on predictive association rules. In *Proceedings of the SIAM International Conference on Data Mining*, pages 331–335, 2003.

[139] 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.

[140] 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.

[141] Zijian Zheng. Constructing x-of-n attributes for decision tree learning. *Machine Learning*, 40(1):35–75, 2000.