Thesis

Automatic Text Categorization of documents in the High Energy Physics domain

Dr. Luis Alfonso Ureña-López (supervisor)
Dr. Ralf Steinberger (supervisor)

Arturo Montejo-Ráez (author)

15 December, 2005

Contents

List of Figures

List of Tables

Chapter 1

Preface

"Necesaria es la experiencia para saber cualquier cosa"

El Libro de Oro, Séneca

This research aims studying the possibility of an automatic method for key word assignment from a controlled vocabulary to documents in the domain of High Energy Physics, and proposing a real solution as result of such study. The problem was detected to be a *text categorization* problem, where key words are the considered categories. During the development of this work, done mostly at CERN, the European Organization for Nuclear Research[1], the document collection presented several problems not covered by the literature. This expressed the need of a solution that could not be just a prototype used for testing the hypothesis proposed along this work.

The results of the final and implemented solution as product of the research have turned into a wide range of applications, giving me the pleasant impression of usability we may lack in pure research. The reader will find in this text how exciting this task was, but something that cannot be included here is the personal enrichment acquired by working in an international environment together with a team focused on bringing top computer-based techniques to the user community of the CERN library.

1.1 Acknowledgements

This research was started at the European Laboratory for Particle Physics, when I received an scholarship from the Spanish Ministry of Science and Technology, within the programme "Becas de Formación para Personal Investigador en el Extranjero" (FPIE). After this two year period I continued my work at CERN

[1] Also known as European Laboratory for Particle Physics

under the Doctoral Student Programme. I am also member of the SINAI[2] group at the Department of Computer Science of the University of Jaén, who is partially supported by Spanish Government (MCYT) with grant TIC2003-07158-C04-04.

If working at CERN has been very pragmatical, being distanced from my PhD directors has been a main drawback. The continuous interchange of ideas and results, the track of development and experiments has suffered from such reality and, although it did not block the process, certain delays were introduced as consequence. But the enthusiasm by which both of them believed in my work pushed me towards the right direction in many times. I have to start, therefore, giving my acknowledgements to Luís Alfonso Ureña López (University of Jaén, Spain) and Ralf Steinberger (Joint Research Center, European Commission, Ispra, Italy) for being the experts behind the novice.

The support of the CERN Document Service team has being enormous, funding me for more than two years. I have to mention here Jens Vigen, CERN Library leader and Jean-Yves Le Meur, CERN Document Service leader both for supervising this work from the practical and technical point of view. Also my thanks to David Dallman, Tibor Simko and Corrado Pettenati for their sincere interest in the project. I would like to give my acknowledgements to Robert Cailliau, leader of the Web Public Education group, and Juan Antonio Rubio, leader of Education and Technology Transfer division (ETT), for hiring me in their team allowing this research to be completed at CERN.

In addition to this people, my thanks to the Department of Computer Science in the University of Jaén and specially the rest of the SINAI group. This fantastic team will produce some of the best advances in Information Retrieval in the future, I am certainly sure about that.

A special and beloved mention to my wife, Cristina, who has being close to me and has given me all her strength for staying in Switzerland and finishing this work successfully. She has suffered this research even more than me. I will never forget her support and how much she has believed in me even during the most difficult periods.

Finally, I dedicate this dissertation to my father, who left us during the development of this research. He will not be able to be proud of me as he used to be no matter what I do, so I will be forever proud of him and proud of my mother for having worked hard all their life to give me the opportunity of becoming what I wanted to be.

1.2 Content overview

The text is organized in a classical manner, introducing the problem with its paradigms, current system and theoretical analysis, and proposing later a set of

[2] http://sinai.ujaen.es

experiments to validate the offered approach.

This work focuses on the problem of automatic indexing of electronic versions of documents in a multi-label way (several classes per document). It deals mainly with a collection of *High Energy Physics* (HEP) related papers taken as samples from the vast digital library of CERN. This collection shows certain properties which deserved the special attention that motivated present research.

In chapter 2 a detailed introduction to the framework where this research takes place is provided, describing CERN and its documentary issues and problems. Also a general overview of the human indexing process based on the use of thesauri is presented. In chapter 3 the reader will find an overview of the research area facing the problem of automatic categorization of text documents, along with notational conventions and the architecture that represents our proposal for text categorization of multi-labeled collections with high class imbalance degree.

Unveiling the different techniques involved in automatic text classification by using supervised learning algorithms will take chapter 3. Chapter 4 introduces some feature extraction approaches. Chapter 5 provides different strategies for reducing the number of features used to represent a text document. Finally, chapter 6 explains how the multi-label classifier is built up from known base classifiers.

In chapter 7, evaluation issues and related discussion are proposed. Chapter 8 analyzes in detail all the current and potential practical applications of these automated systems, exploring the wide range of possibilities offered by computer based categorization. Later on, in chapter 9, an historical review of existing solutions and systems in production follows, pointing out their major benefits and drawbacks.

The second part of the text really defines the proposed solution to the problem of multi-label assignment. It starts at chapter 10, where our system is decomposed to ease describing every single process involved. The data handled is introduced and its properties detailed at chapter 11. At this point, we are ready to being with chapter 12, containing the huge experimental work performed on the former data, to study the feasibility of the general model proposed. These experiments (from section 12.1 to section 12.11) are intended to treat almost every single aspect involved in the empirical definition of our approach. Last documented experiment (section 12.12) studies the behaviour of our system on a totally different set of data, not related to High Energy Physics (HEP) research.

Finally, chapter 13 summarizes main conclusions stated during the whole text, and establishes possible future trends and open issues for multi-label automatic categorization of documents. The work ends with the bibliography followed during the our research and a set of appendices with further additional information of aspects of this work that we have considered were worth detailing to enhance the clarity of the text.

Chapter 2

CERN and the assignment problem

2.1 The paradigm: High Energy Physics research

2.1.1 The European Laboratory for Particle Physics

In 1951, a provisional body was created called the "Conseil Européen pour la Recherche Nucléaire" (CERN). This was a council: a body of people. In 1953 the Council decided to build a central laboratory near Geneva. At that time, pure physics research concentrated on understanding the inside of the atom, hence the word "nuclear".

The official name of the laboratory is European Organization for Nuclear Research. It is the largest particle physics centre in the world. At this centre physicists come to explore what matter is made of and what forces hold it together. It exists primarily to provide them with the necessary tools: accelerators. These machines accelerate particles to almost the speed of light and detectors to make the particles visible. The accelerator complex at CERN is a succession of machines with increasingly higher energies, injecting the beam each time into the next one, which takes over to bring the beam to an energy even higher, and so on. The flagship of the complex will be the Large Hadron Collider.

The scientific and technical staff designs and builds the laboratory's intricate machinery and ensures its smooth operation. It also helps prepare, run, analyse and interpret the complex scientific experiments. Some 6500 visiting scientists, half of the world's particle physicists, come to CERN for their research. They represent 500 universities and over 80 nationalities. CERN was founded thanks to the joint efforts of 12 European States. Nowadays, thanks to this venture,

most of the European countries have the chance to participate in the world's most advanced physics experiments without consuming an unacceptable fraction of their national science budget.

Founded in 1954, the laboratory was one of Europe's first joint ventures and includes now 20 Member States. CERN employs just under 3000 people, representatives of a wide range of skills - physicists, engineers, technicians, craftsmen, administrators, secretaries, workmen, and more. The central role is played by researchers from all around the world working in basic science exploring the basic components of the Universe. For these reasons the scientific production of CERN is huge.

2.1.2 The CERN Document Server

Actually the CERN Document Server (CDS)[1] is a reference catalogue, repository and digital library for High Energy Physics (HEP) publications. Its database contains more than 660,000 records and 320,000 full-text documents organized into more than 500 different collections covering pre-prints, articles, books, journals, photographs, and much more.

The main services offered by CDS are:

1. *Search.* Permits to search through bibliographic information and full-text documents stored in CDS databases. Offers personal baskets, email alerts, and more.
2. *Submissions.* Permits the electronic submission of documents from inside and outside of CERN.
3. *Conversion.* Offers a possibility to convert user-uploaded documents to different document formats (PDF, Microsoft Word, etc.).
4. *Scanning.* Offers a possibility to have your paper-based documents scanned. (CERN Intranet only).
5. *Agenda.* Offers resources to plan meetings and workshops, to store presentations and minutes.
6. *Webcasting.* Archived videos of presentations done at CERN as well as a real-time webcasting of events.
7. *Bulletin.* Electronic version of the CERN Weekly Bulletin.

The data stored in the CDS server is a massive collection of multimedia information including pictures, full-text documents, videos, presentations, and other information. All this data is indexed within the catalogue and accessible

[1] http://cds.cern.ch/

from a common interface. Searching and browsing by the user is independent of the nature of the document. Due to the amount of data available, we can understand why they are interested to profit from such an enriched database to even to the point of performing data-mining operations (clustering, text classification, ranked searches, and so on).

Almost the totality of software used to build up CDS is based on *Open Source* solutions. The CDS team releases the software used under the *CDSware* package. The CERN Document Server Software (CDSware) is the software developed by, maintained by, and used at, the CERN Document Server. It allows you to run your own electronic pre-print server, your own on-line library catalogue or a document system on the web. It complies with the Open Archives Initiative meta-data[2] harvesting protocol (OAI-PMH) and uses MARC 21 as its underlying bibliographic standard. The CDSware is free software, licensed under GNU General Public Licence (GPL)[3].

2.2 The problem: Keyword assignment process

In the HEP environment many papers are written, they come under the form of final articles or pre-prints. The problem of their storage is resolved in several ways: storing a new record on paper with meta-data about the item, or using a large database. Relational databases deal with such a problem efficiently. The searching for a document turns to be a problem implying a more complex solution. How to reach a document among thousands, even millions? If we are looking for an exact item, that is, we know perfectly what we want, then the database system should be more than enough to retrieve for you the desired document. But if we do not know exactly which documents can be the answer to our information needs, then it is not so easy to retrieve a good set of relevant documents.

Traditionally at libraries, indexes and categories are used to "label" documents in such a way a user can try to search using this added meta-data in order to find items more or less related to some premises. For example, a user could need documents talking about technical tips on motors and cars. Using a subject tree, that is, a hierarchy of categories, he could follow a path in the tree of categories until reach a branch with a limited set of documents where he can start a deeper search. Users may follow the Universal Decimal Classification. It is an indexing and retrieval language in the form of a classification for the whole of recorded knowledge, in which subjects are symbolized by a code based on Arabic numerals. He could limit his search to the category 68 (`assembled articles and precision mechanisms`, under the category 6, `technology`).

Even a very small space of searching with few documents can require hard work to review each item in it. To address this problem more meta-data is

[2] More info at `http://www.openarchives.org/`
[3] Details at `http://www.gnu.org/licenses/gpl.html`

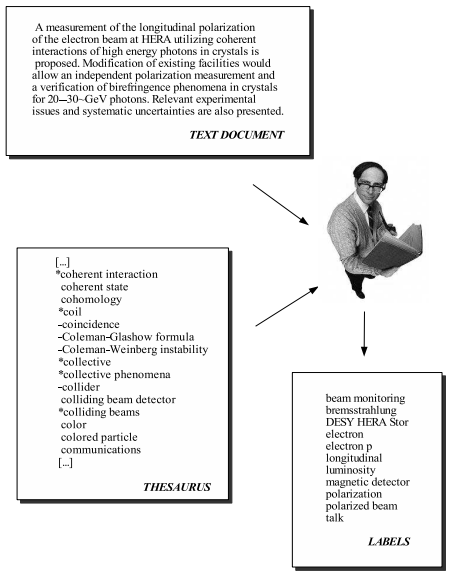


Figure 2.1: *The manual indexing process*: an expert is responsible of reading the text and selecting those topics from the thesaurus that best represent the content of the document.

added to items in order to facilitate the task of "over-viewing" because often the title is not enough to determinate if an item is relevant or not. A common way to do this is to attach some "key words" to the document. These key words try to synthesize the content of the document in few words (from ten to twenty key words usually).

2.2.1 Subject keys in traditional information systems

If we had to organize our personal library, what sort of ideas we would try in order to achieve well organized shelves? Maybe one of the first ideas is to group books by theme, then to label them and put those references in a kind of index. Later on we might find we have so many books that it is better to arrange them by size (actually, large repositories do so). Anyhow, at the end, we will have to *index* them, in one way or another. Now the question must be: which indexes should I use?. It is not an easy task to define them, because several considerations must be taken into account. Vickery already emphasizes this reality ([122]):

> The problem of subject representation is therefore far less straightforward than other aspects of document description.

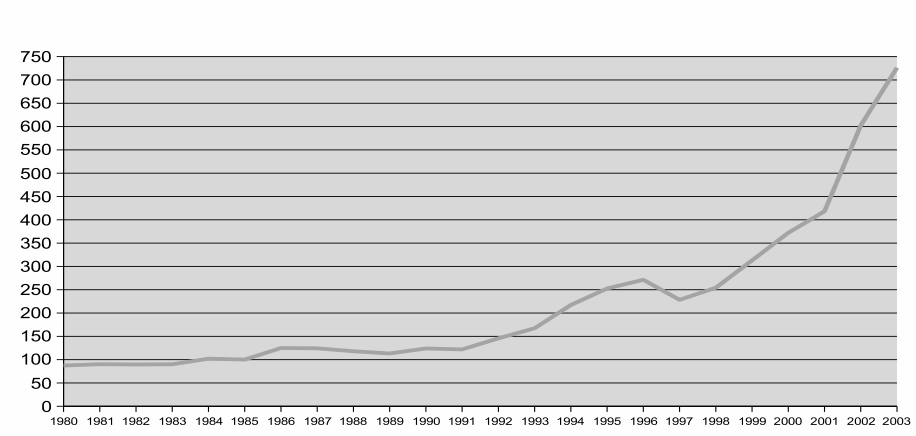


Figure 2.2: Average number of papers weekly submitted to CDS per year

In the beginning, the use of key words in information storage and retrieval was due to two major needs: the need for classification and the need for retrieval. The former one had a double benefit: first, it let librarians organize physical volumes into logical clusters; second, the possibility of search within a defined cluster was regarded as a way to speed up the searching for information (as pointed by the so called *cluster hypothesis* of Rijsbergen [118]).

Hence, two major goals of key-word assignment are:

1. Select records in a file that deal with a specific topic
2. Group in proximity in a file records of similar subjects

The use of alphabetical terminologies and classification structures (known as *thesauri*) were invented as tools to improve the two main **measures** in information retrieval: *precision* and *recall*. They refer to quality of retrieved documents. Precision is the number of relevant documents retrieved over the total number of documents retrieved. Recall is the number of relevant documents retrieved over the total number of relevant documents in the collection. These two measures show the problem of an antagonistic relationship: usually, if we try to improve one of them the other will decay. For example, if we retrieve the whole collection as answer for a given query, our recall will be 100% of course, but our precision will be so low that the result will be untreatable. The challenge resides then in finding a method which shows a good performance for both measures.

In earlier retrieval systems, the use of some techniques aim to improve these two values for a defined retrieval system; i.e. the implementation of these techniques must be oriented to the purpose and content of the retrieval system. The techniques traditionally used rely on setting relationships between words in a controlled vocabulary. Using those relations we can operate on a given query to

improve recall (by expanding to related terms) or precision (by narrowing with less generic terms). These are the reasons for the use of thesauri.

2.2.2 Thesauri

Definition

There are several definitions for the word *thesaurus*. The classic one is (taken from *Webster's Revised Unabridged Dictionary*):

> "A treasury or storehouse (from Greek); hence, a repository, especially of knowledge; often applied to a comprehensive work, like a dictionary or encyclopedia"

There is another definition closer to our idea in information retrieval paradigms:

> "A book of words or of information about a particular field or set of concepts; especially: a book of words and their synonyms or a list of subject headings or descriptors usually a cross-reference system for use in the organization of a collection of documents for reference and retrieval."

As we can see, this last one is quite complete, covering both the purpose and structure of a thesaurus. Already in this definition we find words like *concept*, *subject*, *descriptor*. More forms are used to specify thesaurus's entries, e.g. *topic*, *key-word*, *theme*, *class*... We will refer to these entries, from now onwards, using the expression "key word".

In an old work of Vickery [122] we find a definition of thesaurus which summarizes in few words the rationale associate with it:

> "The thesaurus is a display of the terms in a retrieval language showing semantic relations between them."

Here, Vickery shows on the one hand the main purpose of a thesaurus: it defines a retrieval language, whatever the retrieval method might be. On the other hand, he does not set the kind of relationships between entries (synonymy, broader terms...), specifying only that a set of semantic relations is defined. We will see that this brief definition fits perfectly with any type of existing thesaurus.

Evolution of thesauri

Leibniz attempted the classification of concepts as preliminary to invent a *Universal Language* [92] (which was composed by symbols rather than words as it was supposed to be universally understood).

One of the earliest thesauri (and maybe the most famous one) is *Roget's Thesaurus* [70]. The main idea of Dr. Peter Mark Roget behind this compilation was to offer a system which would offer words to express a given meaning, while traditional dictionaries offer meanings for a given word. This would help writers to express their concepts in most suitable expression form. These kind of users had the thesaurus as a reference book in the generation of texts. Thus, it was mostly intended to be useful in the phase of generation of documents.

The power of reducing a language to its basic concepts has become more and more useful, even more when such "semantic network" arises in electronic form. WordNet ([44]) is *an on-line reference system* (as their authors claim). English nouns, verbs, adverbs and adjectives are organized into *synonym sets* (also called *synsets*), each representing one underlying lexical concept. Nowadays we can assure that almost every thesaurus (specialized or not) is available in electronic form.

There is even a multilingual thesaurus based on WordNet called EuroWordNet ([123]), which maps synsets between different European languages. This work represents a main milestone in multilinguistic information retrieval. This thesaurus is so rich that it is usually seen not as a traditional thesaurus, but as a complete lexical database.

Both WordNet and Roget's Thesaurus are general references, i.e. they don't focus on specialized terminologies. But the particular areas where thesauri become useful tools are in specialized domains (Law, Medicine, Material Science, Physics, Astronomy...). One example is the INSPEC thesaurus ([6]), focused on technical literature; or the ASIS thesaurus, specialized in Information Science ([1]). Also NASA, the European Union and other organizations produce their own specialized thesauri (like the multilingual EUROVOC thesaurus [2]).

Structure

Each thesaurus has its own organization, according to the purpose to accomplish. But we can split any of them into the following components:

Terms this is the set of items in the thesaurus. They are usually referred to as descriptors, index terms, key words, key phrases, topics, concepts or themes. We will use "key word" to name them, being our categories, classes or labels, as we progress in the present text.

Meanings this is the set of subsets of the set of terms. Each subset in the set is a group of terms which are interrelated by the *synonymy* relationship (i.e. words with the same meaning). This relation is very important because the resulting subsets are elements in other relations.

Relationships this is a set of relations term to term, term to meaning, meaning to term and meaning to meaning.

```
penalty
NT1   alternative sentence
NT1   carrying out of sentence
   NT2   barring of penalties by limitation
   NT2   reduction of sentence
         RT   repentance
         RT   terrorism     (0431)
   NT2   release on licence
   NT2   suspension of sentence
NT1   conditional discharge
NT1   confiscation of property
      RT   seizure of goods     (1221)
NT1   criminal record
NT1   death penalty
NT1   deprivation of rights
      RT   civil rights     (1236)
```

Figure 2.3: Excerpt from Eurovoc thesaurus

There are several relations which are commonly used among existing thesauri:

- **Hyponomy**. This is a relationship between meanings. We say that *x is a hyponomy of y* if *x is a kind of y*; for instance, *orchid* is an hyponomy of *flower*. This relation is reflexive, anti-symmetric and transitive, therefore it establishes a *partial order* between meanings. The symmetric relation is called *specialization* and also defines a partial order over the set of terms.

- **Meronymy**. This can be split into three different (but closer) relationships:

  1. *x is part of y*, e.g. *branch* is part of *tree*
  2. *x is a member of y*, e.g. *citizen* is member of *society*
  3. *x is constituent material of y*, e.g. *iron* is constituent material of *knife*

Of course, depending on the purpose of the thesaurus, some of these relations may be ignored. Also new relations could happen. WordNet, for example, includes all relations above. INSPEC and EUROVOC thesauri condense meronomy relations into the "related" one (see figure 2.3, *RT* means "related terms"). Synonymy is implemented by the application of the "USE" statement.

Usually in specialized thesauri either the synonymy is neglected or a preferred word form as representative of the meaning is given, since the purpose of them is to provide a list of controlled terms (and that "control" refers to the

```
*coherent interaction
 coherent state (for quantum mechanical states)
 cohomology
*coil
-coincidence ('fast logic' or 'trigger' or 'associated production')
-Coleman-Glashow formula (baryon, mass difference)
-Coleman-Weinberg instability (symmetry breaking)
*collective (used only in connection with accelerators)
*collective phenomena ('field theory, collective phenomena' or
'nuclear physics, collective phenomena' or 'nuclear matter,
collective phenomena')
-collider ('storage ring' or 'linear collider')
 colliding beam detector (use only in instrumental papers)
*colliding beams (for accelerator use 'storage ring' or
'linear collider')
 color (for colored partons)
 colored particle
 communications
```

Figure 2.4: Excerpt from DESY thesaurus

use of just one word form for a given meaning). Nevertheless, is also true that most of them includes synonymy in one way or another.

There are, however, some special cases of thesauri where there are more than just terms and relations. In some cases the thesaurus is a complex reference of specific relations, with specially defined rules to build a document's key words. This is the case of the DESY [32] thesaurus, specialized in High Energy Physics literature. With the entries given we construct particle combinations, reaction equations and energy declarations among other things. Practical issues take us to the conclusion of Vickery about the need of tight relation between the thesaurus and its domain of retrieval.

2.2.3 DESY's thesaurus

At the library of CERN hundreds of new papers arrive every day. And this arrival is increasing its size year by year, as we can see in figure 2.2. The task of indexing is performed mainly by indexers working at the Deutsche Elektronen-Synchrotron (DESY), the German laboratory. They have developed a thesaurus, to control the vocabulary used in the key-wording process. Due to the growth in the production of HEP related papers a new approach to manual key-wording is required. Since full computer-based indexers are still far away from a real solution, at least a computer-based help tool for indexing should be supplied in order to decrease the charge of work on human indexers.

The DESY Documentation group in Hamburg developed the HEPI (High Energy Physics Information) system from 1963 onwards. In this scheme all documents in the HEP field are indexed by subject specialists who read the entire articles. The thesaurus used contains approximately 2500 terms and has in general been updated every 1-2 years.

The DESY key words have been included with every article in the SLAC HEP database since this database started in mid 1970s. These key words serve the following purposes: they allow the generation of a subject index for the biweekly periodical High Energy Physics Index, they are important for computerized information retrieval and SDI (Selective Dissemination of Information) service at DESY and other high-energy physics centers. The total key words assigned to a paper may also be useful as a sort of abstract.

There are three types of key words in the thesaurus:

1. main key words
2. descriptive (secondary) key words
3. non-key words

Keywords may be used singly or coupled by comma and blank (for example: 'field theory' (single) and 'field theory, nonabelian' (coupled)). While the first term is generally a main key word, the second term may be another main key word, a descriptive key word or a non key word. Non key words which are frequently used are standardized.

For our actual purposes, only main DESY key words are used. The reason is that already most of the main key words appear seldom, so we believe it is not feasible to work with the secondary key words without first studying the distribution of secondary key words across the collection, which is far outside the scope of this work.

For more information about the use of key words by human indexers, please read appendix A.

2.3 The approach: supervised text categorization

Automatic Text Categorization (TC) is a prominent research area within Information Retrieval, and will be introduced it in chapter 3. This discipline studies the problem of classifying automatically (i.e. computer-based) a text document in electronic format into a set of predefined categories. During the discourse of this document, a method for automatic text categorization of documents in multi-label domains is introduced, proposing an approach for assigning multiple possible categories to every single text document at once.

We will adapt such techniques and define a method for resolving the problem of automatic labeling at CERN, facing all the arising problems that the High Energy Papers and the DESY thesaurus may establish. Two main results are expected from this research: the proposal of a innovative method for multi-label text categorization within certain constraints, and the definition of a new benchmark collection. Not additional attention to the definition of the approach is provided here, as later chapters will cover it extensively.

The corpus we worked with is in the English language, but the statistical methods are, of course, language-independent. Nevertheless, the DESY thesaurus has been used to label non-English texts, though in our current research we have not paid attention to additional languages. Some experimental results depending on certain linguistic resources, like stemming or multi-word recognition, may differ when applying our method to other languages. This could be considered as matter for further research, not covered in the present work.

Chapter 3

Text Categorization and Machine Learning

The automatic assignment of key words to documents using full-text data has been enclosed inside the growing area of *Text Categorization*, an area where Information Retrieval techniques and Machine Learning algorithms meet, offering solutions to problems with real world collections.

Text Categorization (TC) is a discipline responsible for the automatic classification of text documents under predefined categories or classes (see [19] for an introduction). The Text Categorization task lies under the Automatic Classification (also known as Pattern Recognition) problem in Machine Learning. If we perform an unsupervised classification, then, there is no clue about the possible final classes. Thus, we use unsupervised learning techniques to find possible groups or classes, i.e. Document Clustering. If we have a predefined class or classes, then we will likely use supervised techniques (whenever a training collection is also available) to achieve an approximation to the solution of the problem. The latter is what is commonly known as Text Categorization, leaving Document Clustering as a different discipline.

We can easily identify three paradigms in text categorization, as shown in figure 3.1: the *binary* case, the *multi-class* case and the *multi-label* case.

- In the *binary case* a sample belongs to exactly one of two given classes. Thus, the classifier has to determine to which of the two sets the sample goes.
- In the *multi-class case* a sample belongs to just one class of a set of m classes.
- Finally, in the *multi-label case*, a sample may belongs to several classes at the same time, that is, classes may *overlap* through documents.

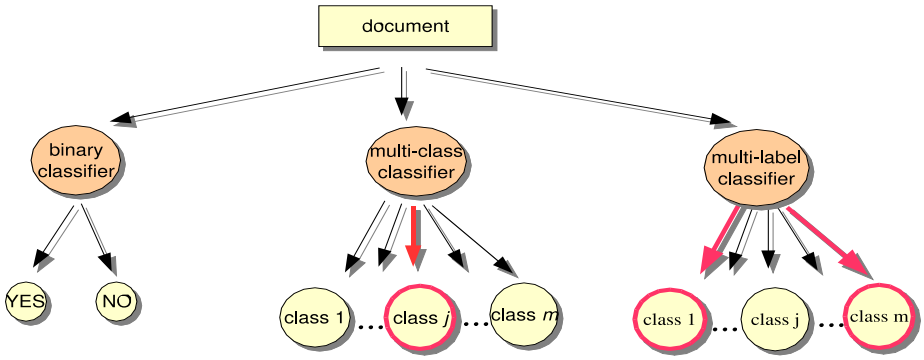


Figure 3.1: Paradigms in text categorization: binary, multi-class and multi-label cases.

In binary classification a classifier is trained, by means of supervised algorithms, to assign a sample document to one of two possible sets. These two sets are usually referred to as belonging samples (positive) and not belonging samples (negative) respectively (the one-against-all approach), or to two disjoint classes (the one-against-one approach). For this we can select among a wide range of algorithms, being Naïve Bayes, Linear Regression, Support Vector Machines (SVM) [52] and LVQ [77] the main candidates for binary classification (with a prominent relevancy of SVM in performance). The binary case has been set as a base case from which the two other cases can be built. In multi-class and multi-label assignment, the traditional approach consists on training a binary classifier for every class, and then, whenever the binary base case returns a measure of confidence on the classification, assigning either the top ranked one (multi-class assignment) or a given number of the top ranked ones (multi-label assignment). More details about these three paradigms can be found in [7], where Allwein and others propose a common framework to reduce any multi-class problem into a binary approach for margin based classifiers. We will refer to the ranking approach as the *battery* strategy, because no consideration of inter-dependency is considered.

Another approach for multi-labeling consists on returning all those classes whose binary classifiers provide a positive answer for the sample. It has the advantage of allowing different binary classifiers through classes, as inter-class scores do not need to be coherent (since there is no ranking afterwards). Better results when applying one-against-one in multi-class classification have been reported [7], but in our multi-label case this cannot be followed because, in principle, any class may appear along with any other class, it being difficult to establish disjoint assignments. This is the reason why one-against-all deserves our attention in the present work.

Although much research is taking place (see [112]), some topics demand more attention than what has been given to them so far. In particular, *multi-label*

problems still demand more work, but due to the lack of available resources (mainly collections) progress in this area is slower compared to other evolving research domains. Furthermore, multi-label assignment should not be studied simply as a more general multi-class problem (which, itself is a more general problem than the binary case), but also as a special case with additional requirements. In multi-label assignment we cannot oversee the fact that some classes are inter-related, that usually the imbalance degree is radically different among classes, that some classes may vary their relevance in the assignment (some classes may be more important for a document than the others) and that, in performance aspects, the need of comparing a sample to every classifier is a waste of resources.

Summarizing, our architecture will define a multi-label document classifier based on binary classifiers. The wide range of choices at each component will drive our experiments in finding a performing system.

3.1 Notation and definitions

In this section a formal definition of the elements involved in a text categorization process is given. This terminology and notation will serve us as basis for describing some aspects of these systems, and which are the key components in them. Unfourtunately, formal models are and on-going discussion in IR research, as shown by the contributions in the SIGIR workshop *Formal Models for Information Retrieval* [34]. Since the retrieval of documents and its automatic manipulation presents a vaste range of applications, the abscence of an agreement in mathematical notation is actually surprising. Some attempts have been made, for example, to stablish notations in a certain subject, like for *key word assignment*, which is, indeed, a Text Categorization matter (see [85]).

We have revised some of the notations and adapted them to the purpose of this text, in the aim of clarifying the text categorization task and setting a base terminology for further discussion. Therefore, the notation and conventions exposed in this section are the result of some consolidation of the most used formalisms we can find in TC literature.

Given

- a collection of documents $X = \mathbf{x_1}, ..., \mathbf{x_i}, ..., \mathbf{x_N}$, and
- a set of classes $C = c_1, ..., c_j, ..., c_M$

we define that

- A *binary classifier* is a function $\Phi_b : X \rightarrow \{true, false\}$ that maps documents to assignment assessment.

- A *m-class classifier* is a function $\Phi_{mc} : X \rightarrow C$ that maps documents to classes.
- A *m-label classifier* is a function $\Phi_{ml} : X \rightarrow 2^C$ where 2^C is the *power set of* C such that $2^C \subseteq C$

Therefore, a *binary classifier* assigns a *true* or *false* to each document; a *m-class classifier* assigns a class from the set of classes to a document; and a *m-label classifier* assigns a subset of the set of classes to a document, that is, it assigns more than just on class to a document.

These are the functions we want to find. In our case, the last one mainly concerns the present work. But both the multi-class (*m-class)* and the multi-label (*m-label)* classifiers can be built from binary classifiers if those classifiers are approximated by a function as follows:

$$\hat{\Phi}_{b-real} : X \rightarrow \Re \tag{3.1}$$

and then, we return *true* if $\hat{\Phi}(\mathbf{x}) > \delta$, where $\delta \in \Re$ is known as the *decision threshold*. The $\hat{\Phi}$ function will allow us to approximate any of the functions Φ_b, Φ_{mc} and Φ_{ml} defined above by applying any of the following strategies:

- $\hat{\Phi}_b$ can be obtained using the function given at equation 3.1 as follows:

$$\hat{\Phi}_b(\mathbf{x}) = \begin{cases} true & \text{if } \hat{\Phi}_{b-real}(\mathbf{x}) > \delta \\ false & \text{otherwise} \end{cases} \tag{3.2}$$

- $\hat{\Phi}_{mc}$ would be the function that returns the class with the maximum classification value. If we define a classifier $\hat{\Phi}^{c_i}_{b-real}$ for each class c_i, then:

$$\hat{\Phi}_{mc}(\mathbf{x}) = \arg\max_{c_i} \{\hat{\Phi}^{c_i}_{b-real}(\mathbf{x}), \forall c_i \in C\} \tag{3.3}$$

- $\hat{\Phi}_{ml}$ must return a set of classes, so the formula could be:

$$\hat{\Phi}_{ml}(\mathbf{x}) = \{c_i \in C : \hat{\Phi}^{c_i}_{b-real}(x) > \delta, \forall c_i \in C\} \tag{3.4}$$

  In this case we could propose another variant: rank classes according to the classification value and then return the l top ranked ones. Furthermore, we may consider even to define a different δ_i for each class.

The main task would be to determine the base classifiers $\hat{\Phi}^{c_i}_{b-real}$ from which we construct a multi-label classifier $\hat{\Phi}_{ml}$ that will be the approximation of the target classifier Φ_{ml}.

This formalism defines the basic paradigm for the current work. As we described in previous chapters, obtaining the base classifiers $\hat{\Phi}^{c_i}_{b-real}$ is achieved

by means of a combination of information retrieval and machine learning techniques: IR for converting documents into weighted lists of attributes and ML for training the base binary classifiers.

There are more notations and proposals for describing this paradigm formally. Allwein, Schapire and Singer propose a common model for handling any multi-class problem using binary classifiers [7]. Their model is more general than ours for the multi-class case, since they propose also a common model for describing both one-against-one and one-against-all learning approaches. But in our case, one-against-one is a very difficult approach, since our classes could appear, in principle, along with any other class. Therefore, we keep the given proposal, even if the one-against-all learning strategy is not explicitly involved in the model.

To conclude, we define an *multi-label classification system* as the algebraic structure:

$$\langle X, C, \Phi_{ml} \rangle \tag{3.5}$$

where

X	collection of documents
C	set of classes
Φ_{ml}	multi-label assignment function

And we already have seen how this last function can be approximated by using real-value based binary classifiers (one per class) (as defined in 3.4). We could propose other variants for this approximation, like the use of boosting algorithms [110] or other voting algorithms [12]. But there is extensive research on those subjects and, even when selecting a different algorithm could lead to some improvement in performance, we are more interested in other topics, like the integration of different sources of data for defining the attributes of a document sample.

The algebraic system defined now is based on the one proposed in [85]. A *multi-label classifier* is represented by the tuple:

$$\langle W, D, \rho \rangle \tag{3.6}$$

where

W	is a set of classes
D	is a set of documents
ρ	is the assignment function. This function is a mapping $\rho : D \to 2^W; d_i \mapsto \rho(d_i) = \rho_i \in 2^W$ That is, ρ takes a document as argument and produces a set of key words belonging to the controlled vocabulary (thesaurus) W.

Note that the thesaurus has been simplified by omitting relations like gen-

eralization, synonymy, etc. Here we consider the thesaurus nothing but a list of controlled terms. Therefore, we may consider it a *controlled vocabulary* rather than a real thesaurus. However, the model could be extended to exhibit those relationships, but with the given description we can already modelize classical models as described briefly in next section. We keep calling it "thesaurus", because in many situations, such information derives from existing ones.

So, the question is how to define the ρ function. As we will see in the following sections, supervised learning based methods are the most common techniques for guessing this function. In fact, many of the methods described work in the way of finding a function that fits with the results, but instead of applying analytical methods, numerical algorithms are designed with this goal.

The machine learning algorithms used will be based on documents represented on a feature space, that is, text will be transformed using linguistic and statistical methods into a list of weighted attributes as follows:

$$F = \{f_1, f_2, ..., f_k, ..., f_T\} \tag{3.7}$$

Where F is the set of all possible features in the corpus. In other words, T is the dimensionality for document vectors. Of course, this may generate very sparse vectors that are better implemented using lists, but from the theoretical point of view the vector space model will ease the understanding of further processing.

To represent the values of features in a document $\mathbf{x_i}$ we will use the tuple:

$$\mathbf{x_i} = (w_{i,1}, w_{i,2}, \cdots, w_{i,k}, \cdots, w_{i,l}) \tag{3.8}$$

where $w_{i,k}$ is the value (weight) of feature f_k in document $\mathbf{x_i}$.

3.2 Architecture for a text categorization system

There are several variants to the scheme we will show here, but they are minor deviations from our common design. A text categorization system may be built up from the following components (see figure 3.2):

1. **Feature extraction**. Text documents in plain format (ASCII) must be processed and indexed. Two steps are given:

   (a) *Feature identification*. We have to identify which are the features to keep: words, bigrams, stemmed terms, entities or any other item that we belive should be considered as a representative component of the original document. This phase usually encompasses techniques

related to natural language processing and statistical information retrieval.

(b) *Feature weighting*. Once features are selected, we have to calculate a weight which measures how relevant they are to the document. Again, many possible approaches are available.

The *bag-of-words* approach is the most common, since it simplifies the processing in a traditional IR way.

2. **Dimensionality reduction** (DR). Information retrieval in general and text categorization in particular suffer from a very high number of distinctive features (hundreds of thousands or even more), due to the fact that each word present in the text is a potential feature. Even though stemming and other lemmatization techniques reduce this number, it has been always an important drawback for applying Machine Learning (ML) techniques to TC. Therefore, some methods have been proposed in order to reduce the dimensionality of the set of features based on a ranking of the ability for a term to be a good indicator for both documents and classes. Thus, measures like *mutual information*, *information gain*, *Chi square* or the simpler *document frequency* (see [130]) can provide useful information for discarding terms. But not only by discarding (also know as *term selection* we can decrease the dimensionality of the feature space, another approach not incompatible with previous one is *feature transformation* (also known as *term extraction*). In this case, we try to replace our terms by other entities that may enclose the existing stems or words, grouping them into semantic sets, where groups of terms point to a unique global concept for each set. For that, clustering techniques (like the supervised clustering method proposed by McCallum [79]) and the *latent semantic indexing* ([11, 31, 71]) are solutions in this direction, as discussed later in this chapter.

   Both, feature extraction and dimensionality reduction, will transform text documents to samples that can be handled by later learning algorithms.

3. **Classifier training**. This is the core of the system, here we use known machine learning methods [81] to build up an autonomous classifier by using supervised learning algorithms. Due to the wide range of classification approaches (statistical, probabilistic, neural networks, fuzzy logic, and so on) proved in thousands of problems (mainly *pattern recognition* and all its variants) we cannot avoid to feel overwhelmed by the vast range of algorithms. Fortunately, plenty of research has been done comparing different classifiers for text categorization (see [67, 129] , so the right choice may lie in few possible approaches like Naïve Bayes, Support Vector Machines [52], LVQ [77], Boosting [110] or perceptrons like PLAUM [127]. As documents have been converted in former stages into list of features, these algorithms can work as they would do on any other type of data different from text.

4. **Thresholding**. If we need "hard" classification (*yes/no* answers) and our classifier outputs a value of matching between a document and a class, we can decide whether to assign the document to the category by applying a threshold to the value returned by the classifier. In approaches like the *S-Cut*, the *R-Cut* or the *P-Cut* ([128]) the threshold is a fixed value used as decision boundary. Another approach is to apply the limit not on the *classification status value* (that is the reason to also call the previous cut-based approaches *CVS thresholding*), but rather on the number of classes assigned to a document. In this case, a fixed number of classes will be attached to each document ([18, 86, 128]).

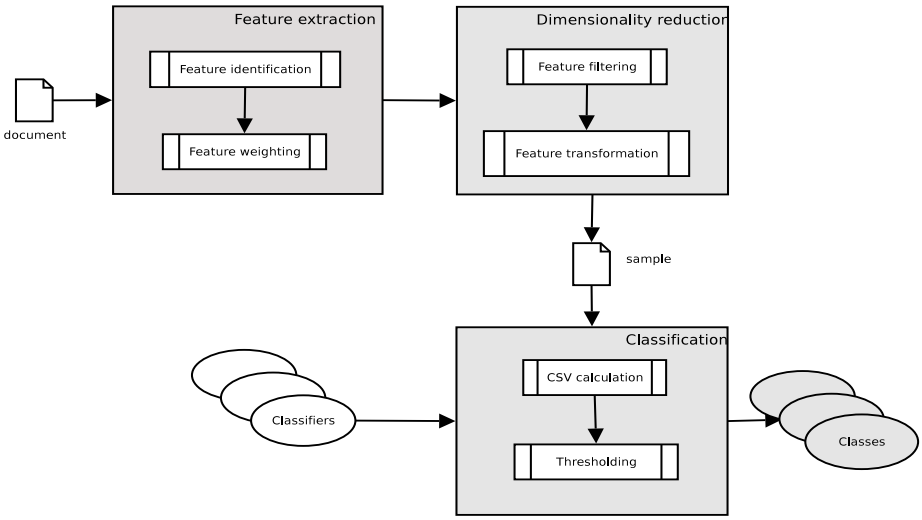


Figure 3.2: Architecture for a text classification system.

Summarizing, we have to convert the document into a set of features that can be handled by the classification algorithm. Since we are in a multi-label case, we will use the one-against-all approach, therefore a classifier per class will be, in principle, available. As result a list of pairs $<class, CSV>$ will be thresholded to obtain the final labels for the document.

The learning process will depend on the algorithms used as building blocks of the proposed architecture. In general, a training collection split into learning and validation sets is used. For HEP many documents are available but the size of the controlled vocabulary while labels are selected and the distribution of them across the collection raises additional problems that have been faced, discussed and solved partially in the present work. The algorithms involved may differ from one system to another, but we believe that, in general, this is a very valid and common architecture for most text classification problems. We can easily identify the given blocks, though the complexity of them and the techniques used for feature identification, selection, filtering and transformation

may differ radically from one system to another (or not even be present at all).

Chapter 4

Feature extraction

A text is a raw sequence of symbols that may appear under different formats in digital environments, some of them are mainly for document distribution, like *Portable Document Format* (PDF) and *PostScript* format (PS), where others are determined by the text processing tool used in its writing like Microsoft Word format (DOC), OpenOffice.org format (SXW), LaTeX and many others. In general, we will convert from all these formats into plain-text ones, with encodings like ASCII or Unicode. The main goal of feature extraction is to transform a document from one of these formats into a list of items (the *features*) easier to be processed by machine learning algorithms.

4.1 Feature identification

A feature is any item that can be considered an attribute of a sample by a classifier. It is important to identify which items will be features for the document. Human language complexity turns our identification into a very difficult task: a word can represent different concepts due to language ambiguity [109, 117], concepts may be represented by pairs of words or even larger combinations (*word n-grams*), some words are meaningless, depending of the *part-of-speech* of a word we may consider it in a different manner, and so on. To overcome these obstacles plenty of algorithms and techniques have been developed during the last years (as we can see, in the review of indexing schemes by Arampatzis [9]). It is important to note here that methods described have as goal to determine which parts of the text are likely to be features, not to decided if those parts should be considered as final features or not, since the goodness of a feature and their filtering are part of the *dimensionality reduction* phase (also known as *feature selection*).

Most common schemes for feature identification are:

- *Bag of words*. We enclose here all those schemes that consider each word as a potential feature with no further inter-dependency. This approach uses to be combined with stemming or lemmatization algorithms. Here, the dimensionality reduction becomes very important due to the large set of resulting features.

- *Word n-grams*. Also know as *multi-word recognition*. Short sequences of words are also considered to be potential features, e.g. *Cherenkov radiation* or *Big bang*.

- *Summarization based*. One of the most sophisticated approaches for feature identification consists in applying summarization techniques to identify as features relevant fragments of text that best comprise the content topics.

A more detailed review of each of these frameworks is provided, emphasizing those techniques applied on our system. Of course, some other approaches could have been mentioned here. Anyhow the schemes above are considered not only the more frequent but defined in an increasing level of complexity. At each scheme, more complex linguistic techniques are involved.

4.1.1 Bag of words (BOW)

This is, maybe, the simplest approach we can find for feature identification. Here, every word can be considered as a candidate to be one attribute of the final document representation. This approach considers the minimal linguistic processing and it has been described as the *fully statistical approach*: a document is considered as a unordered set of words, only their frequency is taken as attribute for them. Anyhow, nowadays, there is no single approach that should be called fully statistical or fully linguistic, since almost any system applies both philosophies to a certain extent. For example, traditionally in the BOW approach, words that are too common in the language are filtered out since they tend to be meaningless terms: determiners, prepositions, auxiliaries, and so on. These words are known as *stop words*. In the English language, the probability of encountering the rth most common word is given roughly by $P(r) = \frac{0.1}{r}$ for r up to 1000 or so, this is the so called *Zipf's Law* [91]. Words like "the", "of" and "to" appear so often in a text that we can discard them along with many very frequent terms. The BOW approach tends to safeguard those words that are more relevant for the semantic content of the document, so effective weighting for them is important [23]. Also some form of lemmatization or stemming [49] is applied, as a cheap method for using word base forms rather than word inflexions as representative features. Thus, we profit from some linguistics characteristics to enhance the effectiveness of the BOW model. We can see and example of these text processing techniques. If we had the following text:

```
New experimental results on the
production of $\phi$ and $f_2'(1525)$
mesons in the annihilation of stopped
antiprotons are discussed. The explanation
of these facts in the framework of the
polarized  strangeness model is considered.
```

After applying stop-words removal and Porter's stemming algorithm [93] we would end up with the following set of *features*:

```
new experimental result
production    $\phi$     $f_2'(1525)$
meson         annihilation    stop
antiproton      discuss        explanation
         fact         framework
polarize   strangeness model    consider
```

4.1.2 Multi-word recognition

Another approach is to make one step forward: some ordering information is considered between certain words whenever they could be potentially representing a unique concept in the form of a fixed sequence, that is, the features will not only be just words alone, but also certain detected word *n-grams* [25, 55]. For example, pairs "top quark" or "Higgs boson" are referring to particle names. Some good results are achieved by selecting candidate multi-words using the *Part Of the Speech* (POS) information, i.e. compositions of words according to certain POS patterns (as a *verb*, as a *noun* plus as an *adjective*, and so on) are identified and selected if a certain co-occurrence index (like Mutual Information) is higher than a given threshold. In our case the patterns to be used will be the following pairs:

- *noun-noun* and
- *adjective-noun*

These two possible combinations of words will be detected after a POS tagging parsing and then marked as multi-words candidates. In table 4.1 some examples of possible pairs are given with their selection decision as candidate depending on the pattern identified[1].

Once we have selected the candidates they are ranked by their *mutual information* value, computed as follows:

$$MI(x,y) = log_2 \frac{p(x,y)}{p(x)p(y)} \tag{4.1}$$

[1]Some pairs may be presented as tri-grams, rather than expected bigrams due to already existing pairs of words joined by hyphenation; e.g. the *the-low-energy* entry

bigram	POS tags	decision
offer-some	VB-DT	INVALID
potential-advantages	JJ-NNS	VALID
in-the	IN-DT	INVALID
the-low-energy	DT-NN	INVALID
low-background-experiments	NN-NNS	VALID
a-500 kg	DT-SYM	INVALID
detector-to	NN-TO	INVALID
be-placed	VB-VBN	INVALID
placed-near	VBN-IN	INVALID
core-of	NN-IN	INVALID
nuclear-power	NP-NP	VALID
power-station	NP-NP	VALID

Table 4.1: Some bigrams candidates with their *part of speech (POS)* tags and the decision taken.

where

$p(x, y)$	is the probability for terms x and y to appear together. This can be computed as the frequency of both terms as sequence in the whole collection divided by the total number of possible pairs: $p(x, y) = freq(x, y)/N_{pairs}$
$p(x)$ and $p(y)$	are the probabilities of terms x and y respectively. This will be calculated as the frequency of each term over the total number of terms (not unique terms) in the collection: $p(x) = freq(x)/N_{terms}$; $p(y) = freq(y)/N_{terms}$
N_{terms}	is the number of terms (not unique terms, all occurrences) present in the corpus

Let $\{a, b, c\}$ be the set of terms in the whole collection, and let the sequence "*abc*" represents the macro-text composed by the joint of all the documents in the collection. Then, the number of possible pairs is 2, since we can only form *ab* or *bc* as candidates. Thus, the number of pairs N_{pairs} is exactly $N_{terms} - 1$, and due to the high number of terms in any text collection we can assume that $N_{pairs} \simeq N_{terms}$. Thus, we can rewrite equation 4.1 as:

$$MI(x, y) = log_2 \frac{N_{terms} freq(x, y)}{freq(x) freq(y)} \tag{4.2}$$

As N_{terms} is constant and the MI value is destined to be a comparison index, we can even remove it from the given equation. One interesting result of applying this formula is that pairs of seldom used terms get a high MI value. Thus, bigrams with very low frequency are ranked in the top. A common sense action is to discard bigrams appearing just once. In that case, table 4.1 shows the 15 top bigrams generated from the abstracts of the *hep-ex* partition (see chapter 11 for details about this corpus).

MI(x,y)	freq(x,y)	x - y
17.137336409474	2	shunt - impedance
17.137336409474	2	nu-mu/nu-e - ratio-of-ratios
17.137336409474	2	low-noise - viking
17.137336409474	2	gross-llewellyn - smith
17.137336409474	2	g.l. - kane
16.5523739087528	2	roman - pot
16.5523739087528	2	orthopositronium - decay-rate
16.5523739087528	2	blue - leds
16.5523739087528	2	bess - rigidity
16.137336409474	4	paul - scherrer
16.137336409474	4	palo - verde
16.137336409474	3	rectangular - bars
16.137336409474	2	unix - workstations
16.137336409474	2	helium - bags

Figure 4.1: Some bigrams with their mutual information value $MI(x, y)$ and their frequencies.

Some experiments have been carried out to check if the use of frequent bigrams as unique features can be an enhancement for the system. Results found are described in section 12.3.

4.1.3 Summarization based

Text summarization is the process of distilling the most important information from a source to produce an shorter version for a particular user or task [73]. Summarization techniques are focused in the creation of brief texts that can condense the content of a longer original text. These techniques can be applied as feature selection methods in text categorization. Mainly, the result of the summarization algorithm is a list of key-paragraphs, key-phrases or key-words that have been considered to be the most relevant ones. Although some methods are able to generate new sentences from the content, usually it consists in a pure selection of textual fragments. We can use those fragments as features in the categorization process.

Summarization is, indeed, a complex task itself, since a wide variety of techniques can be applied in order to condense content information, from pure statisticall approaches to those using closer analysis of text structure involving linguistic and heuristic methods (anafora resolution, named entity recognition, lexical chains, and so on). In fact, many algorithms for feature reduction, feature transformation, feature weighting, etc. are directly related to this task, since they already try to select a proper and limited set of items that can be used as storing the core content of a given text. But the aim of summarization techniques is to go one step forward, rearranging this information to produce

readable texts, although this processing is still in a very early stage (i.e. go from *extraction* to *abstraction*).

Most of the working systems are based in the selection of a certain number of sentences found in the text which are considered to express most of the concepts present in the document. Li and others [68] apply a Naïve Bayes classifier to extract those terms appearing both in the abstract of the document and in the body. The probabilistic classifier determines whether the word is relevant or not for the general content of the document and outputs all these words, that will be later used as input features for the next level of classifiers that will produce the final list of classes. Kolcz and others [59] showed that we can profit also from the fact that the text can be reduced to 10%–15% of the original one, so they act as reduction algorithms too.

4.2 Feature weighting

Once we know the set of features that will represent our text, we may want to weight them according to their relative importance for the document in the collection. In the Vector Space Model [108], the proper weighting of a feature can improve the performance of a system. A weighting scheme is composed of three different types of term weighting: local, global, and normalization (see [24] for a nice comparison among different weighting schemes). The weight of a term i in a document j implies generally the calculation of:

$$w_{ij} = L_{ij} G_i N_i \tag{4.3}$$

where

L_{ij} is the *local weight* of the term i in document j. It is usually based on the number of occurrences of the term in the document.

G_i is the *global weight* of the term in the collection of documents. This factor tends to under-weight those terms that are too common in the collection.

N_j is the *normalization factor* for term weights in document j. This factor is used to adjust the vector of the document to its norm, so all the documents have the same modulus and can be compared no matter the size of the text.

Depending on our choice for these three factors, we can compose a wide range of weighting schemes. Depending of the specific properties of our collection, some schemes could be worth applying over others. Some weighting schemes are a combination of probabilistic factors with statistical ones, like the OKAPI weighting scheme [104] which is reporting very good results in classical information retrieval tasks. In this work, two weighting schemes were used: the *TF.IDF* (*term-frequency inverse-document-frequency*) and the *entropy* based one. Both with *cosine normalization*.

4.2.1 TF.IDF weighting

This scheme [106] is one of the most common weighting schemes used. Almost all the other weighting schemes are variants of it.

In the TF.IDF scheme, we use

- the **frequency** of the term in the document as local weight,
- the **inverse document frequency** as global weight, penalizing those terms too frequent through the documents in the collection; and
- the **cosine normalization**, which merely implies dividing by the geometric average.

We can write the definition of the weight of a term as follows:

$$w_{ij} = \underbrace{f_{ij}}_{L_{ij}} \underbrace{\log(N/n_i)}_{G_i} \cdot \underbrace{\frac{1}{\sqrt{\sum_{k=1}^{T}(f_{kj} \cdot \log(N/n_k))^2}}}_{N_j} \tag{4.4}$$

where

w_{ij}	is the weight of term i in document j
f_{ij}	is the frequency of term i in document j
N	is the total number of documents in the collection
$n_i, (n_k)$	is the number of documents in the collection that contain term i, (k)
T	is the number of terms in the collection

4.2.2 Entropy weighting

Another weighting scheme applied in our experiments is the *entropy* based one [71]. Although the TF.IDF weighting scheme is the preferred method in several systems, an alternative is always an interesting subject of study. In this work, a deep analysis of the effect of every variable is intended. Therefore, this second method has been selected as an additional scheme for feature weighting, in the aim of opening experimental results to the study of how sensible a system is to the weighting scheme selected. Of course, for a complete conclusion on this matter, a wider study of available weighting schemes is needed, but entropy has been found to work pretty well as a method, hence its explanation here.

This approach basically proposes a global weight factor which is related to the number of bits we would need to specify that a document contains a given term i. Taken from the *Mathematical Theory of Communication* by Shannon [114], it uses the idea of considering documents, terms and categories as attributes in a message for information exchange.

The formula is:

$$w_{ij} = \underbrace{(1+\log f_{ij})}_{L_{ij}} \underbrace{\left(1 + \underbrace{\frac{1}{\log_2(N)} \sum_{k=1}^{N} \frac{f_{ik}}{F_i} \log_2 \frac{f_{ik}}{F_i}}_{\bar{H}(i)} \right)}_{G_i} \underbrace{\left(\frac{1}{\sqrt{\sum_{k=1}^{T} (L_{kj} \cdot G_k)^2}} \right)}_{N_j} \tag{4.5}$$

where

N	is the total number of documents in the collection
F_i	is the total frequency of term i through the entire collection: $F_i = \sum_{j=1}^{N} f_{ij}$
f_{ij}	is the frequency of term i in document j
$\bar{H}(i)$	is the average uncertainty or entropy of term i in the collection
T	is the number of terms in the collection

4.2.3 Other weighting measures

Since the beginning of the Information Retrieval discipline, plenty of weighting measures have been proposed. Most of them result as combination of different indexes focusing on capturing a certain aspect about the relevancy of the term in the text and its relationship with the collection. Again, we recommend to read Chisholm & Kolda's paper [24], which also includes other measures like Chi-square or log-likehood, these later two applied by Pouliquen and others in their experiences with the EUROVOC thesaurus [18].

Chapter 5

Dimensionality reduction

When working in the term space, the number of dimensions for document vectors obtained as result of considering each language component a potential feature is prohibitive: our current architectures are insufficient to support efficiently hundreds of thousands of features. The *Dimensionality Reduction* phase (DR) is responsible of decreasing such a number. There are two main approaches: to *filter out* those features that we consider have low informational importance, or to transform features from one space into another space projecting them into a lower dimensionality.

One important aspect of applying feature transformation in our system, is that it has been reported to be a way to reduce the negative effect of *over-fitting*. A system is over-fitted when it is good in classifying documents used for training but very bad for new incoming documents. This is due to the fact that the system has been tuned too much in the direction of the current training data, without generalizing any properties for classification. By reducing the number of terms, we are forcing the system to work in lower dimensionality, so only few features are taken into account for finding the proper classes. This will improve our classifiers speed and robustness.

5.1 DR by feature selection

Feature selection is a very complex task, since it has to do with meanings and relative importance of features in a document. The goal of this process is to dig into the document components, and determine those that better represent the document. We expect to get in return those terms or sentences that the content cannot be without. Adam Kilgariff [54] has nice studies on these matters, and he already identifies how hard it is to know the distinctiveness of a very common word versus a rare one. Many methods have been proposed, from simple ones based on easy computations of a measure to rank the excellence of each word,

to complex transformations with intense linguistic processing.

In any case, when dealing with text categorization, two main paradigms can be considered in order to set the frame wherein all these techniques will be applied:

- **Locally**. We reduce the number of features to be taken into account for each category or/and document.
- **Globally**. We reduce the number of features in the whole collection.

In our work, we have considered a global approach, due to the large number of possible classes in the studied corpus.

We will give a brief overview on summarization techniques, since it is the most complex one, and a more extended description of the *information gain* measure, since it has been considered in our experiments as the main feature selection approach.

5.1.1 Summarization for feature selection

The experiment elaborated by Kolcz et al. [59] shows that using summarization techniques has a similar performance as selection techniques like *Mutual Information* (MI).

In their work, seven methods for summarization were used to produce a minimal text which would be used as a reduced version of the original one, performing in this manner both feature selection and dimensionality reduction. The methods for summarization tested on the Reuters-21578 collection were:

- Title of the story
- First paragraph in the story
- Paragraph of the story with most title words
- Paragraph of the story with most key words
- First two paragraphs of the story
- First and last paragraphs
- Best sentence (that one with at least 3 title words and 4 key words)

The classifier was based on *Support Vector Machines*, training a SVM binary classifier (one-against-rest) on each class. To compute the value obtained at *Break Even Point* (BEP), they micro-averaged the results of all the binary SVM classifiers generated. Each method was compared to MI and similar values of precision and recall at the BEP reached. Their results seem promising, but are not considered in our work. This issue remains as a pending topic for further research.

5.1.2 Information Gain

How do we measure information? This has been a undefined matter until the arrival of *Information Theory*, establishing the basis for a mathematical study of information processing five decades ago ([114]). Shannon defined the information contained in a system in terms of *bits*. The information is measured then as the *number of bits needed to represent all possible states of a system*. It is like the number of *yes/no* questions that we would ask in order to fully describe the configuration of a system. For example, if we throw a die and want to represent the state of it, how many bits would we need? The calculation is performed by looking at all possible states as follows:

$$\begin{aligned} I(die) &= (p(face=1), p(face=2), ..., p(face=6)) = && (5.1)\\ &= -\frac{1}{6}log_2\frac{1}{6} - \frac{1}{6}log_2\frac{1}{6} - ... - \frac{1}{6}log_2\frac{1}{6} = && (5.2)\\ &= 2.5\, bits && (5.3) \end{aligned}$$

where

$p(face = x)$ is the probability for the die to show the face with value x

So, with 2.5 bits we can describe any possible state of the system from an informational point of view. In practice we would use 3 bits, of course, but this is more than the minimum needed since with 3 bits we can represent more than just six possible states ($2^3 = 8$ states). The information as we have understood it is usually called *entropy* and is represented by *H(S)*, which means the entropy of a system S. As we can see, the more disordered a given system is (more entropy), the higher number of bits we will need to describe it. If we already had some information, some clue, about the state of the system to describe then we would need a lower number of bits. Knowing the value of an attribute in the system will decrease the entropy and, as we can imagine, there are attributes which provide more information than others. Thus, it would be helpful to define a value to explore it. This is called *information gain*, and it is crucial in the construction of some classifiers like the C4.5 decision tree ([81]).

The information gain for a set S knowing the value of a certain attribute A is defined as follows:

$$Gain(S, A) \equiv H(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} H(S_v) \qquad (5.4)$$

where

S_v is the set of elements whose value for attribute A is equal to v
$H(x)$ is the entropy of set x

The entropy of a set S is computed taking the probabilities of every element in the set as follows:

$$H(S) = -\sum_{i=1}^{n} p_i \log_2 p_i \tag{5.5}$$

Computing the information gain of a term represents computing a measure of the discriminative power of the term for classes, i.e. how a given term can reduce the entropy of classes in the collection. It has been used mainly for constructing *decision trees* [81], but its use for filtering features has also reported good results [21].

The information gain is computed for each term of the training set, and the terms whose information gain is less than some predetermined threshold, or those which a rank position by information gain less than a given number, are removed.

When working in multi-label assignment, we could think on the information gain in a *per class* basis: compute the information gained by knowing that a given class is assigned to the document. This consequent extension of the information gain value is deployed below and will be useful for use later on.

Now, we want to place equations 5.4 and 5.5 into the domain of our problem. Since we are in a problem of text categorization, we want to calculate which is the *gain of information to predict a category* i *knowing that a given term* t appears in the document to be categorized.

$$\begin{aligned}
Gain(t, C) &= H(C) - \sum_{x \in \{t, \bar{t}\}} \frac{|C_x|}{|C|} H(C_x) = \\
&= H(C) - \left(\frac{|C_t|}{|C|} H(C_t) + \frac{|C_{\bar{t}}|}{|C|} H(C_{\bar{t}})\right) = \\
&= -\sum_{i=1}^{M} p(c_i) \log_2 p(c_i) + \\
&+ p(t) \sum_{i=1}^{M} p(c_i|t) \log_2 p(c_i|t) + \\
&+ p(\bar{t}) \sum_{i=1}^{M} p(c_i|\bar{t}) \log_2 p(c_i|\bar{t})
\end{aligned} \tag{5.6}$$

where

M	is the total number of categories: $\|C\| = M$
C	is the set of all possible categories
$\{t, \bar{t}\}$	are the considered two possible values of term t, i.e. t is present or absent
C_t	is the set of categories attached to documents containing term t
$C_{\bar{t}}$	is the set of categories attached to documents not containing term t
$p(c_i)$	is the probability of class i

We can rewrite this equation using frequencies of terms and categories among the documents in the collection:

$$\begin{aligned} Gain(t, C) &= -\sum_{i=1}^{M} \frac{n_i}{N} \log_2 \frac{n_i}{N} + \\ &+ \frac{n_t}{N} \sum_{i=1}^{M} \frac{n_{it}}{n_t} \log_2 \frac{n_{it}}{n_t} + \\ &+ \frac{N - n_t}{N} \sum_{i=1}^{M} \frac{n_i - n_{it}}{N - n_t} \log_2 \frac{n_i - n_{it}}{N - n_t} \end{aligned}$$

where

N	is the total number of documents in the collection
n_i	is the number of documents labeled with category c_i
n_t	is the number of documents containing term t
n_{it}	is the number of documents labeled with category c_i and containing term t

Thus, we have set the *information gain of a given term t in relation to class distribution*. This is useful, for example, to filter terms in the feature reduction phase. But we can go further with the information gain value. Since we face a multi-label problem, we can establish a information gain *per class* in a basis of classes themselves, i.e. we can measure how a given class helps us in predicting more classes. This reasoning could not be achieved outside the multi-label problem, since classes would be mutually disjoint. The *gain of information to predict a category knowing that the document is labeled with a given class* c would be:

$$
\begin{aligned}
Gain(c, C) &= H(C) - \sum_{x \in \{c, \bar{c}\}} \frac{|C_x|}{|C|} H(C_x) = \\
&= H(C) - (\frac{|C_c|}{|C|} H(C_c) + \frac{|C_{\bar{c}}|}{|C|} H(C_{\bar{c}})) = \\
&= -\sum_{i=1}^{M} p(c_i) \log_2 p(c_i) + \\
&+ p(c) \sum_{i=1}^{M} p(c_i|c) \log_2 p(c_i|c) + \\
&+ p(\bar{c}) \sum_{i=1}^{M} p(c_i|\bar{c}) \log_2 p(c_i|\bar{c}) \qquad (5.7)
\end{aligned}
$$

where

M	is the total number of categories: $\|C\| = M$
C	is the set of all possible categories
$\{c, \bar{c}\}$	are the considered two possible values of the class c, i.e. document is labeled with c or not
C_c	is the set of categories attached to documents labeled with class c
$C_{\bar{c}}$	is the set of categories attached to documents not labeled with class c
$p(c_i)$	is the probability of class i

We can, again, rewrite this equation using frequencies of categories among the documents in the collection:

$$
\begin{aligned}
Gain(c, C) &= -\sum_{i=1}^{M} \frac{n_i}{N} \log_2 \frac{n_i}{N} + \\
&+ \frac{n_c}{N} \sum_{i=1}^{M} \frac{n_{ic}}{n_c} \log_2 \frac{n_{ic}}{n_c} + \\
&+ \frac{N - n_c}{N} \sum_{i=1}^{M} \frac{n_i - n_{ic}}{N - n_c} \log_2 \frac{n_i - n_{ic}}{N - n_c}
\end{aligned}
$$

where

N	is the total number of documents in the collection
n_i	is the number of documents labeled with category c_i
n_c	is the number of documents labeled with category c
n_{ic}	is the number of documents labeled with categories c_i and c

We have carried out some experiments to state the goodness of this filter (see section 12.5), where clearly a strong reduction in the number of features had a tiny influence on the performance of the multi-label classifier [88].

5.1.3 Other techniques

There are plenty of factors that can be used to discard features with low score. A very common one is the *document frequency*. This factor is used to remove those features that either appear in too many documents or in too few of them, i.e. if a term is very rare or too frequent, we discard it. Due to its simplicity, almost any system implements it, since only by removing those features appearing in one document we may reduce considerably the feature space (mainly due to aberrations produced by document conversions to plain text files). On the other hand, by removing too frequent terms we are applying a similar philosophy as stated by the use of a *stop words* list.

Other measures like the *DIA association factor*, *Chi-square* (χ), *NGL coefficient*, *Mutual information*, *Odds ratio*, *Relevancy score*, *GSS coefficient*, have been implement in various systems, but since the information gain factor has shown similar or better behaviour compared to them, we have selected IG and document frequency as our basic filters. Extensive bibliography about all these methods can be, again, found in Sebastiani's overview [112].

5.2 DR by feature transformation

The number of features we are dealing with can also be reduced if we map them to another feature space of lower dimensionality. These techniques are often called *conceptual indexing* [83], since the idea is to related terms with semantic sets so we will manipulate these "semantic entities" or "concepts" as they were now the feature space to be applied onto our classifiers. To be exhaustive, we must say that this is not always a "map" according to its mathematical definition, due to the fact that some terms may point to more than just one concept. In fact, our classification goal can be viewed exactly like that: a transformation from terms to classes and, therefore, a reduction in the dimensionality of the document, i.e. we want that our set of features F is replaced by a target set F', so that $|F'| \ll |F|$.

Summarizing, our object is to reduce dimensionality by changing the feature space. This transformation may be many-to-one function or even a many-to-many relation, the only condition is to have our number of dimensions reduced effectively without losing too much in semantic content.

We have not implemented in our system any of these techniques. Despite the benefits of dimensionality reduction by feature transformation, applying IG filters and document frequency ones were enough for the classifiers selected in

later phases to work properly. Anyhow, for educational purposes, an overview of the two main tendencies is given here.

5.2.1 Term clustering

By using *term clustering* techniques we try to group terms sharing common properties. In fact, the aim is to group all those terms referring to the same concept or, at least, related to the same semantic space. Therefore, when certain terms are strongly interrelated due to their semantic proximity, we will use the cluster, instead of the terms, as feature. Now, the question is how we determine the semantic relationship between two terms. It is important to note that similarity measure must be towards semantic information, since we could produce clusters of terms based on other properties that may produce worse results than using just simple terms instead of clusters as found by Lewis [64].

Although there are more positive studies in this sense [69], it is clear that best approaches are in the direction of supervised algorithms, i.e., use the known categories associated with the documents to determine term clusters.

The benefits of using term clustering are better described by Baker [10] when applying a distributional clustering in text categorization tasks: useful semantic term clustering, higher classification accuracy and, consequently, smaller classification models. The classification algorithm was based on the Naive Bayes model and it outperformed other feature reduction algorithms like information gain of mutual information.

Bekkerman and others [14] compared the former approach to the use of simple words. Based both on Support Vector Machines classification algorithms, they found that it is worth using this distributional clustering on terms rather than the *bag of words* model only when the complexity of the language used is high, i.e. when the vocabulary is too general. The conclusions provided by this last work suggested not to apply it on our collection, since we are dealing in our case with a very specialized vocabulary and, following such conclusions, no big improvement in performance should be expected.

5.2.2 Latent Semantic Indexing

One of the better known approaches in feature transformation is the *Latent Semantic Indexing* (LSI), proposed by Deerwester and others [31]. Correlations among words according to their usage in documents is the key concept underneath this mathematical machinery. LSI is based on the assumption that there is some underlying or latent structure in the pattern of word usage across documents, and that statistical techniques can be used to estimate this structure. LSI uses singular-value decomposition (SVD), a technique based on matrix operations related to eigenvector decomposition and factor analysis. We see now how SVD works.

Assume that we have a $M \times N$ word-by-document matrix $\mathbf{A}$, where M is the number of words and N the number of documents. The singular value decomposition of $\mathbf{A}$ is given by:

$$\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^{\mathbf{T}} \tag{5.8}$$

where $\mathbf{U}(M \times R)$ and $\mathbf{V}(R \times N)$ have orthonormal columns and $\mathbf{\Sigma}(\mathbf{R} \times \mathbf{R})$ is the diagonal matrix of singular values. $R \leq min(M, N)$ is the rank of $\mathbf{A}$. If singular values of $\mathbf{\Sigma}$ are ordered by size, the K largest may be kept and the remaining smaller ones set to zero. The product of the resulting matrices is a matrix $\mathbf{A_K}$ which is an approximation to $\mathbf{A}$ with rank K.

$$\mathbf{A_K} = \mathbf{U_K}\mathbf{\Sigma_K}\mathbf{V_K^T} \tag{5.9}$$

where $\mathbf{\Sigma_K}(K \times K)$ is obtained by deleting the zero rows and columns of $\mathbf{\Sigma}$, and $\mathbf{U_K}(M \times K)$ and $\mathbf{V_K}(N \times K)$ are obtained by deleting the corresponding rows and columns of $\mathbf{U}$ and $\mathbf{V}$.

$\mathbf{A_K}$ in one sense captures most of the underlying structure in $\mathbf{A}$, yet at the same time removes the noise or variability in word usage. Since the number of dimensions K is much smaller than the number of unique words M, minor differences in terminology will be ignored. Words which occur in similar documents may be near each other in the K-dimensional space even if they never co-occur in the same document. Moreover, documents that do not share any words with each other, may turn out to be similar.

Some works underline the robust theoretical base of the LSI [11, 71] and its benefits in information retrieval tasks. But one of the main drawbacks against this space transformation is the computational cost involved in getting the singular value decomposition. When dealing with big corpora like ours, the numerical packages implementing solvers for the SVD demand a long computation time and large memory resources, and including new documents may need to recompute the whole decomposition. All these inconveniences made us discard this technique from the final system, although its promising future may bring us brilliant results in a short time.

Chapter 6

Classifiers

Up to this point, we have merely converted documents into a list of attributes (features) that we believe condense the content of the original text. Now that documents are represented in a format that can be very well managed by learning algorithms, classifiers can be trained. Here is when we apply Machine Learning techniques. It is important to note that all these chapters consider a machine learning approach to text classification, therefore the classifiers built following the presented architecture need to be trained on a corpus of documents processed and converted into the feature space from plain text format.

Machine learning research has produce a very wide range of supervised algorithms to train binary classifiers. This number of possibilities is so huge that a detailed description of them cannot be given here. We will only describe those our system is based on and briefly comment some remaining ones.

We recall here the equation 3.2, that defined the binary classifier as a function $\hat{\Phi}_{b-real} : X \rightarrow \Re$. Following our notation given earlier we can describe three different elements:

1. **Documents**. The collection of documents is a set $X = x_1, ..., \mathbf{x_i}, ..., x_N$

2. **Classes**. The set of possible classes for a given document is $C = c_1, ..., c_j, ..., c_M$. There is a relation Φ that determines which classes are associated to which documents. This is the function that we want to approximate by $\hat{\Phi}_{b_real}$.

3. **Features**. Each document $\mathbf{x_i}$ is considered a vector of real values with a dimension for every feature in the feature space $F = \{f_1, f_2, ..., f_k, ..., f_T\}$.

As we can see, features are related to documents and documents are related to classes. We will see in the next section how a multi-label classifier can be built up from binary learners.

6.1 Classification using binary classifiers

As pointed out in the introduction section of this chapter, we can reduce any multi-label or multi-class classification to binary problems, where a document is classified as either relevant or not relevant with respect to a predefined topic or class. There are two main straightforward approaches to combine binary decisions to produce a multi-label one:

- *By thresholding*. Each binary classifier will produce a *classification status value* (CSV). The final set of automatically assigned classes are those with a CSV over a predefined threshold, i.e. for each classifier, we know if the class will be assigned to the document or not. This approach has the benefit of considering each classifier independent from the rest, therefore we can decide whether to return a label or not as soon as each classifier finishes its computation. Distributing the described processes feasibly and the number of resulting classes for a document is only tuned by adjusting the threshold, but might certainly produce a different number of classes for each document, desirable in many cases. This threshold can be *global* or *local*, i.e. we may have a unique threshold over all the classifiers or a different one per class. A global threshold makes sense only when CSVs are comparable, and that is something that is not always so easy to assert: margin based values are not good indicators of the proximity of a document to a class and, moreover, we may have a different classification algorithm per class. All these problems are not present when using local thresholds.

- *By Ranking*. If we rank all the resulting CSVs and then select only the top N classifiers we can control precisely the number of classes assigned to a document, but we have to wait for all the classifiers to finish to compute the final assignment. Again, comparable CSVs are needed, so the same imposed restrictions found in global thresholding are present here.

Allwein and others [7] propose a clean method to reduce any multi-class problem into multiple binary classification tasks. Even when they focus on margin based classifiers, we can without much effort apply this method to any binary classifier. Based on what they call the *coding matrix* $\mathbf{M}$, a map between documents and classes can be established in two different ways:

- One-against-all: $\mathbf{M}$ is a $m \times m$ matrix where

$$\mathbf{M}(i,j) = \begin{cases} +1 \text{ if } i = j \\ -1 \text{ if } i \neq j \end{cases} \tag{6.1}$$

$$\begin{pmatrix} +1 & -1 & -1 & \cdots & -1 \\ -1 & +1 & -1 & \cdots & -1 \\ -1 & -1 & +1 & \cdots & -1 \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ -1 & -1 & -1 & \cdots & +1 \end{pmatrix}$$

In this way, we train each binary classifier considering as positive sample all those documents assigned to the class, and negative samples all the rest of documents. This has the disadvantage of having maybe many positive samples compared to negative ones (as noticed by our study in experiment 12.7).

- All pairs approach: In this case, $\mathbf{M} \in \{-1, 0, +1\}^{k \times \binom{k}{2}}$ in which each column corresponds to a distinct pair (r_1, r_2) where $\mathbf{M}$ has $+1$ in row r_1, -1 in row r_2 and zeros in all other rows.

$$\begin{pmatrix} +1 & -1 & 0 & \cdots & 0 & 0 \\ -1 & +1 & 0 & \cdots & 0 & 0 \\ 0 & +1 & -1 & \cdots & 0 & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & 0 & \cdots & -1 & +1 \end{pmatrix}$$

Now, each classifier is trained over two *disjoint* classes, and the output of the classifier will always be a class. The main drawback for this second encoding is that it is not applicable on multi-label problems, since in principle any two classes may appear together as labels for the same document. On the other hand, when managing with multi-class collections, this is a very good approach, since classifiers over disjoint classes seem to work better than the one-against all approach, even when classification status values among classes may not be comparable (and that is, again, another reason to discard this method for multi-labeling classifications).

Using any of the two given definitions, we train k classifiers and each classifier $\hat{\Phi}_s$ is trained with all pairs $(\mathbf{x_i}, \mathbf{M}(y_i, s)$ (they called this *multi-call* training), where y_i is the class associated with document $\mathbf{x_i}$ ($y_i \in C$). For example, for class c_1 we train $\hat{\Phi}_1$ with pairs $\{(x_1, \mathbf{M}(y_1, 1)), (x_2, \mathbf{M}(y_2, 1)), \cdots, (x_n, \mathbf{M}(y_n, 1))\}$. We could train just one classifier $\hat{\Phi}$ extending the domain with an additional dimension carrying the class: $\hat{\Phi}_s(\mathbf{x_i}) \equiv \hat{\Phi}(\mathbf{x_i}, s)$ (what they called *single-call*). Notice that in the one-against-all we have one classifier per class, while in all-pairs the number of trained classifiers is $\binom{k}{2}$, which can be a prohibitive number of classifiers.

As we can see, what the coding matrix indeed does is to recode document labels into two possible choices: $+1$ or -1. But besides this technique for classifier training, they propose also a way to "decode" the result of the classifiers to select the final class. Since for each column at $\mathbf{M}$ a binary classifier is trained

with documents labeled as the coding matrix states, we will produce a vector of predictions (a vector of CSVs) $\hat{\phi}(\mathbf{x_i}) = \{\hat{\Phi}_1(\mathbf{x_i}), \hat{\Phi}_2(\mathbf{x_i}), \cdot\hat{\Phi}_l(\mathbf{x_i})\}$ (being l either k or $\binom{k}{2}$ depending on the selected approach). The final class will be the one whose row $\mathbf{M}(r)$ is "closest" to the vector. To calculate that distance they propose two possible measures: the *hamming decoding* and the *loss based decoding*.

We will not go in further details here, since our problem is multi-label and that implies:

1. Only the one-against-all approach can be applied
2. The decoding would imply not to select just one single class (the closest one), but a number of them. So the decoding should be rewritten to allow this, which is quite simple by applying any of the two methods mentioned in the beginning of this chapter.

Note that, since for a single classifier a document belongs to a class or not, we will define $y_i \in \{-1, +1\}$ as follows:

$$y_i = \begin{cases} +1 \text{ if } \mathbf{x_i} \text{ belongs to class} \\ -1 \text{ if } \mathbf{x_i} \text{ does not belong to class} \end{cases} \tag{6.2}$$

A classifier $\hat{\Phi}(x_i)$ will try to approximate y_i.

6.2 Classifiers overview

Now that we have strongly established how the multi-label classification will be implemented by means of binary algorithms, an study of these algorithms can be carried out to select some of them as candidates for our experiments. We give here an overview on some of them with special focus on those that will play a main role on our experiments.

- **Probabilistic**. The use of probabilities to predict a class was one of the first attempts in text categorization. *Naïve Bayes* formulas have been so widely used (not in our case, though) that we will describe them in section 6.4.
- **Decision trees**. These algorithms match a document $\mathbf{x_i}$ against the successive nodes of a tree to determine whether a class belongs to the document or not. Usually, the decision taken in a node is based on the value of a certain feature or by applying a logic decision rule. For example, we can built a CART tree [17] which grows by finding first which features best reduce the entropy (i.e. information gain) among possible classes and, once generated, the tree will be walked split after split until reaching a final

node which is associated to a class. Another example is C4.5 [95]. It is similar to CART, but it produces trees with varying numbers of branches per node. Both trees have to be pruned once generated in order to reduce over-fitting, and each one has its own method.

- **Linear classifiers**. A linear classifier consists of a vector w that acts as linear separator between two possible values: +1 and -1. In order to determine whether a document belongs to a class or not, we just compute the dot product between the vector w and the document $\mathbf{x_i}$. The sign of the resulting product will tell us if the document belongs to the class (positive) or not (negative). Well known examples of these algorithms are Widrow-Hoff, Support Vector Machines, Linear Logistic Regression and Perceptron. They deserve our special attention in section 6.5.

- **Decision rule**: The use of logic in text categorization systems has not been as success as more numeric oriented approaches. CPAR (*Classification based on Predictive Association Rules*) by Xiaoxin Yin and Jiawei Han [47] generates *Classification Association Rules* (CARs). The resulting classifier comprises a linked-list of rules ordered according to Laplace accuracy of individual rules. CPAR benefits are:

  1. uses greedy approach in rule generation, which is much more efficient than generating all candidate rules,
  2. uses a dynamic programming approach to avoid repeated calculation in rule generation,
  3. it selects multiple literals and builds multiple rules simultaneously, and
  4. it uses expected accuracy to evaluate rules, and uses the best k rules in prediction.

  It seems to perform better than any other related classifiers like FOIL [96], RIPPER [27] or the C4.5 decision tree.

- **Sample based**: Supervised learning means that we need, before any classification is performed by the future system, a set of samples that have been already labeled (generally by human experts). Then, we train a system to predict new labels. This later step, the training, is not needed in general in sample based systems. The *K nearest neighbours* algorithm is the best example: we assign the labels of those documents that are closer to the incoming document. As we can see, only a distance measure is involved in the process, the problem is that we do not really abstract a model for classes or infer any rule or hyperplane as for the rest of algorithms. Thus, we have to store all those samples for the K-NN to work and compute the distance to every document (although this could be optimized). Even with its simplicity, the results obtained by applying this algorithm are competitive, as shown by Yang and Liu [129].

- **Neural networks**: Neural learning has been used with effectiveness in natural language processing tasks. The Kohonen LVQ algorithm has been applied successfully in text categorization tasks [77]. Closely related to linear learning algorithms, this neural learning competitive algorithm is yet another available choice for text categorization.
- **Boosting algorithms**: boosting algorithms like *AdaBoost* [111] have shown very effective responses to the text classification task. The construction of a classifier as a linear compound of base classifiers is one of the most promising areas in Machine Learning.
- **Other algorithms**: Promising algorithms have appeared during past years. Among them, the leading ones are the *Maximum entropy modeling* [46] and *Logistic Regression* [45],

6.3 Bounding the error

In the revision of some of these algorithms, some authors make an strong effort in analyzing error bounds as a mean to study of the theoretical behaviour of the algorithm. In fact, this analysis is sometimes driven by the algorithm itself, as in the case of SVM or PLAUM (see Vapnik work [119]), where the determination of an expression for limiting the expected error of the method allow numerical methods in order to minimize it, producing as result the way the algorithm computes the classifier.

For any binary classifier $\hat{\Phi}_\alpha$ defined by its parameters α we can define analytically the *expected error* as:

$$E[\alpha] = \int_{\mathbf{x}} \frac{1}{2}|y - \hat{\Phi}_\alpha| dP(\mathbf{x}, y) \tag{6.3}$$

This is nothing but measuring the area with no overlap between the $\hat{\Phi}_\alpha$ and y_i functions. Since y_i moves between $\{+1, -1\}$ the fraction $1/2$ has been added. Empirically this turns into:

$$E[\alpha] = \frac{1}{2l} \sum_{i=1}^{l} |y_i - \hat{\Phi}_\alpha(\mathbf{x_i}, \alpha)| \tag{6.4}$$

where

$$\frac{1}{2}|y_i - \hat{\Phi}_\alpha(\mathbf{x_i}, \alpha)| \tag{6.5}$$

is the *loss* in the classification of $\mathbf{x_i}$. Equation 6.4 shows a very important measure that leads to the problem known as *Empirical Risk Minimization*

(ERM). This minimization is focused on reducing as much as possible the empirical error. As pointed out before, the minimization of bounds like the given one has lead to some of the most powerful learning algorithms proposed so far.

6.4 Naïve Bayes

A text classification network based on the Naïve Bayes model is said to be "naïve" because of the assumption of independence between features. Widely used due mainly to its simplicity, this classifier has shown good performance even compared to more sophisticated algorithms [41]. This makes possible to reduce the computational cost of finding the probability for a document $\mathbf{x_i}$ to pertain to a certain class c_j by using the equation:

$$P(c_j|\mathbf{x_i}) = \frac{P(c_j)P(\mathbf{x_i}|c_j)}{P(\mathbf{x_i})} \tag{6.6}$$

In a corpus, every document has the same probability, so $P(\mathbf{x_i})$ is the same for any i. Being a constant, we can eliminate it from the previous equation. Because of the independence assumption of features, we can replace $P(\mathbf{x_i}|c_j)$ by the product:

$$P(c_j|\mathbf{x_i}) = P(c_j) \prod_{k=1}^{l} P(f_k|c_j) \tag{6.7}$$

An estimate $\hat{P}(c_j)$ for $P(c_j)$ can be calculated using the relative frequency of assignment for the class in the collection:

$$\hat{P}(c_j) = \frac{n_j}{n} \tag{6.8}$$

where

n_j is the number of documents assigned to class c_j

In the same way, by using frequency of co-occurrence of features and classes (number of times a feature f_k appear in a document labeled by class c_j) we can approximate the probability $P(f_k|c_j)$ as follows:

$$\hat{P}(f_k|c_j) = \frac{1 + n_{kj}}{l + \sum_{h=1}^{l} n_{hj}} \tag{6.9}$$

where n_hj is the number of documents assigned to class c_j containing feature f_h and l was the total number of distinctive features in our space.

So, finally, we can conclude with an expression for computing the probability of a class c_j to belong to a certain document $\mathbf{x_i}$ by calculating a formula that does not need to weight the feature:

$$\hat{P}(c_j|\mathbf{x_i}) = \frac{n_j}{n} \prod_{k=1}^{l} \frac{1 + n_{kj}}{l + \sum_{h=1}^{l} n_{hj}} \tag{6.10}$$

Bayesian classifiers have reported good performance despite their simplicity. It is very desirable because of the low demand in information involved in this model: no weight for features and just simple computation of co-occurrences that can be very quickly calculated using bit set structures and the like. But some improvements in certain paradigms can be obtained by applying variations to the model shown above (for instance, the *multi-variate Bernoulli event model*).

A more sophisticated approach is the *multi-nomial model*, which captures also the frequency of a term in a document. This will, inevitably, produce a more complex model where the formula for the computation of the probability would be:

$$\hat{P}(c_j|\mathbf{x_i}) = P(|\mathbf{x_i}|)|\mathbf{x_i}|! \prod_{h=1}^{l} \frac{P(f_h|c_j)^{freq(f_h,\mathbf{x_i})}}{freq(f_h,\mathbf{x_i})!} \tag{6.11}$$

where $freq(f_h, \mathbf{x_i})$ is the frequency of feature f_h (generally, a word) in document $\mathbf{x_i}$.

McCallum and Nigam [78] found, by comparing both models over four different corpora, that the multi-nomial model was almost uniformly better than the multi-variate Bernoulli model. In empirical results on five real-world corpora they found that the multi-nomial model reduces error by an average of 27%, and sometimes by more than 50%.

6.5 Linear classifiers

Linear classifiers range from very basic algorithms like Widrow-Hoff, to complex ones like Support Vector Machines. A linear text classifier consists on a vector $\mathbf{w} = (w_1, w_2, \cdots, w_l)$ where each component w_i corresponds to a feature in the feature space. The complexity relies when estimating this vector from sample data. The classification is performed by testing if the dot product between the vector $\mathbf{w}$ and the document $\mathbf{x_i}$ is above a certain threshold t and then return the l top ones:

$$\mathbf{x_i} \text{ belongs to class if } \mathbf{x_i} \cdot \mathbf{w} > t \tag{6.12}$$

where

$$\mathbf{x_i} \cdot \mathbf{w} = \sum_{k=1}^{l} w_k x_{i,k} \tag{6.13}$$

Vector **w** represents the difference between a good linear classifier and a poor one. Although the final form of the vector is quite simple, the mathematical machinery involved in finding the best $(w_1, w_2, \cdots, w_l)$ values may involved expensive algorithms and numerical methods. We will start by describing some of the simplest ones in order to give a better idea of the way this vector can be generated and, later on, pass to two of the most performing methods: SVM and PLAUM.

Different training algorithms can be produced by varying the criterion function and weights-adjust procedure used. In this way we differentiate between *on-line* and *batch* searches. An on-line algorithm works by updating the vector weights for a sample document at a time. A weight vector at a given time is recomputed from the previous one and using document features and document class (either positive or negative), passing to the next updating for the next document in the training collection. This process stops when all the documents are processed or a certain criterion is satisfied. One of the benefits of this training is that we can always update the classifier with new incoming samples.

In the other hand, batch algorithms work by generating the final vector using all the training samples at a time. Therefore, the classifier is computed over the full set of training samples, no stop criteria is used. They use to provide better results, compared to on-line algorithms, but we will need to feed the whole collection to the learning algorithm if we want to consider additonal samples in the training collection, while this is not the case for on-line algorithms.

6.5.1 Basic classifiers

We will describe here three algorithms for training linear classifiers: *Rocchio algorithm*, *Widrow-Hoff algorithm* and *Exponentiated gradient* [67].

The **Rocchio algorithm** computes the weight vector **w** from an existing initial vector $\mathbf{w_1}$ and the set of training samples. Therefore, this is a batch algorithm:

$$w_j = \alpha w_{1,j} + \beta \frac{\sum_{i^+} x_{i,j}}{n^+} - \gamma \frac{\sum_{i^-} x_{i,j}}{n^-} \tag{6.14}$$

where

$i^+ = \{i : y_i = +1\}$; that is, those x_i belonging to the class

$i^- = \{i : y_i = +1\}$; that is, those x_i not belonging to the class

n^+ and n^- are the total number of positive samples and negative ones respectively

α, β and γ are parameters that control the relevance of the initial vector, positive samples and negative samples respectively. When $\alpha = 0$, $\beta = 1$ and $\gamma = 1$, $\mathbf{w}/|\mathbf{w}|$ is the difference in the mean scores for positive and negative training instances.

The **Widrow-Hoff algorithm** of Widrow & Stearns is an on-line algorithm. Thus, it updates the weight vector at each step. Initially the vector weights are set to zero. The update formula is:

$$w_{i+1,j} = w_{i,j} - 2\eta(\mathbf{w_i} \cdot \mathbf{x_i} - y_i)x_{i,j} \tag{6.15}$$

where η is the *learning rate* and controls the capacity of vector $\mathbf{w}$ to change by the influence of each new sample. Widrow-Hoff algorithm tries to adjust $\mathbf{w}$ towards a local minimal square loss $(\mathbf{w} \cdot \mathbf{x} - y)$. As final weight vector, it is considered to be better applying the average of the consecutive weight vectors generated during the process:

$$\mathbf{w} = \frac{1}{n+1} \sum_{i=1}^{n+1} \mathbf{w_i} \tag{6.16}$$

The **exponentiated gradient** algorithm of Kivinen & Warmuth is another on-line algorithm. Similar to Widrow-Hoff algorithm will update the weight vector at each run of the following formula:

$$w_{i+1,j} = \frac{w_{i,j} exp(-2\eta(\mathbf{w_i} \cdot \mathbf{x_i} - \mathbf{y_i})x_{i,j})}{\sum_{k=1}^{l} w_{i,k} exp(-2\eta(\mathbf{w_i} \cdot \mathbf{x_i} - \mathbf{y_i})x_{i,k})} \tag{6.17}$$

Kivinen and Warmuth [56] have studied these two algorithms (WH and EG) analytically in order to determine their error bounds. The error bound depends on the learning rate η chosen. If we take $\eta = 1/X^2$ for WH (the one used by default in our experiments) the square loss is bounded as follows:

$$E[(\mathbf{w} \cdot \mathbf{x} - y)^2] \leq 2\left(E[(\mathbf{u} \cdot \mathbf{x} - y)^2] + \frac{|\mathbf{u}|^2 X^2}{n+1}\right) \tag{6.18}$$

We can read in this formula that the expected square loss of $\mathbf{w}$ is upper bounded by twice the expected square loss of the best vector $\mathbf{u}$ (the one with the best fit to the data), plus a term that is quadratic in the norm of $\mathbf{u}$ and the maximum length (X^2) of any sample. As conclusions we can state that WH will have a low bound when the data is homogeneous (low entropy so we can find a vector $\mathbf{u}$ that fits the data well) and the number of training samples is relatively large.

In the case of EG we have chosen (as Lewis in his review [67]) $\eta = 2/(3R^2)$ as learning rate, where R is a value that bounds difference between minimal and maximal feature weights for any sample:

$$max_j x_{i,j} - min_j x_{i,j} \leq R; \forall x_i \in X \tag{6.19}$$

Then expected loss is bounded as given by the following expression:

$$E[(\mathbf{w} \cdot \mathbf{x} - y)^2] \leq \frac{3}{2}\left(E[(\mathbf{u} \cdot \mathbf{x} - y)^2] + \frac{R^2 \ln l}{n+1}\right) \tag{6.20}$$

Which can be understood simply that EG works better for large feature spaces, which is a good point due to the inner nature of text categorization tasks.

6.5.2 Logistic regression

The linear methods describe from now onwards carry a heavy machinery in probabilistic and statistical mathematical theory. Therefore, we will only provide a sight on the surface of these methods at the points we consider more relevant. Usually, the computational complexity involved to solve them (except for PLAUM) involves numerical methods of additional complexity.

Logistic regression models the conditional probability of a sample for a class as [131]:

$$P(y = 1|\mathbf{x}, \mathbf{w}) = \frac{1}{exp(-\mathbf{w}^T)} \tag{6.21}$$

To obtain an estimate of $\mathbf{w}$ we can apply the maximum likelihood, thus obtaining:

$$\hat{\mathbf{w}} = \arg\min_{\mathbf{w}} \frac{1}{n} \sum_{i=1}^{n} \ln(1 + exp(-\mathbf{w}^T x_i y_i)) + \lambda \mathbf{w}^2 \tag{6.22}$$

The last component λw^2 is an addition to reduce the computational cost of this minimization.

6.5.3 Support vector machines

Support Vector Machines are also a product of applied complexity theory developed by Vapnik [119]. Some years ago, Joachims proposed them for text categorization tasks, to profit from its robustness in high dimensional spaces [52].

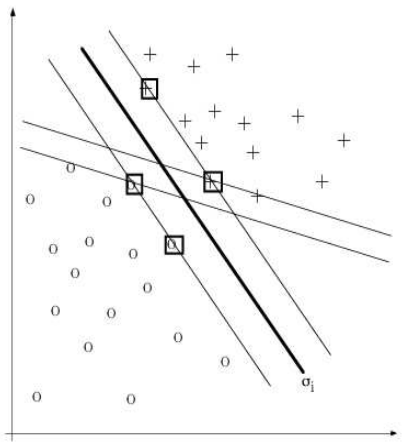


Figure 6.1: Two possible margins that linearly separate positive and negative samples. σ_i would be preferred since it represents better the distinction between classes (widest margin).

As for logistic regression, this is again a problem of optimization (see [20] for a detailed introduction to SVM). The name of the algorithm is given by the basic idea behind it: find those samples (support vectors) that delimit the widest frontier between positive and negative samples in the feature space (see figure 6.1).

The width of such border is know as the *margin hyperplane*, and SVM tries to find the maximal margin by applying *constraint quadratic optimization*, although every day new algorithms appear to solve this problem. The *Empirical Risk Minimization* problem described in section 6.3 is transformed into another problem by means of what Vapnik calls *VC bounds*. This new problem is called *Structural Risk Minimization* and SVMs are a solution to them.

At the end, what we will get is the hyperplane $\mathbf{w} + b$ with the characteristics already mentioned. The decision function will be:

$$\hat{\Phi}(\mathbf{x}) = \sum_{i=1}^{s} \alpha_i y_i K(\mathbf{x}, \mathbf{w_i}) + b \tag{6.23}$$

where s is the number of *support vectors* (samples which define the margin), α_i is a value that makes the algorithm allow a certain error, $w_i + b$ is the hyperplane. We have K as an operator for space transformation known as *kernel function*, that can map the space to another one where the separability between samples may be easier (i.e. for *non-linear* cases).

Support Vector Machines have been applied successfully in many text classification tasks due to their principal advantages:

1. They are robust in high dimensional spaces. Over-fitting does not affects so much the computation of the final decision margin.
2. Any feature is important. Even some features that could be considered as irrelevant ones have been found to be good when calculating the margin.
3. They are robust when there is a sparsity of samples.
4. Most text categorization problems are linearly separable.

6.5.4 Perceptron learning algorithm with uneven margins

Another linear classifier very close to SVM is the perceptron learning algorithm with uneven margins (PLAUM) [105, 127]. This algorithm shares with the previous one its foundations as it also derives from the idea of finding a margin, and their authors claim that it works better than SVM for text classification tasks. The classification is done in the same way as for SVM:

$$\hat{\Phi}(\mathbf{x}) = \mathbf{x} \cdot \mathbf{w} + b \tag{6.24}$$

But the way $\mathbf{w}$ and $\mathbf{b}$ are computed is so simple that makes it very easy to implement compared to the numerical needs of the SVM algorithm. The algorithm is given in figure 6.2. For non linear cases, PLAUM uses the λ-trick which consists in rewriting the dot product $\mathbf{x} \cdot \mathbf{x}$ by:

$$\mathbf{w} \cdot \mathbf{x_i} = \sum_{j=1}^{l} w_j x_{i,j} + \lambda w_{l+i} \tag{6.25}$$

where λ is the value for the ith coordinate and zero elsewhere granting the separability of samples.

Require:
n linearly separable training samples X_t
A learning rate $\eta \in \Re^+$
A maximum epochs parameter T
Two margin parameters $\tau_{+1}, \tau_{-1} \in \Re$
Algorithm:
epoch $\leftarrow$ 0; i $\leftarrow$ 1; update $\leftarrow$ m
$\mathbf{w} \leftarrow \vec{0}$; b $\leftarrow$ 0; $R \leftarrow \max_{\mathbf{x_i} \in \mathbf{X}} |\mathbf{x_i}|$
repeat
if $y_i(\mathbf{w} \cdot \mathbf{x_i} + b) \leq \tau_{y_i}$ **then**
$\mathbf{w} \leftarrow \mathbf{w} + \eta y_i \mathbf{x_i}$
$b \leftarrow b + \eta y_i R^2$
updated $\leftarrow$ i
end if
i $\leftarrow$ i + 1
if (i > n) **then**
i $\leftarrow$ 1
epoch $\leftarrow$ epoch + 1
end if until (i = updated) **or** (epoch $\geq T$)
return$(\mathbf{w}, b)$

Figure 6.2: The PLAUM learning algorithm

6.6 Boosting algorithms

Boosting is a technique in machine learning for combining classifiers in an iterative way so that performance is improved. The final classifier is composed by several *weak* classifiers, which have been generated through the learning process. A more formal definition would describe that, basically:

Given:
$(x_1, y_1), \cdots, (x_m, y_m)$ where $x_i \in X, y_i \in Y = \{-1, +1\}$
For $t = 1, \cdots, T$:
Train weak learner and produce weak hypothesis $h_t : X \rightarrow \{-1, +1\}$
Update some weights and parameters
Return final hypothesis H as combination of $h_1, \cdots, h_T$

AdaBoost is one example of these techniques [111]. It works by maintaining a distribution of weights (one per sample) that is updated to force weak learners concentrate on those samples whose classification is "harder". The generic AdaBoost algorithm is shown in figure 6.3. As we can see, this algorithm

- increases the weight of miss-classified samples at each round

- the final hypothesis H is a weighted combination of T weak hypothesis h_t in function of their error

---

Given:
$(x_1, y_1), \cdots, (x_m, y_m)$ where $x_i \in X, y_i \in Y = \{-1, +1\}$
Initialize $D_1(i) = 1/m$
For $t = 1, \cdots, T$:
· Train weak learner using distribution D_t
· Get weak hypothesis $h_t : X \to \{-1, +1\}$ with error:
$E_t = P_{r_i \sim D_t}[h_t(x_i) \neq y_i] = \sum_{i:h_t(x_i) \neq y_i} D_t(i)$
· Let $\alpha_t = \frac{1}{2} \ln(\frac{1-E_t}{E_t})$
· Update D:

$$D_{t+1}(i) = \frac{D_t(i)}{Z_t} \times \begin{cases} e^{-\alpha t} & \text{if } h_t(x_i) = y_i, \\ e^{\alpha t} & \text{if } h_t(x_i) \neq y_i, \end{cases} =$$
$$= \frac{D_t(i) exp(-\alpha_t y_i h_t(x_i))}{Z_t}$$

where Z_t is a normalization factor so that D is a distribution
· Output final hypothesis:
$H(x) = sign(\sum_{t=1}^{T} \alpha_t h_t(X))$

Figure 6.3: AdaBoost algorithm

The algorithm only focuses on the case of binary valued weak hypothesis predictions. It is different to previous boosting algorithms in that it is *adaptive*, since it depends on error rates of each individual weak hypothesis (hence its name).

The benefits of this algorithm are due to mathematical properties of the training error and the generalization error. The first one implies that AdaBoost does not need a lower bound a priory to work, since it is computed adaptively. The second one, as proved by Schapire and his colleagues, tells us about the independence of T from the upper bound of the generalization error, which makes the algorithm robust against over-fitting.

Boosting is closely related to linear programming and on-line learning. Many similarities in their formulae can be found between this algorithm and Support Vector Machines (and some significant differences, though). Two variants of this algorithm were proposed to deal with multi-class classification: *AdaBoost.MH*, defined on the basis of the *Hamming loss* minimization, and *AdaBoost.MR*, which relies on ranking loss minimization. We will not go into further details about these two variants here, since we did not apply this specific algorithm in our experiments, but due to its growing relevance in the TC paradigm, this brief introduction was imposed.

Whether we choose a loss to minimize or another, we have to design how weak learners are generated. Schapire proposes a very simple one for text classification, it has the basic form of a one-level decisions tree:

$$h(x,l) = \begin{cases} c_{0l} & \text{if } w \in x \\ c_{1l} & \text{if } w \notin x \end{cases} \tag{6.26}$$

where $c_{0l}, c_{1l} \in \Re$, w being a term and x a sample document. The weak learner is chosen by computing those values for all terms, then the weak learner returns the hypothesis (the term) with the lowest score. So, each weak hypothesis just returns a real value when the terms appears and another value if the term is not present in the document. This value used to be the score Z_t in AdaBoost.MH and an approximation of Z_t in AdaBoost.MR (since there is no analytical solution).

Results shown by Schapire and Singer on the Reuters corpus [111] promoted AdaBoost.MH outperforming other algorithms like Rocchio, Naïve Bayes and Sleeping Experts. Unfortunately, in our opinion the Reuters collection is not a very suitable collection for performing multi-label classification experiments due to the low number of categories assigned to each document (rather few, two or three categories per sample in rare cases). Similar results were found by Weiss et al [124] also using the Reuters collection.

A very interesting point about loss minimization and margin based algorithms like Support Vector Machines is that they are closely related. Some researchers, including Robert Schapire, state the close similarity between these minimization algorithms, as also seen by Rätsch et al [102] when designing unsupervised methods. Although more recent boosting algorithms are available, like *SmoothBoost* [113] and *BrownBoost* [40], leveraging approach studies all them under the same view, linking the basis of these techniques with the Theory of Optimization [80].

6.7 Thresholding

At this point, we know, how to process a collection of text documents to feed a learning algorithm in order to generate a model that will allow us to perform classification tasks. We have already seen the different classification paradigms: binary, multi-class and multi-label, the last one being the most complex, since it needs of all the properties of the former two. As seen, we can construct a multi-label classifier by running a binary classifier for each single class.

The result of the binary classifiers will be a sequence of values (margin distances, similarity measures, probabilities, etc.), one per class, that will help us in determining the n most related classes, i.e. the classes that will be the final output of the multi-label classifier for every given document to be categorized. The expect result from a binary classifier is just a *yes or no* answer, and to select

the final choice the resulting measure of the algorithm must be *thresholded*, that is, compared to a threshold to establish the discrete goodness of the class for the document: if the value is under the threshold, the class is discarded. This threshold is usually called *cut*. The most well known cuts defined in the literature are those ones determined empirically, that is, over evaluation samples to adjust it to the one we hope will provide best performance (Yiming Yang has a nice review of them [130]):

- **R-cut**. This is, simply, directly applied when selecting a set of categories. The t top ranked classes are selected as positive classes, the rest is considered negative (not assigned).
- **P-cut**. Here, the focus is on documents, rather than on categories. The documents are ranked for a given category, and the k_j top-ranking documents are assigned to the class:

$$k_j = P(c_j) \times x \times m \tag{6.27}$$

  where

  k_j is the number of documents assigned to category c_j,

  $P(c_j)$ is the prior probability (over the training set) for a document to be member of class c_j,

  m is the total number of classes in the collection, and

  x is a real value that must be tuned to get best global performance (its range is $[0, n]$, being n is the total number of documents in the training set)

- **S-cut**. This threshold is fixed per category, setting the cut that produces the best performance over the training set. The difference with respect to the former two is that it is optimized per class. It does not guarantee that the global optimum for the training set will be reached.

Yang observed that *S-cut* tends to over-fit while *P-cut* performed well on rare categories. Nevertheless, *P-cut* is not applicable in our experiments, since documents may be classified individually.

In a multi-label environment, we may profit from those measures without considering thresholding and by using that information in selecting the final set of classes. Basically, two main approaches can be considered for determining these top classes: *all-positive* or *most similar*. We will not dive into additional existing and complex approaches, since our main goal is to demonstrate the feasibility of a multi-label classifier for massive labeled collections like the HEP collection. The two presented approaches are strongly related to the way binary classifiers are selected. The output value of a margin based classifier like SVM is not always a comparable number when weighting a choice of a class against another one. Therefore, thresholding is not suitable for margin based classifiers as shown in our experiments.

Chapter 7

Evaluation

The evaluation of a text categorization system is based on test samples that have been already labeled by human experts. For text categorization systems, the evaluation strategy used is inherited from traditional Information Retrieval experience. David D. Lewis has an interesting review on how evaluation is carried out in TC systems [63]. The start point is to compare the matches between human-assigned key words and computer-assigned ones. We can summarize four possible situations in the following *contingency table*:

class c_i		**assigned by expert?**	
		YES	NO
assigned by classifier?	YES	TP_i	FP_i
	NO	FN_i	TN_i

where

- TP_i *True positive.* Those assessments where system and human expert agree for a label assignment.
- FP_i *False positive.* Those labels assigned by the system that does not agree with expert assignment.
- FN_i *False negative.* Those labels the system failed to assign as they were by human expert.
- TN_i *True negative.* Those non assigned labels that also were discarded by the expert.

Notice the subscript i at every value. That means that we compute those numbers for every class by looking the documents it has been assigned to, and the same could be done for every document by looking at assigned classes. The consideration about what to consider for final global values (we will see how to compute them later) depends on whether we want to focus on the quality of the system assigning classes to documents or the quality of the system for certain

classes.

By combining these values some well known measures can be computed:

$$\textit{precision (P)} = \frac{TP}{TP + FP} \tag{7.1}$$

$$\textit{recall (R)} = \frac{TP}{TP + FN} \tag{7.2}$$

$$fallout = \frac{FP}{FP + TN} \tag{7.3}$$

$$accuracy = \frac{TP + TN}{TP + FP + FN + TN} \tag{7.4}$$

$$error = \frac{FP + FN}{TP + FP + FN + TN} \tag{7.5}$$

$$F1 = \frac{2PR}{P + R} \tag{7.6}$$

$$\textit{interpolated break-even (IB)} = \frac{P + R}{2} \tag{7.7}$$

The *precision* (7.1) tells us hown well the labels are assigned by our system (the fraction of assigned labels that are correct). The *recall* (7.2) measures the fraction of experts labels found by the system. These two measures are well known in information retrieval systems ([118]). The balance between these two values is a difficult task, since usually the improvement in one leads to a decline in the other. A multi-label text categorization system returning few labels for a document may have high precision but low recall. On the other hand, a system returning many labels for the same document will show a low precision but a higher recall, in general. Since we are looking for systems showing both high precision and recall, we may want to define an overall measure that will gives us a notion of the overall performance obtained by the categorization engine. The two most used measures are the *interpolated break-even* and the *F1* measure (proposed by Van Rijsbergen [118], pp. 168–176).

The former is nothing but the average between precision and recall, and some researchers do not consider it a very realistic measure, since it is not always possible to configure a system so that both precision and recall end up with a similar value. The use of the *F1* measure is more realistic, since this quotient is more sensitive to low values of precision or recall. Therefore, in our experiments, the *F1* measure will play a central role as indicator of the goodness of the system.

7.1 Computing global measures

Formulae given before provide nothing but measures for a single class c_i. It is very important to remark that most of the experimental frameworks traditionally set just focus on classes and average over them all. In our case, this choice is not trivial: we want to construct a TC for classifying documents with a considerable possible number of topics (from five to twenty per document approximately). Therefore, in our opinion, the average should be done over documents rather than classes. In any case, to obtain global values we can use two different averaging approaches: *micro-averaging* and *macro-averaging*, first introduced by Lewis [66].

Micro-averaging

This averaging method computes defined measures by adding up all the individual TP_i, TN_i, FP_i and FN_i of every class (or document) to global ones. The formulas for precision and recall would be:

$$precision^m = \frac{\sum_i \sum_j TP_{ij}}{\sum_i \sum_j TP_{ij} + \sum_i \sum_j FP_{ij}} \tag{7.8}$$

$$recall^m = \frac{\sum_i \sum_j TP_{ij}}{\sum_i \sum_j TP_{ij} + \sum_i \sum_j FN_{ij}} \tag{7.9}$$

where TP_{ij}, FP_{ij} and FN_{ij} are the number of true positives, false positives and false negatives, respectively, found for the class i in the evaluation document j.

7.1.1 Macro-averaging

When we average by macro-averaging we have to set beforehand on what basis the averaging will be performed. It works by computing the evaluation measure first at class or document level, but one of the two must be chosen. Then, all the measures are averaged into the global ones. Considering the precision and recall measures we would have:

$$precision_j = \frac{\sum_i TP_{ij}}{\sum_i TP_{ij} + \sum_i FP_{ij}} \tag{7.10}$$

$$recall_j = \frac{\sum_i TP_{ij}}{\sum_i TP_{ij} + \sum_i FN_{ij}} \tag{7.11}$$

$$precision^M = \frac{\sum_j precision_j}{n} \tag{7.12}$$

$$recall^M = \frac{\sum_j recall_j}{n} \tag{7.13}$$

n being the total number of documents in the collection. Here, we have considered a macro-averaging in the basis of documents.

7.1.2 Cross validation

A supervised learning algorithm needs labeled data to be trained and labeled data to be tested. Of course, these two sets must be disjoint to avoid a false estimation of performance, and this is the reason why multiple runs of the experiments are usually launched with a different partition at each turn. In our experiments, data will be split into three different sets: *learning*, *validation* and *testing*. The first set will be used for generating the samples that will be used to train classifiers. The evaluation set will determine the best classifier for each class and its threshold (see previous chapter) when needed. The last set will be used to compute the evaluation measures described here.

As soon as data has to be split into disjoint sets, one for training and another for statistical analysis, our results may differ depending on how we have chosen such partitions. For that, cross validation strategies tend to reduce the possible bias introduced by this process, by partitioning randomly several times the same data set and, thus, allowing some items to be in each of the two sets (see [74] for a broad introduction to cross validation). Thus, several partitions are made and the final result is an averaged measurement of the experiment over every partition made.

Stratified n-fold Cross validation [58, 81] consists in dividing the document collection into sets of equal size. At each turn, one set is used for testing and the rest for training the system. In our case, 10 disjoint folds are generated. At each turn, 6 folds will be used for learning, 3 for evaluation and 1 for testing, in such a way that every subset will be used once for testing purposes. Some studies, as pointed by Witten and Frank in their book [125], estimate that 10 is a stable number for assuring a certain statistical independence when computing the evaluation values from the partitions done on the collection.

We have elaborated our own experiments to justify this decision, and the results are very interesting (see experiment 12.1).

7.2 Statistical tests

In order to be sure that two distributions are significantly different, we perform some statistical tests. For any test, we define a "null hypothesis", which says that there is no difference between the two populations of interest. In text classification the two populations may be two learning algorithms to be compared, two weighting schemes or two lemmatization algorithms, and the values for each distribution would be the values of a certain measure through all classes. Based on the way these distributions are generated we can apply different tests, like the Wilcoxon test for paired comparison if we used a fixed split of training and test samples.

Many authors used a paired t-test on the results of n-fold cross-validation experiments with n=5, 10, 20 or 30, or n experiments with random (stratified)

train/test splits. Dietterich [33] reviewed these and other tests and found that the paired t test often has high *type 1* error (i.e. it finds a difference when no difference exists), especially with random train/test splits. He proposed to use the 5x2cv test instead. For this test, 2-fold cross-validation with random split is performed five times and a statistic is measured to verify if the null hypothesis is rejected (so the differences are significant) or not (so we cannot consider any improvement of one approach over the other). Another interesting conclusion is that n-fold cross validation shows a high Type-I error, mainly due to overlap among different splits.

Alpaydin [8] proposed a variant of the 5x2cv test that combines the results of the 5x2=10 cross-validation trials in a more robust way, resulting in lower type 1 error and higher power of the test.

But in our case, the n-fold cross validation is not performed by applying the test computation that Dietterich proposes for this testing approach: we compare over *averaged* values. Therefore, the effects of high variance are lower (as studied in experiment 12.1). These tests are non-applicable because each algorithm is tested over different partitioned folds.

7.3 Final considerations

In the literature, evaluation of categorization systems is usually performed by micro-averaging rather than macro-averaging. We can see that in the case of micro-averaging is does not matter whether we focus on classes or on documents. On the other hand, when macro-averaging we have to be very careful in stating on which basis it has been done, since results may differ drastically. Furthermore, in multi-label classification problems, measures based on macro-averaging must be considered with caution: if we have classes that are very frequent and the performance for them is bad, less used classes which are better assigned can produce good results for both precision and recall when macro-averaging by class, while the behaviour of the system per document is not that good. Obviously, this preference in the averaging system used by current TC research is due to the lack of real multi-labeled corpora, where the described problem becomes relevant. Lewis [63] **emphasizes the importance of development and release of standard collections**.

Another important issue considered in some analysis (see [18]) is the *inter-annotator agreement*, that is, even for human experts, the labeling varies. Therefore, two different experts are very likely to select different sets of labels. This level of agreement can be used as base for our evaluating against it, e.g. if the inter-annotator agreement is 80% in recall and our systems reaches that percentage, then we would conclude a system recall of 100%, since our system is as good a manual annotation. In our case we will not take this into consideration, due to the fact that no precise information about such agreement in labeling High Energy Physics documents is available.

To conclude this review on evaluation measures, we consider that our system should be evaluated mainly on the basis of macro-averaging in a per document basis, since the goal of the system is to be used for labeling documents and we consider this is the best approach to search for a good system configuration.

Chapter 8

Applications of key word assignment

Keywords have many applications [87], since they have been useful in the librarian community for centuries. But the feasibility of automated systems is bringing new ideas not only to this community, but to all the information society.

The construction of hand-crafted thesauri for use in computer applications dates back to the early 1950s with the work of H. P. Luhn, who developed a thesaurus for indexing scientific literature at IBM. The number of thesauri and systems is now growing steadily because manually or automatically key worded documents have many advantages and offer additional applications compared to simple documents that do not have the added value of being linked to thesauri. Depending on whether people or machines make use of key words assigned to documents, we distinguish different uses depending in the way these key words are manipulated:

- **Human manipulation of key words**. Human users mainly use key words for *browsing* and *searching* of document collections.

- **Using key words for automated computer-based manipulation**. The fact that key words were traditionally developed for human readers does not necessarily mean that they can only be used by people. Several powerful applications have shown that descriptors can well be used to represent document contents for a number of automatic procedures.

8.1 Human manipulation of key words

8.1.1 Browsing

Keywords can be (actually, they *are*) used to facilitate the browsing of document collections. These can either be part of a whole collection or of the small subset returned by a search operation.

- Use of key words as a document summary. Thesaurus descriptors are usually a small list of carefully chosen terms that represent the document contents particularly well. Depending on the thesaurus, they are of a summarising, conceptual nature. They often do not occur explicitly in text so that they are of a completely different nature from full text indexes. Descriptors function as kind of an abstract summary and give users a quick and rough idea of the document contents. This helps the users to quickly sieve out the most important or relevant documents out of a large collection. The use of key words as a means for automatic summarization is an interesting application already in practice in many digital libraries and on-line catalogues.

  For High Energy Physics documents this can speed up the search process in specialized collections which grow in hundreds of documents every week [86].

  If the thesaurus is multilingual, this summarising function also works across languages, i.e. a user will see a list of key words (a summary) in their own language of a document written in another language [116].

- Use of key words for Document Navigation. If the database containing full texts and their key words offers hyperlinks based on key words, it is possible to navigate through the document collection by starting with one document and searching for similar documents by clicking on one or more of the key words to see other documents indexed with the same descriptors.

- Classification of documents. In some search engines the results of a search are classified "on the fly" into categories so that the browsing of documents is easier and more self-organized. The user can distinguish faster between interesting documents and irrelevant ones. Although many of these systems use words from documents to label automatically generated categories, others select them from a controlled vocabulary. An example of such a system is GRACE [5].

  However, the use of key words for classification goes beyond the pure retrieval domain. The *freedesktop.org* ([3]) project promotes the use of key words for icon and menus arrangement (representing application launchers) in the main menu of the desktop where applications are internally attached to a list of categories. It means that there is not a predefined

taxonomy to which program launchers are classified. Instead, programs are labelled with key words from which menus are created in the graphical interface of an operating system (like the Gnome [4] desktop available for Linux and other operating systems).

- An interesting feature that will be tested at CERN's library is the use of key words to help users in navigating through document references, allowing them to recognize the subject that is shared by the reference and the document the reference belongs to.

8.1.2 Searching

Key words are also helpful during the search phase. For example:

- For query expansion. Some authors, like Vassilevskaya [120] among others, propose the use of a controlled vocabulary for query expansion. In the query formulating process, the query is passed to the automatic assignment tool and some thesaurus key words are suggested to the user. These can then either be chosen instead of or in addition to the query.

- For cross-lingual searching. When the thesaurus used for indexing is multilingual, users can be given the option to use thesaurus descriptors as search terms. The search can then be carried out using another language version of the search terms to achieve cross-lingual document search and retrieval, as achieved by the system developed by Pouliquen et al. [99].

- Conceptual search. Descriptors provided by a thesaurus have a relevant advantage over terms selected for automatic full-text indexing. Concepts which can be expressed in several synonymous ways, such as "Chemotherapy" and "Drug Therapy", are conflated to only one form ("Drug Therapy"), while phrases can be treated as single concepts ("stage IIIB breast cancer", for example). This is one of the conclusions found by using the MetaMap Indexer for medical documents [62].

- Guided search. When searching in highly specialized domains, key words can be used as a directory or subject tree for the user who is not able to make his/her information needs explicit as a set of query terms. If documents in the collection have been labelled with key words and the structure of the thesaurus is hierarchical, the user can drill down the categories narrowing the search space. This could be the electronic equivalent of browsing the Universal Decimal Code (UDC) when we enter in a library for the first time. As we can see, this tree can be navigated profiting from semantic relations between key words, hence we may find related topics, general topics, and further relationships.

8.2 Using key words for automated computer-based manipulation

A computer can use key words to perform certain automated operations. Some of the most useful ones are:

- Using descriptors from a hierarchically organised thesaurus allows searching by subject field, rather than full text search terms. For example, searching for "RADIOACTIVE MATERIALS" can automatically be extended to all individual instances of radioactive elements, such as 'uranium', 'plutonium', etc. This form of query expansion is being used for some domains like High Energy Physics [120].

- Multilingual document similarity calculation. As the list of thesaurus descriptors of a text is a semantic representation of this text, texts can be compared with each other via these descriptor lists. The idea is that, the higher the number of content descriptors two documents have in common, the more similar the documents are. For multilingual thesauri like Eurovoc ([2]), which currently exists in over twenty languages with one-to-one translations for each descriptor, the document similarity calculation is even possible for texts written in different languages. [99] have shown that the translations of a given document can quite reliably be identified in a multilingual document collection because they are correctly identified as the most similar document to the original document.

- Multilingual clustering and multilingual document maps. The same cross-lingual document similarity measure can be used as input to further multilingual applications. These include multilingual clustering and classification of documents, as well as the visualisation of multilingual document collections in a single document map ([100]).

- The Semantic Web. Our opinion is that all these technologies will play a key role in the development of the Semantic Web [15]. Considering that the whole structure of the semantic web depends on RDF (*Resource Description Framework*) and that there are already some projects to use thesauri like a schema (ontology) defining the terms used to represent the RDF version of WordNet, we can conclude that this is a promising area of research. A web of documents can be related via their associations between key words.

- The Semantic Grid. Not only documents can be linked using key words, but any type of service (for example, any web service) can be attached with key words which could be automatically assigned using the description of the service as basis. The Semantic GRID ([39]) objective can be reached faster using subject enhancement, i.e. key-wording.

> Therefore, we could imagine a scenario where we look for a certain service, e.g. a database of iron providers. We get the key words of the service which may have been generated from the content of top web pages in the portal of this service (the pages which let us access the database via web forms or any other web based interaction). These key words show us that there is another database which offers specific iron made toys, since the thesaurus splits that key word *iron manufacturer* into the subtopics *iron-made toys manufacturer, naval manufacturer*, etc. Thanks to the semantic network created from key words relationships we are able to find the provider we need.

There are more areas where multi-label classication may be useful. For instance, question-answering systems may profit from the techniques developed in text classification, as we can see in the paper by García-Cumbreras and others [29]. Another example is the classification of medical images using small pieces of information attached to them as found by Martín-Valdivia and others [76]. We are sure that not all possible applications have been introduced here, and that with time new benefits will be discovered.

Chapter 9

Current working systems

Nowadays, the use of text categorization covers many different areas, like document routing or topic detection and tracking, but we are not aware of operational production tools which implement multi-label categorization.

Automatic indexing based on word frequency can be traced back to the 1950's and the work of Luhn [72] and Baxendale [13]. Several works have appeared since then, adding current approaches (conflation algorithms, weight normalization...). Many approaches have arisen to give an automatic solution to indexing tasks. But, since wide knowledge is required in this process, we can assume that there is no full automatic indexing system available at this point of time (by providing indexes from a controlled vocabulary). Even good systems must be supervised by experts to control and retouch generated results.

9.1 Approaches

The availability of large collections of documents in full text format has represented the beginning of a new era in information retrieval. Much research is being done around natural language processing. Early works of Salton are a good starting reference [107]. Fields like information extraction, text analysis, text mining and others are sharing their knowledge to propose solutions in a new environment. This environment is growing and demanding more and better tools to be used in text processing. Multi-label classifiers for large numbers of classes are not yet good good enough to be used fully automatically, although results obtained so far may be considered enough in certain cases.

In this field many approaches from other areas in information retrieval can be applied. Automatic key-wording can be related to automatic summarization (which also tries to generate abstracts from full text documents). In this area many relevant algorithms have been developed. They range from classic confla-

tion algorithms to reduce the components of a document to its essential items (see [103]), to those which treat document as a whole, identifying discourse trees [75] or conceptual phrases [28]. Nevertheless, the algorithms that focuse on text categorization are the ones based on binary classifiers, margin-based classifiers and, in general, all those techniques coming from the are of machine learning. We will describe some of them in the following sections and specify why we have preferred the use of some of them over others. The number of applied systems is also wide, and it is not our intention to provide a deep historical review of all of them. The systems introduced in this chapter have been selected for historical reasons and for their close (in the data managed or in the approach implemented) relation to the system this work proposes.

9.1.1 BIOSIS

One of the most sophisticated programs for automatic indexing, developed at BIOSIS, was discussed by Vleduts-Stokolov [121]. Words appearing in the titles of journal articles were matched against a Semantic Vocabulary, consisting of about 15,000 biological terms, and these in turn were mapped to a vocabulary of 600 Concept Headings. Thus, a Concept Heading could be assigned by computer on the basis of words/phrases occurring in titles. Vleduts-Stokolov reported that about 61% of the Concept Headings assigned by humans could be assigned by a computer based on titles alone. If only primary and secondary assignments are considered (BIOSIS used a three level term assignment scheme: primary, secondary, and tertiary), about 75% of the assignments could be performed automatically.

9.1.2 MeSH

Indexers at the National Library of Medicine use the Medical Subject Headings (MeSH) [90] which currently contains about 18,000 unique subject terms. The indexers previously used a printed annotated alphabetical list. Increasingly, the indexers consult the on-line vocabulary. The list includes annotations. The numbers beside the terms are the MeSH tree numbers. MeSH is arranged into hierarchies or trees.

The indexers apply about 8-10 terms to each article depending on its length and how much coordination of terms is needed. MeSH is used first to match a text word in the title or abstract to the terms in the permuted vocabulary. Indexers may also look at previously indexed terms in MEDLINE. The indexer can interact with the MeSH file from within the indexing application. The indexer can neighbor terms and access the display or tree terms within the indexing application. Indexers can also display the MeSH annotations and other relevant information. In addition to the 18,000 terms in the MeSH vocabulary, there are also 100,000 supplemental chemical terms. These terms are indexed differently because the file contains a large amount of chemistry information,

in the areas of drug administration and diseases studied at the molecular and biochemical level. If the chemical term is not found, an entry is created by the indexer in the Supplementary Chemical list, rather than adding it into the MeSH thesaurus itself.

Each indexer produces about four articles per hour. Typically, about 400,000 items per year are indexed from about 4,000 biomedical journals world-wide. That is work for about 54 indexers full-time.

Important work has been done in producing an automated system for this manual procedure for the related ADM thesaurus. The system is called *Nomindex* and has been developed by Bruno Pouliquen[94]. The vast number of labels available from the thesaurus are selected for texts in French by a method composed by different techniques: multi-word detection, document similarity, document segmentation, etc. Results obtained proved that these kind of systems can be a valid solution in that context, and that probably could be exported to other environments, as we have tested in our work.

9.1.3 NASA MAI System

The NASA Center for AeroSpace Information (CASI) is part of the NASA Scientific and Technical Information (STI) Program. The STI databases contain over 3 million records. At least 2 million of them are technical reports and journal articles. The CASI indexers use the full document for abstracting and indexing with exceptions for cases where the original document is not available. The NASA MAI system [53] is an aid system, i.e. no final list of key words is provided by the system, but rather a set of possible key words that the user can select and browse.

The importance of the system resides in the capability of helping the indexers to produce more coherent labellings, since human experts are driven by the system to narrower sets of possible descriptors. Hence, we cannot provide an automatic evaluation of the system and not very much information is given about how the list of proposed key words is generated. In any case, we must mention it to underline the importance of automatic classification systems not only operating in standalone versions, but helping the human expert in the search of summarizing topics for complex texts, reducing the work overload of the expensive indexing task.

NASA CASI has used Machine Aided Indexing (MAI) for many years. The MAI system does not perform full natural language processing or grammatical parsing. Instead it uses certain rules which process the text and give results approximating the results of full grammatical parsing– but without the computational overhead. The rules have been developed to get at those features of text that have the most potential for representing index concepts.

MAI uses a large knowledge base (KB) of over 170,000 words and phrases. Maintenance is ongoing. The KB also contains other types of entries. Some

entries in the KB function in coordination with the computational rules during the text analysis process to direct the creation of extended phrases; the KB plays a role in the parsing of the text; it is not just a list of phrases with a straight look up.

The values of precision and recall are around 50%. It is a quite good value due to the wide range of forms of documents processed and fields covered.

9.1.4 Eurovoc

There are very few multilingual thesaursi around, and automatizing one of them leads to many practical applications. That is the case of the Eurovoc thesaurus [2], used by about twenty European parliaments, such as the Europen Parliament itself, the European Commissions Publications Office (OPOCE), and several European Union member state parliaments such as the Swedish Riksdag and the Spanish Congress of Deputies. In all these environments, the thesaurus is applied for manual indexing of generated documents. As Eurovoc is multilingual with one-to-one translations, we can search documents in one language and then retrieve related documents in many other languages [99] (up to 190 language pair combinations). Actually, as direct practical use of the automatic indexing engine, the Spanish Congress of Deputies is currently installing an interactive version of this system.

The Joint Research Centre has worked on automatic assignment during last the years [18, 100, 116] and is using the system in its daily news analysis system NewsExplorer[1] to link news articles across languages [98].

Eurovoc has over 6,000 classes (this number varies depending on the version of the thesaurus), although many descriptors are never or rarely used so that the system could only be trained for approx. 3,500 descriptors. The system maps documents onto Eurovoc by carrying our category-ranking classification using Machine Learning methods. In an inductive process, it builds a profile-based classifier by observing the manual classification on a training set of documents with only positive samples. Before the classifier acts, some linguistic pre-processing is carried out to such lemmatization, stop words removal and multi-words detection. This preprocessing is strongly linked to the type of language being processed, and specilized pre-processing was computed for every language handled. A detailed description of the process can be found in [18].

The results obtained in a per-document basis for an English and a Spanish set of parliamentary documents were produced by a manual inspection of the output from the computer-based indexer. The test collection was composed by almost 60,000 English texts of different types; this number was smaller for the Spanish corpus. With an average of 5.65 descriptors per document, we can consider this as a truly multi-labeled collection. Taking human performance as benchmark, values of 86% and 80%, repectively, were found. These results

[1] available publicly at `http://press.jrc.it/NewsExplorer`

showed that a system like this performs significantly well for such a complicated task.

9.1.5 Others

Fall et al [37] have applied multi-class (but not multi-label) text categorization against a collection of legal documents about patents using different approaches like k-NN, SVM, Naïve Bayes and SNoW. Already in their study it is clear how difficult it is to work with collections with a high number of possible classes (114 different classes with 451 subclasses). It is a pity that they only look at precision to evaluate the four different approaches, but the conclusion points in the direction of SVM as best performer.

The indexer of Reginal Ferber [38] is very similar to the solution provided by HEPindexer: also values *term-to-class* are built up from *term-to-document* and *document-to-class* relations, using a very simple formula:

$$wtk(t,k) = \frac{P(t \wedge k)}{P(t)P(k)} \tag{9.1}$$

These probabilities are computed from simple document frequency quotients as used commonly through the text. Results obtained on the Idis corpus (a bibliographic database of the British Library for Development Studies) with more than 80,000 records produce a top performance of 42.5% precision and 50% recall.

Cong Li and others [68] have proposed a mixed solution for m-class classification (not multi-labeling). In their work a Naïve Bayes network is used to extract relevant "keys". These keys are nothing but terms appearing in the summary (abstract) of the document that are weighted within the text using a probabilistic network. These keys are the input to a perceptron algorithm with margins (PAM) (although SVMs were also tested as classifier). Results obtained for e-mail categorization showed that about 20 to 30 keys produced best classification results: 63.6% of accuracy when considering only the top ranked class, and 81.5 % when using SVM and considering top 3 ranked labels). It could be an interesting model to be applied over large multi-label collections like ours.

B. Lausser and A. Hotho [61] have applied also Support Vector Machines for multi-label of multilingual documents using the AGROVOC thesaurus. The multi-label problem is transformed into m independent binary classification problems that are fed with a traditional $tf \cdot idf$ document vector. After running the SVMs, all classes with a score greater than a certain threshold were assigned to the document. Due to the nature of the thesaurus, a two-level categorization was applied. A break-even-point for precision and recall of 61% was obtained at 0.6 threshold (although they claimed best results with 0.1 threshold). Unfortunately the low number of documents involved (just 50) and some

other incongruities found in this article motivate us to take these results with caution.

It is relevant to point out how SVM scores are used for ranking labels in the two last works presented. SVM scores are related to margin distance; thus, this measure is a bad value about the relative proximity of a document to a certain class. SVM and other margin-based classifiers should not be considered as proper base classifiers when ranking over classes is applied for producing the final set of candidates.

9.2 The High Energy Physics domain

9.2.1 The AIR/PHYS System

The old AIR/PHYS system [16] was one of the first multi-label classifiers in production for real applications (a database on Physics papers). It is an application of the AIR/X system [43, 57] and many innovative concepts were implemented inside it like a basic segmentation with specific relevance calculation, search for certain relations between terms appearing in different parts of the text (title, abstract and content) and other techniques. But its complexity makes it difficult to apply it to other domains. In any case, despite the complexity of the indexing phase, it brings out the importance of knowledge for building real-world classifiers. The deep study on the characteristics of the collection where the classifier will act, along with a wide range of strategies combined for the classification task are references for the design and implementation of systems like AIR/PHYS, operating with direct practical application.

We can summarize the data flow in AIR/X in the following steps:

1. Identification of terms (i.e. single words and noun phrases)
2. Calculation of association value $z(t, c_i)$ between a term t and a class c_i as $z(t, c_i) = n_{ti}/n_t$, where n_{ti} is the number of documents with term t and class c_i and n_t is the number of documents of term t.
3. Apply thresholds c_1 and c_2 to discard z values that are too high or too low.
4. Computation of an indexing function $a(x)$ that selects from x (the ranked list of classes) the appropriate final ones. This is done using a supervised learning based classifier (ID3 as found in their descriptions).

AIR/PHYS is based on AIR/X, but goes beyond it, adding a more complex indexing approach (the *Darmstadt Indexing Approache* or DIA) that considers some segmentation weighting (whether term appears in title or abstract) and further specific corrections. An evaluation of the AIR/PHYS system [42]

showed the complexity of finding a good scheme of indexing due to the different domains involved in Physics. The best scheme produced 68% of recall and 57% in precision, which are still quite good values today. As a conclusion, we keep the need of deeper studies depending on the collection/application the system will work for.

9.2.2 Citometer

Again within the domain of High Energy Physics indexing we can find an interesting proposal for multi-labeling. This work of Ezhela and others [36] can be summarized in their statement: *one can attach to a new text meta-data of texts being quoted (cited) by it.* After reviewing the indexing process for HEP papers they reached two important conclusions. The first one is that to assign proper key words not only the text should be used, also the comprehension of cited texts is relevant to ensure completeness in the labels selected. The second point is that more precise indexing can be obtained by considering prominent relevance for articles containing numerical experimental data.

The automatic indexing *Citometer* works by applying one simple rule: assign all the labels of cited papers whenever they share with the paper being indexed further citation co-occurrences. Precision values range from 41% to 77% depending on what level of indexing we consider (they use the PACS thesaurus, which is a three-level rubricator).

Main issues can be summarized as follows:

1. Citometer idea works sufficiently well but
2. imprecise indexing schemes like PACS and DESY (different levels, many key words entries very similar) are the main obstacles for optimal performance,
3. not all documents showed an uniform way of storing the key words, and
4. not all documents are properly cited

We can consider that this source of information is very interesting and, in fact, it has inspired some of our approaches, in particular the use of meta-data for enhancing the classification process with this kind of available knowledge. In very technical domains like HEP, some works for creating these citation networks like the citation extractor used by the CERN Document Server [26] may become crucial resources in text classification tasks.

9.2.3 SPIRES-HEP

Of course, rule based approaches can be found in the literature. These techniques consists in the identification of certain patterns within the text and the

association of classes to each pattern or to a combination of them. One system has been developed for High Energy Physics using a so-called *Associative Patterns Dictionary* (APD) for the electronic papers stored in the database SPIRES-HEP (Stanford Public Information Retrieval System, HEP collection) [120]. The ADP parses the full-text of a paper to find associative patterns (chunks of text). When a pattern is matched the program returns the *key-chain*. A key-chain is a valid descriptor from the DESY [32] thesaurus and a pattern can be:

- A single word (*astrophysics, axiom, neutrino, ...*)
- An abbreviation (*GRB, QCD, ...*)
- A sequence of words or an entire sentence

Patterns can be restricted to certain parts of the text (which implies that a basic segmentation is performed) like the title or the abstract. Patterns can also be "negative" in the sense that instead of activating a certain key-chain, they can also force to ignore it. In figure 9.1 we have some examples of associative patterns.

pattern		**key-chain**
neutrino transitions with the photon and electron-positron pair creation	$\Longrightarrow$	`neutrino, transition`
magnetized WDs	$\Longrightarrow$	`astrophysics, white dwarf`
dispersion relation for longitudinal plasmons	$\Longrightarrow$	`plasmon, longitudinal`

Figure 9.1: Examples of associative patterns in an APD

It is very interesting that they have used almost the same collection of HEP papers, and exactly the same thesaurus, DESY; but unfortunately no evaluation of the system has been published, making it difficult to asses its excellence as automatic key-wording system. The current system does not consider the possibility of creating associative patterns with logical operators to enrich the expressiveness of a rule by combining different basic patters (using AND, OR, etc.). The main criticism for this system is that associative patterns are added *by hand*, turning the architecture into a too rigid solution making impossible to port the program to a domain different from HEP.

9.2.4 Sokrates

SOKRATES stands for *Self-organizing Object-oriented Key-term Recognition And Text-Editing System*. It is a set of programs developed by Ivo Steinacker

from Innsbruck (Germany). These system is included here because of the interest CERN showed in it at an early stage [30]. The information about the system is not very accessible, and most of its architecture and algorithms remain behind a dark curtain.

There is a first learning phase in which a fair amount of similar text material is used to train the system, constructing a set of longest possible noun-type phrases. The SOKRATES program works more or less automatic, and does not require experts to read the articles to be key worded. So such a system is from the beginning completely autonomous, it is not intended to be a tool to aid indexers.

An objective evaluation of the system is difficult, since proposed key words are compared manually with DESY key words and those similar (or, at least, considered to be similar) are counted as matches. The value of precision found was near to 40%, but experiments were, in our opinion, not cleanly performed.

9.2.5 HEPindexer

The *HEPindexer* project intended to propose a first solution to automatic classification at CERN, opening a door to research for automatic indexing tools in the scientific area of HEP. This project has been developed at the ETT division of CERN, under the supervision of Jean-Yves Le Meur and Jens Vigen, by myself. We can consider that HEPindexer is the predecessor of the TECAT system, the result of this thesis [82].

The goal of the training process is to build the *WTK* matrix. This matrix relates two items: key words and terms. These items are related by a common weight which is calculated from documents in the training set. These weights are computed from training documents in a purely statistical way. The matrix is the result of the combination of two previous matrices: *WTD* and *WKD*.

The **WTD matrix** relates terms and documents. Each cell in the matrix is a weight $wtd_{t,d}$ which establishes a relation for term t to document d. These values are computed after apply to the document a *conflation* algorithm. This algorithm takes the full text document as input and generates a list of stemmed words which will be ranked in frequency and normalized as is detailed in next sections. Its values are computed by equation 9.2.

$$wtd_{t,d} = \frac{freqtd_{t,d}}{maxtfreq_d} \cdot \log \frac{N}{N_t} \tag{9.2}$$

where

$wtd_{t,d}$	weight between term t and document d
$freqtd_{t,d}$	frequency of term t in document d
$maxtfreq_d$	maximum frequency of a term in document d
N	total number of documents in the collection

N_t total number of documents containing term t

The **WKD matrix** comes from a similar process. Here, the list of predefined key words for the document are processed: primary key words are extracted and weighted. Its values are computed as detailed in equation 9.3.

$$wkd_{k,d} = \frac{freqkd_{k,d}}{maxkfreq_d} \tag{9.3}$$

where
$wkd_{k,d}$ weight for main key word k in document d
$freqkd_{k,d}$ frequency of main key word k in document d
$maxkfreq_d$ maximum frequency of a key word in document d

The calculation to obtain the weight which relates a term and a key word is quite simple since it is fully based on statistics. Equations 9.4 and 9.5 show how each value $wtk_{t,k}$ of the WTK matrix is computed from values $wtd_{t,d}$ and $wkd_{k,d}$ of matrices WTD and WKD respectively.

$$wtk_{t,k} = \sum_{d=1}^{N} wtd_{t,d} \cdot wkd_{k,d} \tag{9.4}$$

$$wtk_{t,k} = \log \frac{M}{M_t} \cdot \sum_{d=1}^{N} wtd_{t,d} \cdot wkd_{k,d} \tag{9.5}$$

where
N number of documents in the training collection
M number of different main key words in the training collection
M_t number of different main key words related to the term t
$wkd_{k,d}$ weight between a key word k and a document d

The equation 9.4 is very simple to compute, but the equation 9.5 should return better results. The reason we use two different approaches here is because it is proved that inverse document frequency improves retrieval when used to compute the weight of terms in documents, but we introduce here a measure we call *inverse key word frequency* (IKF). This second option will assign lower weights to those terms that are related to many key words.

Once the WTK matrix is computed we have, for example, the term *acceler* (which is a stemmed word) connected to the key word *electron muon* with a weight of 0.34 and the term *collision* related to same key word with a weight of 1.76 we can conclude that *collision* has more to do with *electron muon* than *acceler*. Following this reasoning we can propose key words starting from a

list of terms obtained from a new incoming document. The document must be preprocessed also, to have the list of its unique stemmed words properly weighted.

This system is similar to the one proposed by Sheridan and Schuble [115], as we mainly perform a "dual" operation between two retrieval systems, associating descriptors (classes) and terms via the document nexus, instead of terms from two different languages as in the work mentioned.

Best performances are obtained when we use *inverse key word frequency* (IKF) combined with the normalization of key word frequency. We can also find that we can get better results with sort size for the vector of terms for documents. This also improves retrieval speed (5-10 seconds on a Pentium III running at 800 megahertz with 128 megabytes of RAM memory). The best configuration obtained after experiments gave a precision of 52.7% and a recall value of 58.5%.

9.3 Some remarks

We have started this chapter putting some emphasis on the lack of multi-label classifiers. As we have seen, relevant work for the MeSH thesaurus has been carried out, though little or no work has been done in other domains. An additional crucial point is the handicap imposed by the size of the thesaurus each system tries to automate: it is not the same to produce three or four labels for a document from a set of thousands of potential ones than proposing the same number of labels from a lower cardinality, like few hundreds. Therefore, we must look at measurements in the light this fact. For instance, the Nomindex tool introduced for the MeSH thesaurus reached 30% in precision and 31% in recall. Those final results are very good values, taking into account that more than 22,500 concepts were handled.

Chapter 10

TECAT: the Text Categorization Toolkit

10.1 Introduction

For our experiments, we have built our own classification system as demanded by the following main requirements at the CERN library:

1. Speed: the system must return an assignment of classes to a document within less than ten seconds as limit for a "real time" response.
2. Modularity: the system must be able to grow by adding new binary base classifiers.
3. Experimentation: the system not only must run in a production mode, but also allow to play with different parameters in order to find the most suitable configuration for a given corpus.
4. Portability: the system must run on a variety of possible architectures: Windows, UNIX platforms, etc.

The program is written in ANSI C and makes use of the GNU GLib library. It is called TECAT.

TECAT stands for TExt CATegorization. It is a tool for creating multi-label automatic text classifiers. With TECAT you can experiment with different collections and classifiers in order to build a multi-labeled automatic text classifier.

When indexing by assignment we know a priorys the set of indexes that we can use to label a document. This set is usually known as the controlled vocabulary or thesaurus. The assignment task has been performed traditionally by

professional experts who have read the text and selected the most representative "key words" (also known as "descriptors", "indexes", "categories" or "topics") for that text.

Indexing by assignment has been studied for automatic computer processing by the Automatic Text Categorization (TC) research community. Within this area, Information Retrieval (IR) and Machine Learning (ML) techniques work together to create computer based systems able to achieve this task with reasonable success.

From Information Retrieval we borrow the document processing (parsing, stemming, lemmatization, stop words removal, vector space model representation, similarity measures, feature identification, extraction, transformation and weighting, performance measures, etc.), and from Machine Learning its Supervised Learning algorithms (Rocchio, Neural Networks, K-NN, Genetic Algorithms, Support Vector Machines, Bayesian Networks, etc.). All these techniques are combined to end up with a system architecture that will allow a computer to perform automatic indexing by assignment.

10.2 Architecture

Even though we can find many different systems for text classification, we can mostly agree that the main phases performed by a TC system are the following, though the specific implementation of each one may vary:

1. *Collection preparation.* Since we are in a Supervised Learning approach, we have to let our system "learn" from training data. Therefore, a collection of already indexed documents must be provided.

2. *Document representation.* These documents are then processed and converted into internal models, which are representations that will allow learning algorithms to work properly. Usually this phase implies many other transformations (those ones that we have identified as IR techniques). The result will be a list of pairs <*feature*, *weight*> per document.

3. *Classifiers learning.* For each class we train a binary classifier using documents labeled with such a class as positive samples and the rest as negative samples. This is know as the one-against-all approach, the basic approach used by TECAT. After the training, a classifier per class is ready.

4. *Testing/classification.* Once the system is trained, we can either perform an automatic classification of new incoming documents or use an already-labeled collection to test the performance of the trained system.

How the collection is divided into the three different subsets: *training* (generating candidate classifiers), *validation* (selecting best classifiers and parameters

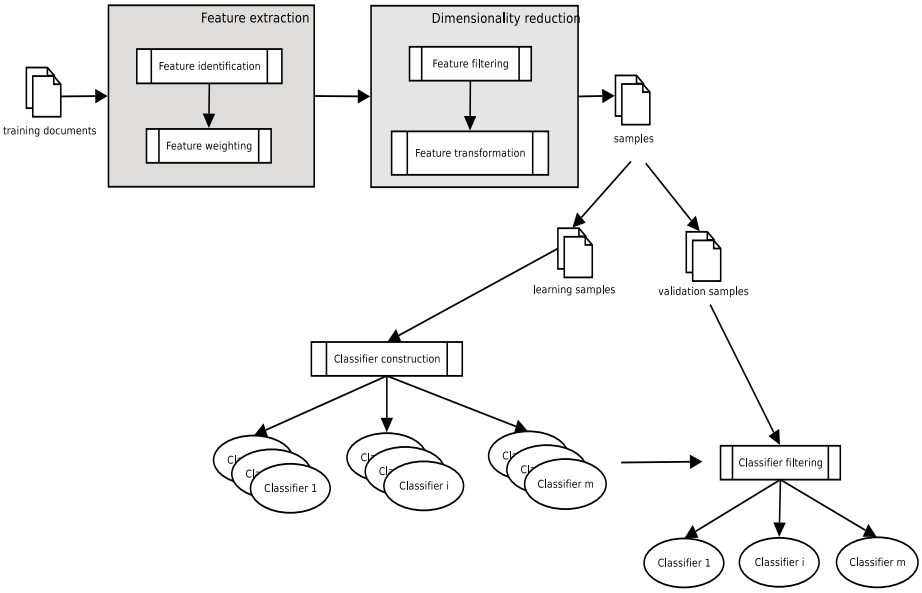


Figure 10.1: Training process in TECAT.

adjusting) and *testing* (computing performance measures) is detailed in chapter 11. In figure 10.1 the steps performed in training the system are shown. As we can see, the training samples are themselves divided into two disjoint sets used to train and select base classifiers.

10.3 Collection preparation

TECAT needs a collection of already-labeled documents. These documents must be organized in two different directories: the one with full text data and the one where key words are stored. A document is therefore represented by two files with the same name at these two different directories. So, if we have a training collection of 1,000 documents, we will have a directory with 1,000 text files and another one with 1,000 files containing the key words (one per line) related to the text files.

TECAT will start processing the text and key word documents. In this phase already some text processing and feature handling is performed:

- *Stemming.* The Porter Stemmer algorithm is used [93].
- *Minimal term frequency filter.* Those terms appearing less than a certain number of times in the document will be discarded.

- *Minimal and maximal term lengths.* Sometimes, too long words appear due to corruption in the text file after transformations for producing plain text version from PDF or PS documents.
- *Stop words removal.* Words appearing in the specified stop words list are ignored for later processing (they are just removed from document representation).

The quality of such methods for our specific collection have been studied in experiment 12.4

10.4 Document representation

Although, as we have already seen in the previous section, some processing is performed during collection digestion, additional filtering and weighting the features are performed in this phase. It is know as the "folding" phase in TECAT because the tool implements cross-fold validation, and, therefore, the weighting of features and other operations cannot be performed before knowing the split of documents into training, validation and test sets:

- *Training set*: these documents are used for training the classifiers.
- *Validation set*: these documents are used for discarding invalid classifiers and/or filter them if their performance is not good enough according to the specified criteria.
- *Test set*: these documents are used to compute the final performance of the system.

Thus, these sets are produced in N different partitions (N being the number of folders used). At every turn (N in total) of the folding process, just one fold is used as test set, 1/3 of remaining folds as validation set, and the rest (2/3 of folds) is used as training set.

Filtering operations performed at this phase are:

- *Document frequency feature selection.* We can discard those terms appearing in less than a given number of documents in the training set (at least in one, by default).
- *Document frequency class selection.* It is clear that when a class is not represented in the three sets, we cannot train it nor test it. Therefore, we can discard that class specifying an small threshold. But we can also make it larger, since rare classes will be difficult to be trained for. The default value is 1, which means that if the class is not in the training set it will be ignored for training (but not for testing, since we may want to see how good the system is considering even discarded classes).

- *Information gain filtering* (see section 5.1.2). TECAT will take by default the 50,000 features with the highest information gain value.

We have studied how this filtering affects our collection and which measure is the best in our case (see section 12.5).

Once terms and classes have been filtered, the next step will assign to terms (a.k.a features) a weight at every document. There are several weighting schemes available in the literature. TECAT offers two of them: the entropy weighting and the TD.IDF weighting [23].

- *Entropy weighting*. The global weighting is done by using term entropy. See section 4.2.1 for a detailed description of this weighting scheme.
- *TF.IDF*. The classical *term-frequency* · *inverse-document-frequency* weighting scheme is used. See section 4.2.2 for further details.

Weights can be normalized in the vector using the cosine normalization. It is performed by default since it has reported usually good performance as it controls the heterogeneity in document sizes. See experiment 12.6 for a study on the behaviour of these two weighting schemes in our collection.

The number of folds is set to 10 by default. A value of 10 means that 10 different partitions into train, validation and test sets of the collection of documents will be done and that, when testing, 10 tests will be launched, one per fold, averaging final measurements.

10.5 Classifiers learning

This is the most complex phase in the training. Here we will generate a model for each class. This model will allow us to predict if the class has to be assigned to a new document or not. In order to do so, we apply for every class a training process based on binary learning algorithms, using documents belonging to the class as positive samples and the rest as negative samples. This is known as the one-against-all approach (see section 6.1).

The training phase is, itself, divided into two sub-phases: training and validation. We can tell TECAT to try with a list of possible algorithms to be trained with the training set and then select the best one based on a given performance measure over a validation set. TECAT has some classifiers built-in, but it allows you to plug-in external classifiers easily (provided they understand TECAT sample format).

When a binary classifier is trained a model is generated for the algorithm used (usually a vector of weights and a bias value). For a new document vector, the classifier will return a value as result of a calculation between the document vector and the class model. This value represents, usually, the proximity of the

document to the class, so in order to decide if the document belongs to the class or not a threshold is also learned. Values returned above that threshold are considered positive answers (the document belongs to the class), and values equal or below that threshold are understood as that the class should not be assigned to the document.

In figure 10.1 the process is clarified: the training collection is split into training samples and validation samples. The first subset is used to train k base classifiers for every class. From those classifiers, by using the validation subset, only one classifier will be selected as the base classifier for that class. This kind of voting system (where only one classifier ends with all votes) allows the system to have a totally different algorithm per class and select it from a set of candidates according to its performance for that class. At the end, the number of base classifiers will be less or equal to the number of trained classes (less in the case none of the candidates is able to produce a minimal performance for some classes). This *adaptive* selection of classifiers is tested in experiment 12.2 and is proven to be interesting for large multi-labeling problems as studied in experiment 12.7, introduced also by [88].

10.5.1 Specifying base binary classifiers

Classifiers implemented built-in are: Rocchio, Widrow-Hoff, Exponentiated Gradient (see [67] for these three methods), LVQ [77] and PLAUM [127]. But also external classifiers can be specified. For example, in our experiments we have used the SVM-Light[1] [52] package as an external (not built-in) classifier. As many classifiers as wanted can be used simultaneously: different algorithms and/or with different parameters.

As a sample of invocation to this complex operation, we show here what could be a command line syntax of a training in TECAT:

```
$ tecat --train-by-class                                          \
  --selection-threshold=0.1                                       \
  --classifier=method=plaum:pos_tau=2.0:neg_tau=-1.5:iter=100 \
  --classifier=method=plaum:pos_tau=1.0:neg_tau=-1.0:iter=10  \
  --classifier=method=rocchio                                     \
  --classifier=method=rocchio:beta=2.0                            \
  --compute-scut                                                  \
  /data/tecat-trained
```

10.5.2 Specifying selection measures

In the example given above two different classifiers with two different parameter set each are given, which turn out to be four different classification schemes to

[1] Available from http://svmlight.joachims.org/

be trained and evaluated. From those, the most performing classifier for the category will be select. This performance is computed on a certain measure. TECAT computes precision, recall, F1, F, accuracy, error, fallout and optimal distance as performance measures (see chapter 7). By default, the F1 measure is used to select the best classification approach. Also, once one classifier is selected as the best one, it will be assigned to the class only if it reaches a minimal threshold on the performance measure selected. By default that threshold is 0.1, so the best one must, at least, reach 0.1 for the F1 value measured. This threshold is very important since it will allow discarding non-trainable classes (or, at least, those that are difficult to train) reducing considerably the number of classifiers taken into account (see experiments 12.7).

10.5.3 Computing the S-Cut threshold

Another threshold to be taken into account is the one that determines whether a samples is assigned to a class or not based on the value returned by the classifier. By default this threshold is zero when working with margin based classifiers (SVM and PLAUM), since negative samples are passed to the learning algorithm labeled with -1 weight and positive ones with $+1$ weights. But some distance-based algorithms like Rocchio and Widrow-Hoff need to tune that threshold to the value that maximizes the performance of the classifier. Again, the selection measure is used as value to find that optimal threshold on the validation set. The algorithm used is the *S-Cut* approach (see section 6.7). By default, the threshold remains to zero, so if we want to apply S-Cut to compute that threshold we will specify it explicitly.

A recommendation is not to use S-Cut when working with margin-based classifiers like PLAUM or Support Vector Machines. Since these algorithms are optimized to find the widest margin between positive and negative samples, the value returned (the margin) is not really a measure of the distance of the sample to the class. Applying S-Cut does not report significant improvement in general as found in our experiments (see section 12.7).

10.6 Testing/classification

Once the system has been trained, it can assign labels automatically to new plain text documents, following the steps shown in figure 10.2.

TECAT produces a lot of verbose output during testing. At each fold, values for different measures for the most frequent classes are given and the values of the averaged measures for the fold are shown. At the end, the averaged values of fold measures are computed. It is important to understand this output, as they are the data used to establish conclusions in our experiments.

In figure 10.3 we can see an example showing the last fold results and the

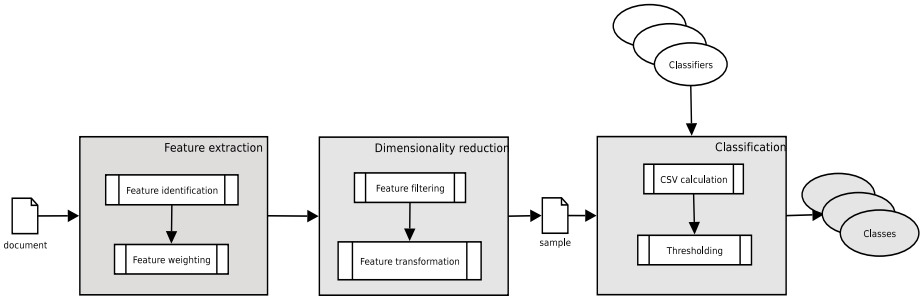


Figure 10.2: Classification process in TECAT.

average results over all folds after test processing.

```
[...]
------------------------------------------------------------------------------------------------------------ (Fold  3, 79 docs)
BY CLASS TP     TN     FP     FN  Precision    Recall  Accuracy     Error   Fallout      Beta         F        F1   Opt.Dist.      BEP
avg)    340   8245   2060    336   0.141667  0.502959  0.781805  0.218195  0.199903  1.000000  0.221066  0.221066   0.298649  0.322313
AVG)      0      0      0      0   0.112081  0.274848  0.781805  0.218195  0.210595  1.000000  0.102058  0.102058   0.148477  0.193465
BY DOC   TP     TN     FP     FN  Precision    Recall  Accuracy     Error   Fallout      Beta         F        F1   Opt.Dist.      BEP
avg)    340   8245   2060    336   0.141667  0.502959  0.781805  0.218195  0.199903  1.000000  0.221066  0.221066   0.298649  0.322313
AVG)      0      0      0      0   0.150311  0.495448  0.781805  0.218195  0.199907  1.000000  0.214758  0.214758   0.287987  0.322880
139 classes tested, 63 classes where successfully trained
10 Most frequent classes:
-------------------- 'experimental results' (in 161 docs)
         49      1     29      0   0.628205  1.000000  0.632911  0.367089  0.966667  1.000000  0.771654  0.771654   0.737101  0.814103
-------------------- 'magnetic detector' (in 133 docs)
         43      8     27      1   0.614286  0.977273  0.645570  0.354430  0.771429  1.000000  0.754386  0.754386   0.726786  0.795779
-------------------- 'electron positron' (in 87 docs)
         10     53      0     16   1.000000  0.384615  0.797468  0.202532  0.000000  1.000000  0.555556  0.555556   0.564857  0.692308
-------------------- 'talk' (in 58 docs)
         16      4     59      0   0.213333  1.000000  0.253165  0.746835  0.936508  1.000000  0.351648  0.351648   0.443743  0.606667
-------------------- 'quark' (in 45 docs)
         15      2     60      2   0.200000  0.882353  0.215190  0.784810  0.967742  1.000000  0.326087  0.326087   0.428230  0.541176
-------------------- 'quantum chromodynamics' (in 43 docs)
          4     66      2      7   0.666667  0.363636  0.886076  0.113924  0.029412  1.000000  0.470588  0.470588   0.492029  0.515152
-------------------- 'anti-p p' (in 43 docs)
          8     52     15      4   0.347826  0.666667  0.759494  0.240506  0.223881  1.000000  0.457143  0.457143   0.482099  0.507246
-------------------- 'jet' (in 42 docs)
          9     49     21      0   0.300000  1.000000  0.734177  0.265823  0.300000  1.000000  0.461538  0.461538   0.505025  0.650000
-------------------- 'DESY HERA Stor' (in 40 docs)
         10     58      4      7   0.714286  0.588235  0.860759  0.139241  0.064516  1.000000  0.645161  0.645161   0.645611  0.651261
-------------------- 'mass spectrum' (in 38 docs)
         13      4     62      0   0.173333  1.000000  0.215190  0.784810  0.939394  1.000000  0.295455  0.295455   0.415458  0.586667
------------------------------------------------------------------------------------------------------------------------------------
FINAL RESULTS
139.00 classes tested, 60.67 classes where successfully trained
BY CLASS TP     TN     FP     FN  Precision    Recall  Accuracy     Error   Fallout      Beta         F        F1   Opt.Dist.      BEP
avg)   1086  23190   6676   1018   0.139912  0.516160  0.759337  0.240663  0.223532  1.000000  0.220150  0.220150   0.302199  0.328036
AVG)      0      0      0      0   0.106680  0.288463  0.758499  0.241501  0.236860  1.000000  0.095578  0.095578   0.148598  0.197572
BY DOC.  TP     TN     FP     FN  Precision    Recall  Accuracy     Error   Fallout      Beta         F        F1   Opt.Dist.      BEP
avg)   1086  23190   6676   1018   0.139912  0.516160  0.759337  0.240663  0.223532  1.000000  0.220150  0.220150   0.302199  0.328036
AVG)      0      0      0      0   0.134365  0.500749  0.758499  0.241501  0.224547  1.000000  0.203454  0.203454   0.281297  0.317557
```

Figure 10.3: Sample output of TECAT test results

For a better understanding of the evaluation measures involved here, please refer to chapter 7. Anyhow, we summarise here those measures detailing how they are represented in the sample output given at figure 10.3:

- The TP, TN, FP and FN are the 'true positive', 'true negative', 'false positive' and 'false negative' counts. So, for each decision taken by TECAT (assign a class or not) we increase one of these counters by comparing to classes assigned to documents by human experts.

- Each evaluation measure appears, in final results, four times: two when evaluating by class, and another two when evaluating by document. When evaluating by class, we are computing TPs, TNs, FPs and FNs for every document at each class tested. When evaluating by document these values are computed for every class at each document tested. Macro-averaged measures (AVG) and micro-averaged measures (avg) are calculated from those values. We can see that, as stated in chapter 7, micro-averaging is the same for both evaluation approaches.

- Evaluation measures are: precision, recall, accuracy, error, fall-out, F and F1 (the last two share the same value in the above example because *beta* was set to value of 1), optimal distance and break-even-point (BEP). This last value is computed by TECAT as a simple mean between precision and recall:

$$BEP = \frac{precision + recall}{2} \tag{10.1}$$

- In the training phase, some classes are not trained because the classifier did not produced good performance (see section 10.5.1). In that case, we cannot decide whether to assign it or not to a certain document. TECAT just considers that non-trainable classes should never be assigned, and its effects are tested counting FPs and FNs for those classes. The number of tested classes is the number of classes that, after filtering in the indexing and folding phase, were passed to the learning phase. Thus, in the given example, 139 different classes were considered in the test, though TECAT produced a classifier for only 60.67% of classes in average (averaged over folds).

10.7 Parameters overview

This last section in the chapter describes the convention used for naming different parameters involved in a TECAT experiment. These names will be used when explaining each experiment configuration in next chapters. Following there is the name of the parameter along with its meaning organized into the four phases involved in a TECAT experiment: indexing, folding, learning and testing.

1. **Indexing**. This phase corresponds to the processing described at section 10.2.

   `stop-words` It determines whether a stop words list has been used to filter out mean-less words. Its value is, therefore, `yes` or `no`.

   `stemming` This flag enables or disables the application of the Porter's stemmer. Its values are, also, `yes` or `no`.

`min. term freq.` This is an integer number for filtering those terms with a frequency higher than the value specified.

`min. term length` When the number of characters of a term is larger than this value, the term is not indexed.

`max. term length` Similarly, terms with less characters than the specified value are discarded.

2. **Folding**. This stage corresponds to the process described in section 10.3

`DF term filter` This integer value is used to filter out those terms appearing in less than the given number of documents. Thus, a term must have, at least, the specified document frequency at the three subsets: training, validation and testing of the cross-validation process.

`no. folds` This is the number of folds to be created for cross validation.

`IG filter` This is the number of terms with the highest information gain value that will be kept.

`DF class filter` This is, as for terms, a filter to discard classes with a lower document frequency than the given value.

`normalization` This flag determines whether term weights in a document vectors will be normalized using the cosine normalization (to force all vectors having the same norm). Its possible values are `yes` or `no`.

`term weighting` This parameter establishes the weighting scheme to be used for terms in documents. It can be either `TF.IDF` for the classical $term - frequency \cdot inverse - document - frequency$, or `entropy` for using the entropy weighting formula.

3. **Learning**. This stage corresponds to what is described in sections 10.4, 10.5, 10.5.1 and 10.5.2.

`S-cut` This is a flag that enables/disables (value `yes` or `no`) the computation of a threshold using the S-cut approach.

`measure` This is the evaluation measure that will be used to select the best binary classifier for each class. We usually use `F1` as the discriminative measure.

`threshold` This real parameter in the interval [0,1] determines the minimal value that the former measure must reach in order to consider the classifier as candidate.

`methods` This is the list of methods (one or more) that will be evaluated for every class to select most suitable one.

4. **Testing**. This operation is described in section 10.5.3

`mode` This parameter just specifies the classification approach used. If its value is `by rank`, classification status values will be ranked and the n top classes will be selected. If the value is `by class`, then all classes with positive status value will be assigned.

`n-top` When using the `mode` *by rank* this is the number of classes selected from the top of the ranked list.

An example of configuration is shown in table 10.4. Tables like this one will be included in the description of every experiment performed. Now that the parameters have been detailed, we can describe precisely each of the TECAT runs. In appendix B we can find how these parameters are passed to TECAT when invoking it from the command line.

indexing		**folding**	
stop-words	*yes*	DF term filter	*1*
stemming	*yes*	no. folds	*10*
min. term freq.	*0*	IG filter	*50000*
min. term length	*0*	DF class filter	*1*
max. term length	*40*	normalization	*yes*
		term weighting	*TF.IDF*

learning		**testing**	
S-cut	*yes*	mode	*by class*
measure	*F1*		
threshold	*0.1*		
methods	*rocchio*		

Figure 10.4: Sample parametrization of TECAT for a potential experiment

Chapter 11

The HEP collection

11.1 Overview

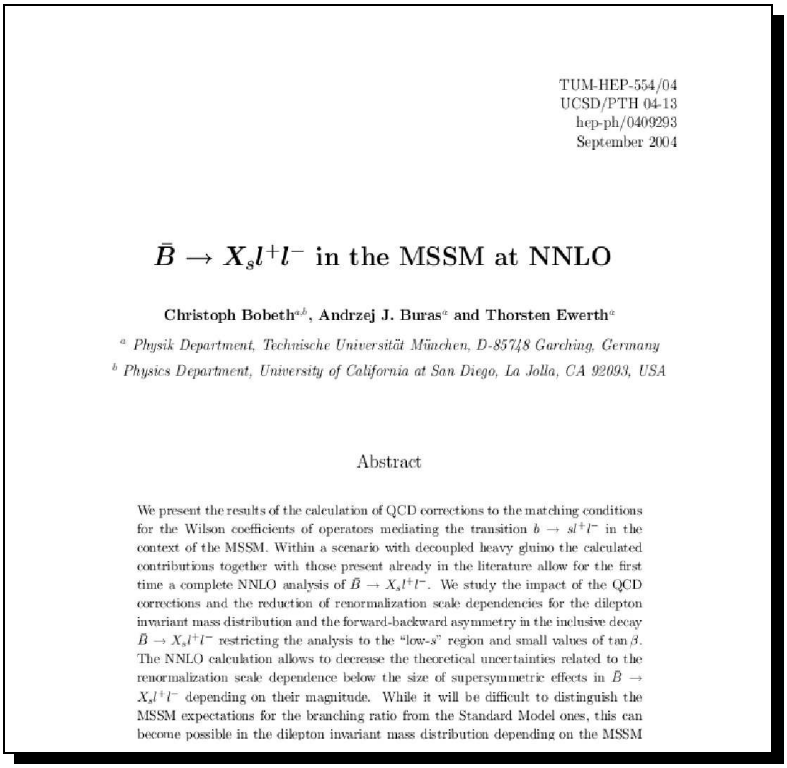
In this thesis, we focus on a collection of High Energy Physics papers[1]. The Articles & Preprints collection aims to cover as far as possible the published and pre-published literature in particle physics and its related technologies. The collection contains something like 400,000 documents, out of which about 50 % can be accessed electronically. The documents originate from articles published in journals, preprints, technical reports, conference presentations, scientific committee documents and theses; all are comprehensively indexed by the CERN Scientific Information Service. The collection starts, comprising only the most important documents in the first decennia, from the mid of the 19th century. The full coverage starts from 1980 onwards. Today, more than 320,000 documents are available on the CERN Document Server (CDS).

The HEP papers are technical papers that usually contain plenty of graphs, diagrams, formulas and other specific scientific notations. Their vocabulary should have, in our opinion, a low degree of ambiguity, due to the specialization of texts in every area (mainly Astrophysics, Theoretical Physics and Experimental Physics). This is, of course, an important advantage, but it is also true that the lack of specific resources (lemmatizers, stemmers, stop-word lists, etc.) will be a main drawback in our system. Therefore, some experiments oriented toward testing the validity of the resources used have been performed.

[1] These documents are freely available from the CERN Document Server (CDS) at `http://cds.cern.ch`

11.2 Data format

In figure 11.1 we can see a sample of the first page of a typical HEP paper. We already identify some special symbols in the title. The collection we have used in the present work consists on 22,903 labeled documents (6,1 gigabytes) converted into plain-text using the *pdftotext* tool included in the *xpdf* package (version 2.01)[2]. The files containing the key words (labels) for that set of papers represents 112 megabytes of files.



TUM-HEP-554/04
UCSD/PTH 04-13
hep-ph/0409293
September 2004

$\bar{B} \to X_s l^+ l^-$ in the MSSM at NNLO

Christoph Bobeth[a,b], **Andrzej J. Buras**[a] **and Thorsten Ewerth**[a]

[a] *Physik Department, Technische Universität München, D-85748 Garching, Germany*

[b] *Physics Department, University of California at San Diego, La Jolla, CA 92093, USA*

Abstract

We present the results of the calculation of QCD corrections to the matching conditions for the Wilson coefficients of operators mediating the transition $b \to s l^+ l^-$ in the context of the MSSM. Within a scenario with decoupled heavy gluino the calculated contributions together with those present already in the literature allow for the first time a complete NNLO analysis of $\bar{B} \to X_s l^+ l^-$. We study the impact of the QCD corrections and the reduction of renormalization scale dependencies for the dilepton invariant mass distribution and the forward-backward asymmetry in the inclusive decay $\bar{B} \to X_s l^+ l^-$ restricting the analysis to the "low-s" region and small values of $\tan\beta$. The NNLO calculation allows to decrease the theoretical uncertainties related to the renormalization scale dependence below the size of supersymmetric effects in $\bar{B} \to X_s l^+ l^-$ depending on their magnitude. While it will be difficult to distinguish the MSSM expectations for the branching ratio from the Standard Model ones, this can become possible in the dilepton invariant mass distribution depending on the MSSM

Figure 11.1: Sample for a High Energy Physics document.

No special hardware has been provided for developing the automatic classifier. A Pentium IV with 512 megabytes of RAM has been the hardware used for implementation and experimentation. It is important to note that the size of the collection described above and the expected number and variety of experiments performed would have involved prohibitive times, so we considered the creation of a partition for experimental matters and leave the complete one to test the final system built up with those algorithms that performed best on the smaller collection. This collection was obtained by keeping only experimental

[2]The *xpdf* package is available from http://www.foolabs.com/xpdf/

HEP papers. We call this collection the *hep-ex* partition. This partition contains 2967 documents and 2793 main key words (1103 if energy related keys are removed, and 825 if also without reaction related ones). Full-text papers from this partition produce more than 300,000 features after stemming is applied and stop words removed, so we can see that the high dimensionality of the data will demand a method to decrease this number if our intention is to run supervised learning algorithms in a reasonable computing time. Another possibility to accelerate the experimentation is to consider, instead of the full-text document, just the "abstract" of it. Any HEP paper comes with an section where authors summarize the content of the text, this is the abstract and its size varies from a few lines to almost one page of text. Some experiments have been carried out on additional data, like the full-text data of the *hep-ex* partition and the whole HEP collection available.

An interesting characteristic of the data stored in the CDS server is its additional information: the *meta-data*. This information plays a main role in the purpose of our study. Meta-data is data about data, or information known about the document in order to provide access to the document. Meta-data Usually includes information about the intellectual content of the document, digital representation data, and security or rights management information. Information of this kind becomes very important in specialized libraries, since it can act as an index to documents. Meta-data allows users to browse documents by author, date, format and further useful information. The meta-data can also be used as a nexus between documents and whole libraries, it has represented traditionally the information that can form a query and, now that digital libraries are becoming as important as hard-copy based ones, the interoperability of information and exchange of data can be determined by the compatibility of meta-data available between two libraries, as the Dublin Core Meta-data Initiative promotes[3].

From CDS we can obtain the meta-data in XML format, either in *Dublin Core format* (see figure 11.2) or in *MARC format* (see figure 11.3. In the HEP collection we have considered the following fields among all the available meta-data:

- **Creator**. The author or list of authors of the document.
- **Title**. String containing the title of the paper.
- **Subject**. Area where the paper fits in: Experimental Physics, Theoretical Physics or Astrophysics.
- **Date**. Date of publication

[3]The Dublin Core Meta-data Initiative is an open forum engaged in the development of interoperable on-line meta-data standards that support a broad range of purposes and business models. DCMI's activities include consensus-driven working groups, global conferences and workshops, standards liaison, and educational efforts to promote widespread acceptance of meta-data standards and practices

- **Description**. A summary of the content (also known as *abstract*).

```
<?xml version="1.0" encoding="UTF-8"?>
<collection>
    <dc xmlns="http://purl.org/dc/elements/1.1/"
          xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
            xsi:schemaLocation="http://purl.org/dc/elements/1.1/
                                http://www.openarchives.org/OAI/1.1/dc.xsd">
        <language>eng</language>
        <creator>Bassett, B A</creator>
        <creator>Liberati, S</creator>
        <creator>Molina-Paris, C</creator>
        <creator>Visser, M</creator>
        <title>Geometrodynamics of Variable-Speed-of-Light Cosmologies</title>
        <subject>Astrophysics and Astronomy</subject>
        <identifier>http://preprints.cern.ch/id=0001441</identifier>
        <description>This paper is dedicated to the memory of Dennis
        Sciama. Variable-Speed-of-Light (VSL) cosmologies are
        currently attracting great interest as an alternative to
        inflation. We investigate the fundamental geometrodynamic
        aspects of VSL cosmologies and provide several implementations
        which do not explicitly break Lorentz invariance (no ‘‘hard’’
        breaking), and answer the question ‘‘VSL with respect to
        what?’’. This large class of VSL cosmologies are compatible
        with both classical Einstein gravity and low-energy particle
        physics. These models solve the ‘‘kinematic’’ puzzles of
        cosmology as well as inflation does, but cannot by themselves
        solve the flatness problem, since in their purest form no
        violation of the strong energy condition occurs. We also
        consider a heterotic model (VSL scalar field chi plus inflaton
        field phi) which provides a number of observational
        implications for the low-redshift universe if chi contributes
        to the ‘‘dark energy’’ either as CDM or quintessence. These
        implications include modified gravitational lensing,
        birefringence, variation of fundamental constants, and
        rotation of the plane of polarization of light from distant
        sources.</description>
        <date>2000-01-26</date>
    </dc>
</collection>
```

Figure 11.2: Sample for meta-data information in XML format.

Some experiments have been already carried out using this collection ([83, 86]), and its interesting distribution of classes allows us to carry out a number of experiments and to design new approaches. An analysis of the collection has shown that there is the typical high level of imbalance among classes. If a given class is rarely represented in a collection, we can intuitively foresee a biased training that will yield classifiers with a low performance. It is clear that, if the collection were perfectly balanced, we could expect better categorization results, due to better learning.

The architecture chosen for the multi-label classifier can be viewed as a *comittee* with *weighted linear combination* of binary classifier where one of the

classifiers has a weight of one and the rest is zero, i.e. at the end, only one classifier is used for generating the *classification status value* (CSV), but the advantage is that we can have a different classifier for each class. In order to select the best classifier for the class, the training collection has to be divided into *learning* and *validation* subsets: the former to train the classifiers and the latter to select most performing classifier for each class. Figure 11.4 shows how the different sets are used at different stages within our classification architecture.

```
<?xml version="1.0" encoding="UTF-8"?>
<collection xmlns="http://www.loc.gov/MARC21/slim">
<record xmlns="http://www.loc.gov/MARC21/slim">
   <controlfield tag="001">826204</controlfield>
   <controlfield tag="003">SzGeCERN</controlfield>

   <datafield tag="037" ind1="" ind2="">
     <subfield code="a">cond-mat/0503152</subfield>
   </datafield>
   <datafield tag="041" ind1="" ind2="">
     <subfield code="a">eng</subfield>
   </datafield>
   <datafield tag="100" ind1="" ind2="">
     <subfield code="a">Kitamura, T</subfield>
   </datafield>
   <datafield tag="245" ind1="" ind2="">
     <subfield code="a">Theory of liquid-glass transition in water</subfield>
   </datafield>
   <datafield tag="260" ind1="" ind2="">
     <subfield code="c">2005</subfield>
   </datafield>
   <datafield tag="269" ind1="" ind2="">
     <subfield code="c">7 Mar 2005</subfield>
   </datafield>
   <datafield tag="300" ind1="" ind2="">
     <subfield code="a">26 p</subfield>
   </datafield>

   <datafield tag="520" ind1="" ind2="">

   <subfield code="a">A quantum field theory of the liquid-glass
   transition in water based on the two band model in the harmonic
   potential approximation is presented by taking into account of the
   hydrogen bonding effect and the polarization effect. The sound and
   diffusion associated with intra-band density fluctuations, and the
   phonons and viscocity associated with inter-band density
   fluctuations are calculated. The Kauzmann paradox on the Kauzmann's
   entropy crisis and the Vogel-Tamman-Fulcher (VTF) law on the
   relaxation times and the transport coefficients are elucidated from
   the sound instability at a reciprocal particle distance
   corresponding a hydrogen bond length and at the sound instability
   temperature very close to the Kauzmann temperature. The gap of
   specific heat at the glass transition temperature and the boson
   peaks are also presented.</subfield> </datafield>

   <datafield tag="650" ind1="1" ind2="7">
     <subfield code="2">SzGeCERN</subfield>
     <subfield code="a">Condensed Matter</subfield>
   </datafield>

   <datafield tag="695" ind1="" ind2="">
     <subfield code="9">LANL EDS</subfield>
     <subfield code="a">Statistical Mechanics</subfield>
   </datafield>

   <datafield tag="695" ind1="" ind2="">
      <subfield code="9">LANL EDS</subfield>
      <subfield code="a">Condensed Matter</subfield>
   </datafield>

   <datafield tag="856" ind1="4" ind2="">
      <subfield code="u">http://documents.cern.ch/[...]id=0503152</subfield>
      <subfield code="y">Access to fulltext document</subfield>
   </datafield>

   [...]

</record>
</collection>
```

Figure 11.3: Sample for metadata information in XML MARC format.

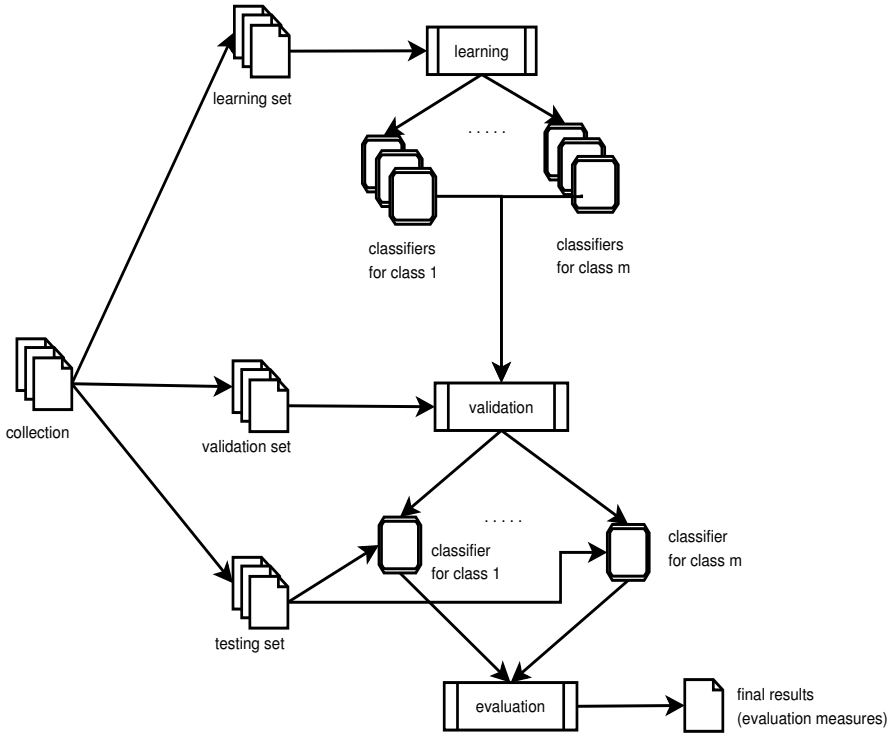
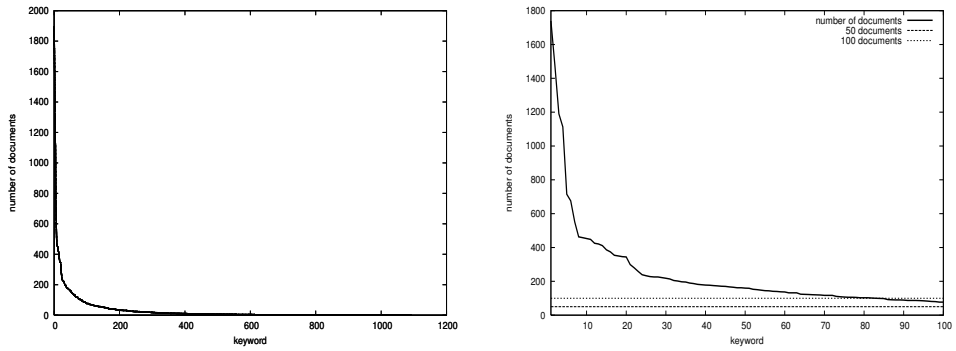


Figure 11.4: Basic data flow within TECAT



(a) All classes (b) 100 most frequent

Figure 11.5: Distribution of classes across documents in the `hep-ex` partition.

No. docs.	Keyword
1898 (67%)	electron positron
1739 (62%)	experimental results
1478 (52%)	magnetic detector
1190 (42%)	quark
1113 (39%)	talk
715 (25%)	Z0
676 (24%)	anti-p p
551 (19%)	neutrino
463 (16%)	W
458 (16%)	jet

Figure 11.6: The ten most frequent main key words in the *hep-ex* partition

11.3 HEP-EX partition

The *hep-ex* partition of the HEP collection is composed of 2839 records and full-text related to experimental high-energy physics that are indexed with 1093 main key words (the categories).[4] Figures 11.5a and 11.5b show the distribution of key words across the collection. As we can see, this partition is **very** imbalanced: only 84 classes are represented by more than 100 samples and only five classes by more than 1000. The uneven use is particularly noticeable for the ten most frequent key words: In table 11.6 the left column shows the number of positive samples of a key word and the right column shows the percentage over the total of samples in the collection.

In some experiments (like 12.10, additional corpora have been used to check the validity of our approach to different data. For that, two additional corpora have been set up: *astro-ph* (with 2,766 documents about Astrophysics), and *hep-th* (with 17,270 documents about theoretical Physics).

[4] We did not consider the key words related to reaction and energy because they are based on formulae and other specific data that is not easily identifiable in the plain-text version of a paper.

Chapter 12

Experiments

12.1 EXP 12.1 - Establishing the evaluation framework

The amount of data available is enough to consider a *k-fold cross validation* methodology in the evaluation of different configurations for the experiments carried out. As we described in chapter 7, a cross validation (a big part of the collection for training and the rest for testing) is performed k different times ([81], pp. 112) using a different partitioning at each turn into training and testing sets. The overall results are computed as an average of the subsequent k measurements registered. Some studies claim that the number of folds k should be set to 10, thus having a 10-fold cross-validation scheme to test the system.

We wanted to ensure this choice to really guarantee the robustness of results. Despite the fact that many research papers do not pay attention to this point for the selection of the averaging strategy (macro vs. micro averaging), we want to justify this decision by carrying out the evaluation of some learning algorithms.

12.1.1 Configuration

In table 12.1 we show the configuration used for this experiment. An explanation of the parameters involved is given in section 10.6. For this set of experiments the *hep-ex* partition was used chapter (see chapter 11 for further details).

Three different algorithms have been used: the Rocchio algorithm, the PLAUM perceptron and Support Vector Machines (SVM). Each algorithm has been used as unique base binary classifier in its own set of experiments, i.e. the classifier for each class has not been trained by selecting the best of the three. We are not interested at this point to test the performance of these algorithms, but rather to check the fluctuation of measurements when using a different num-

indexing		folding	
stop-words	*yes*	DF term filter	*1*
stemming	*yes*	no. folds	*various*
min. term freq.	*0*	IG filter	*50000*
min. term length	*0*	DF class filter	*1*
max. term length	*40*	normalization	*yes*
		term weighting	*TF.IDF*

learning		testing	
S-cut	*yes*	mode	*by class*
measure	*F1*		
threshold	*0.1*		
methods	*Rocchio, SVM or PLAUM*		

Table 12.1: Parametrization of TECAT for experiment 12.1

ber of folds. Therefore, we have run experiments using n-fold cross-validation with different values of n.

The **Rocchio** algorithm has been used with values n={1, 3, 5, 7, 10, 15, 20, 25, 30, 40, 50}. Each cross validation has been run 10 different times. Thus, 2060 experiments have been launched for collecting statistical values that will inform us about the stability of test results depending on the number of folds. If runs for a given number of folds show high variance, then we should use another number of folds. The objective is to determine the minimum number of folds to use in order to guarantee the validity of our future experimental results.

With the **PLAUM** algorithm, the series of folds has been n={3, 5, 7, 10, 15, 20, 25, 30, 40, 50}, with 10 runs each. Thus 2050 experiments have been launched using this perceptron as base classifier. These two former algorithms are built-in to TECAT, therefore regardless the number of runs, we were able to execute them in a reasonable period of time (about 54 hours for PLAUM).

For **SVM** we set n={3, 5, 7, 10, 15}, with 10 runs for each configuration except the 15-fold runs which were executed only 5 times. This was due to the computational cost involved in the runs, since the implementation used was *SVM-Light*, involving an external command invocation for each learning and classification process at every class. 325 runs were performed in total using this algorithm as base binary classifier.

Documents are distributed in a random way but uniformly across folds. The number of folds assigned to each subset depending on the number of folds in total is detailed below, in table 12.2. We should keep in mind that we had three

subsets: one for training base classifiers, one for evaluating and filtering them, and another one for testing the system.

total	training	evaluation	test
1	1	1	1
3	1	1	1
5	2	2	1
7	4	2	1
10	6	3	1
15	9	5	1
20	12	7	1
25	16	8	1
30	19	10	1
40	26	13	1
50	32	17	1

Table 12.2: Number of folds assigned to each subset depending on the total number of folds considered.

12.1.2 Results with Rocchio

From the results obtained with Rocchio we have computed two values: the *standard deviation* and the *range*. The standard deviation tells us how variable the results are for different runs with the same numbe of folds. If a number of folds shows a higher value than another, then the first one is less stable than the second one, which should be preferred as number of folds to be used in the cross-validation process. Figure 12.1 shows the standard deviation obtained for different values of n. We can see how fast it decreases at the beginning, staying below 0.004 for almost any measurement. We can also notice a very interesting behaviour: different measures are sensitive to different folding strategies in different ways. For instance, the precision has reported to be the most sensitive measurement. Its values are more unstable than for the rest of measurements which, more or less, show a similar response to the number of folds applied.

The diagram clearly reports a rare behaviour when the number of folds used is 7, maybe due to the imbalance of the *hep-ex* partition. The imbalance of the data makes many classes be sparsely represented across the collection, therefore being much more sensitive in the way partitions are performed. These would also explain why the curve mounts over the 0,004 deviation value when using a number of folds between 15 and 20.

In figure 12.2 the value measured is the "range" in the variation of measurements, that is, the maximum value obtained minus the minimum one. This gives us the amplitude of the variation in absolute terms (it is not an statistic number). Despite the fact of its apparent irrelevancy from the statistical point

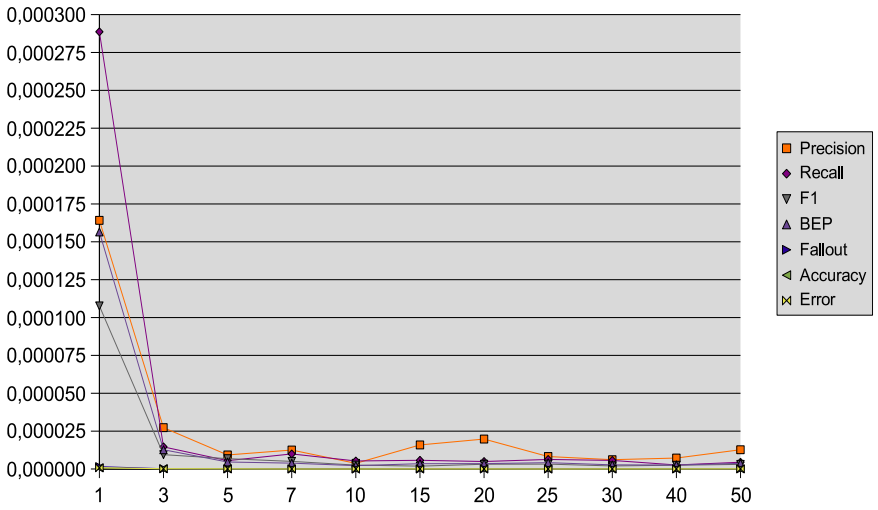


Figure 12.1: Number of folds Vs. standard deviation in cross-validation for Rocchio algorithm

of view, we can see how easy these values can jump. Together with the standard deviation, an idea of the dispersion of possible measurements can be formed depending on the number of folds used to average these evaluation measurements. From this figure we can see that values vary only about 0,2 % when using 10-fold cross validation with the Rocchio algorithm.

12.1.3 Results with PLAUM

To avoid basing our decision of the number of folds to be used on only one classification algorithm, PLAUM experiments with a changing number of folds have been carried out. The Perceptron Learning Algorithm with Uneven Margins has been used this time with the same configuration as for Rocchio in the previous experiments. We can see in figure 12.3 how the standard deviation varies according to the number of folds used to average the different measurements. Again we can notice the random behaviour of the precision value, while for other measurements it seems that a number of folds between 7 and 25 reports more stable values.

Looking at figure 12.4, we can observe the absolute variation between minimum and maximum values obtained for each measure at each of the 10 runs executed for every possible number of folds. Precision could vary over 1,5 %, whereas for other measures this is less than 1 %.

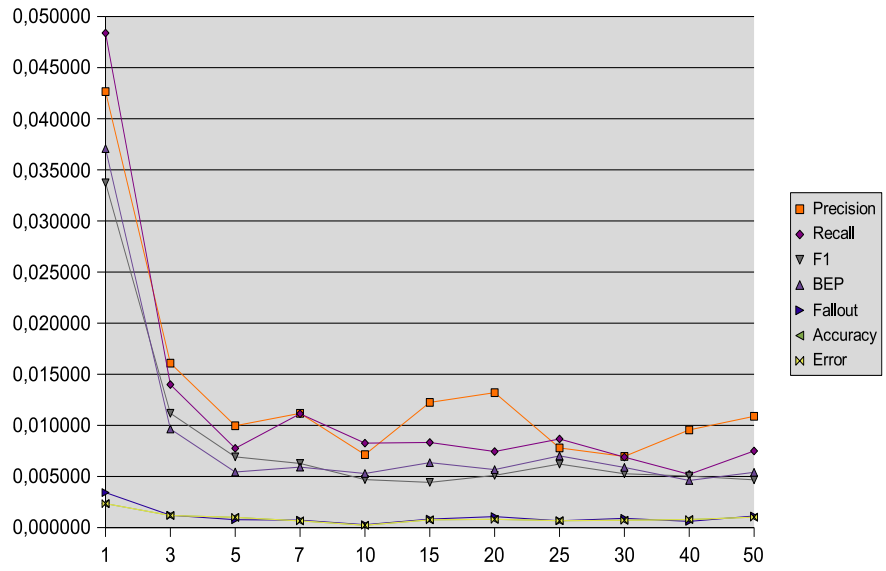


Figure 12.2: Number of folds Vs. range in cross-validation for Rocchio algorithm

12.1.4 Results with SVM

Using Support Vector Machines as base classifier we find a more stable behaviour of the precision, which now shows even a better response with increasing number of folds. Though the variation range shown in figure 12.6 for 10-folds is only under 1,2% (which is pretty high), the standard deviation presents approximately the same values as for other algorithms.

It is noticeable also that the behaviour for SVM is not so random as for PLAUM. We cannot appreciate any "jump" in the diagrams and the standard deviation (figure 12.5) reaches a maximum for 7 folds and then begins to decrease. Folds number 3 seems to behave even better than 10 (except for precision), but looking at figure 12.7 gives us a clue on the subject: the precision of SVM increases very fast when moving from 3 to 7 folds (it is similar for other evaluation measurements). The reason for this is that SVM is *very* sensitive to imbalance. With an increasing number of folds, the training set is bigger and the test set smaller, therefore SVM finds more robust margins and becomes more stable.

12.1.5 Analysis of results

Once all these experiments have been run, and with the results described above, we can justify the validation framework used in experiments described in following pages. 3, 5 or 7 fold is not good enough to assure the stability of the results

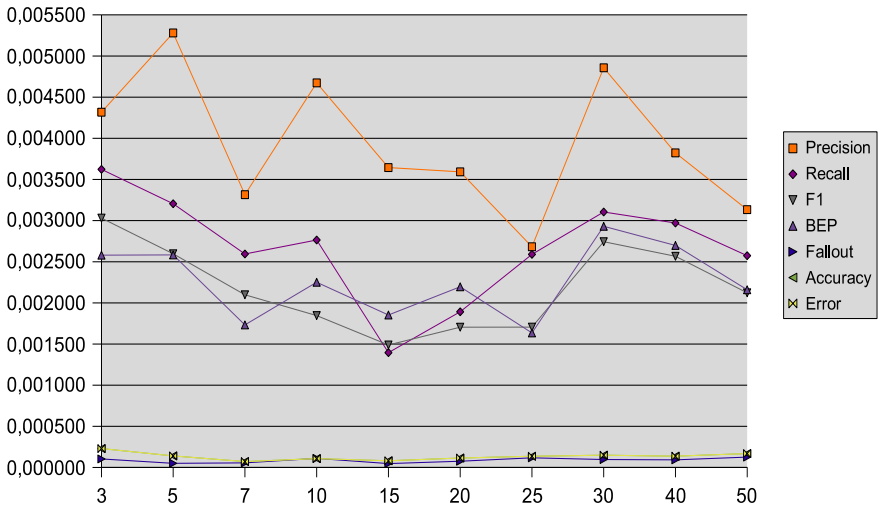


Figure 12.3: Number of folds Vs. standard deviation in cross-validation for PLAUM algorithm

and, consequently, our conclusions based on them. 15-fold cross validation is better for margin based classifiers like PLAUM and SVM, but not that good for Rocchio, for instance. Another main point against a number of folds beyond 15 is the computational cost that it implies. We can see how the number of folds affects the three algorithms in precision and recall looking at comparison diagrams 12.8 and 12.9.

The conclusions obtained from this study reflects that:

- Imbalance affects the stability of experimental results, so a cross validation is needed with no doubt
- Using 10-fold cross validation is a compromise of stability for margin and similarity based algorithm
- The standard deviation is lower for PLAUM and SVM, but the range is wider because margin based algorithms are more sensitive to imbalance
- To guarantee the validity of variations in results we will apply 10-fold cross validation according to the following table:

algorithm	*precision*	*recall*	*F1*
Rocchio	$\geq$ 0,72 %	$\geq$ 0,84 %	$\geq$ 0,47 %
PLAUM	$\geq$ 1,57 %	$\geq$ 0,90 %	$\geq$ 0,67 %
SVM	$\geq$ 0,88 %	$\geq$ 1,11 %	$\geq$ 1,16 %

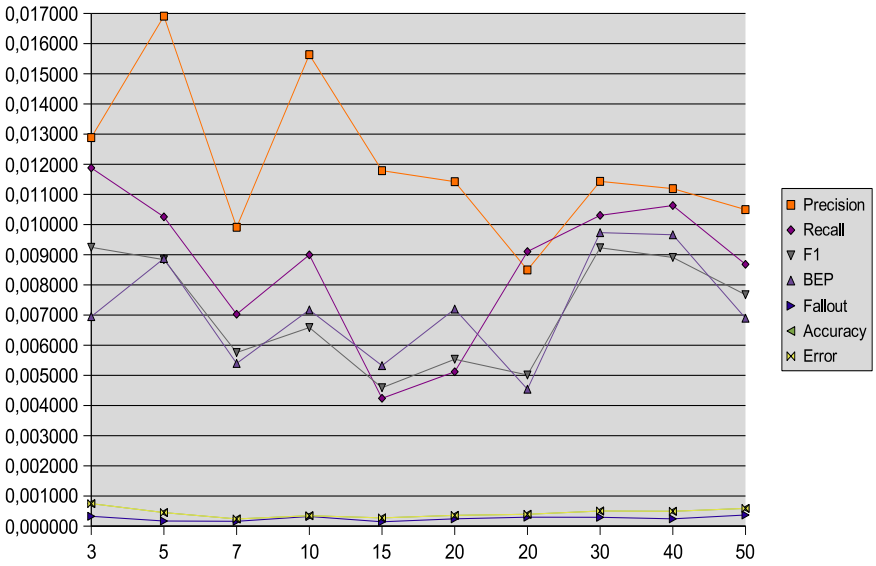


Figure 12.4: Number of folds Vs. range in cross-validation for PLAUM algorithm

Variations of measures for different configurations below given margins will not be considered as relevant. For example, if using a filter in information gain of 50000 terms increases the precision of SVM in 0,5 % over a filter with 20000, we will consider that both filters have same performance and, therefore, prefer the second one which economizes the number of terms involved in the representation of a document vector.

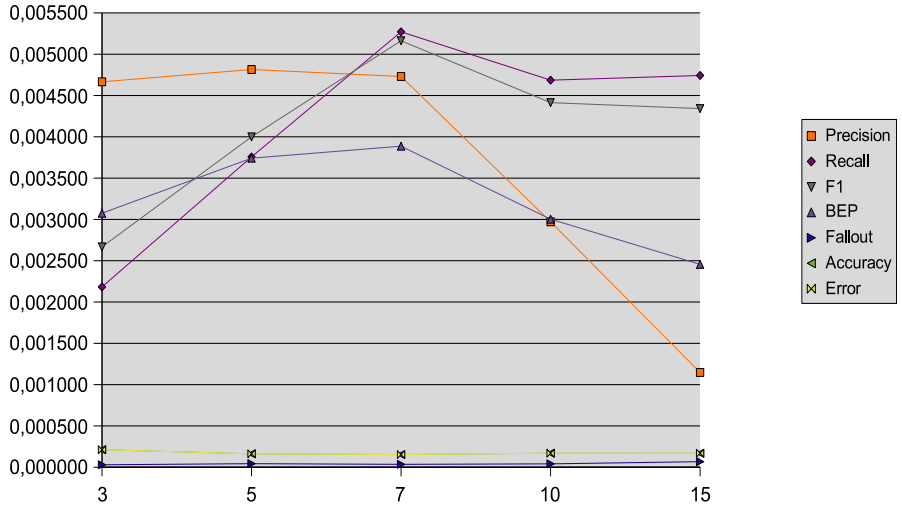


Figure 12.5: Number of folds Vs. standard deviation in cross-validation for SVM algorithm

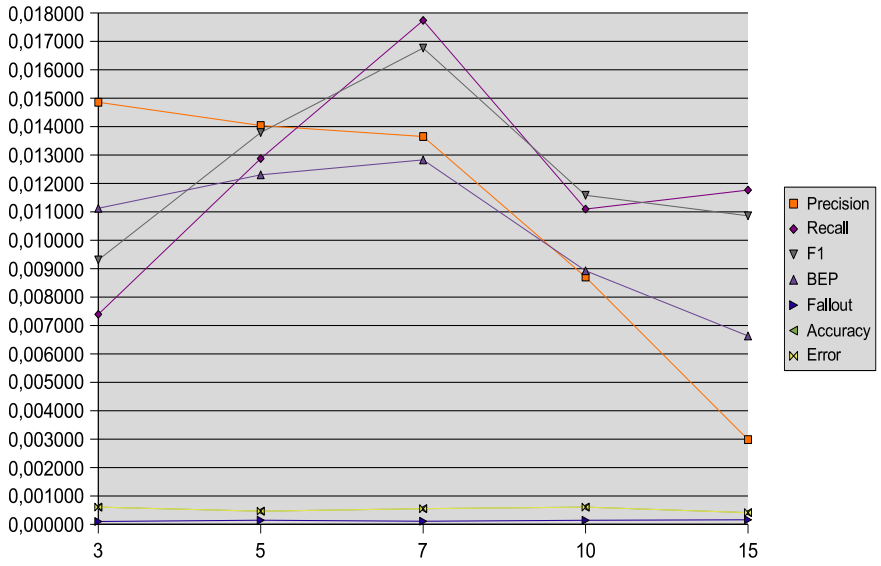


Figure 12.6: Number of folds Vs. range in cross-validation for SVM algorithm

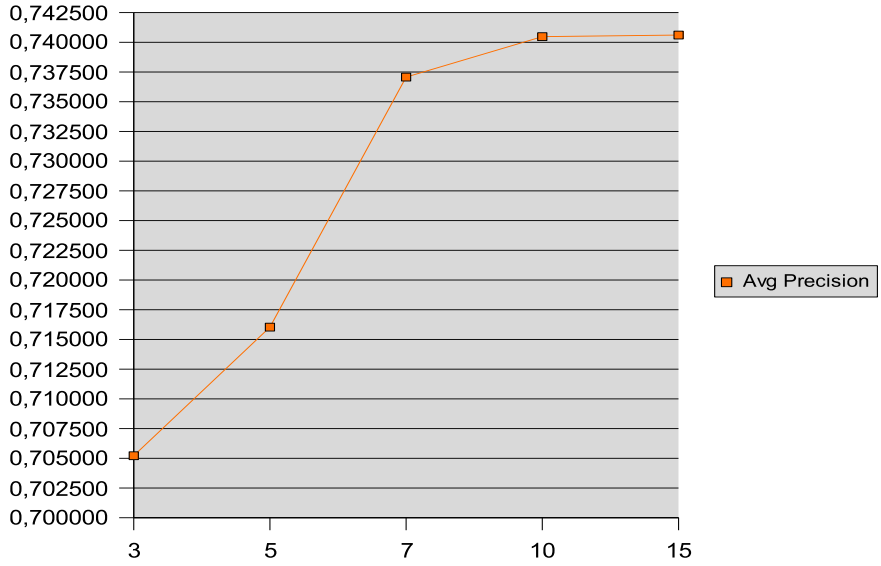


Figure 12.7: Number of folds Vs. range in cross-validation for SVM algorithm

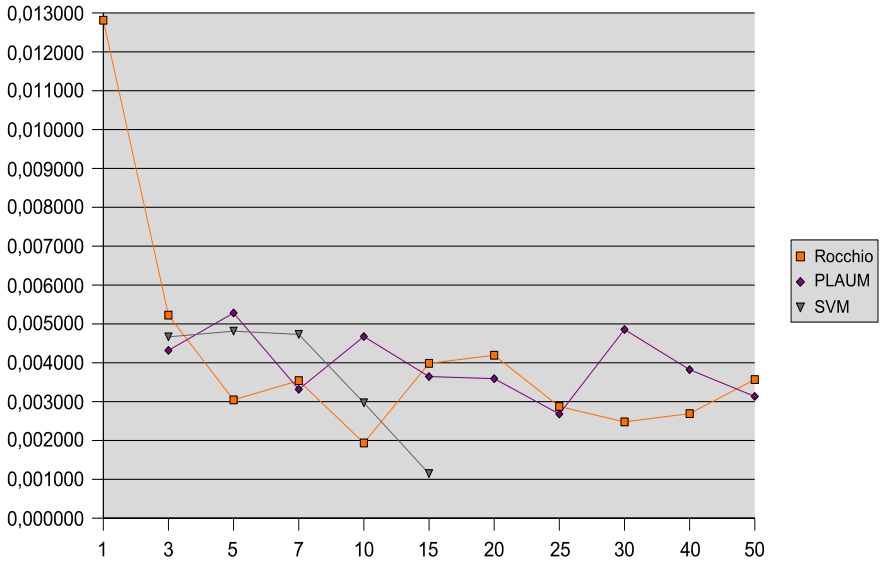


Figure 12.8: Number of folds Vs. standard deviation of precision in cross-validation for Rocchio, PLAUM and SVM algorithms

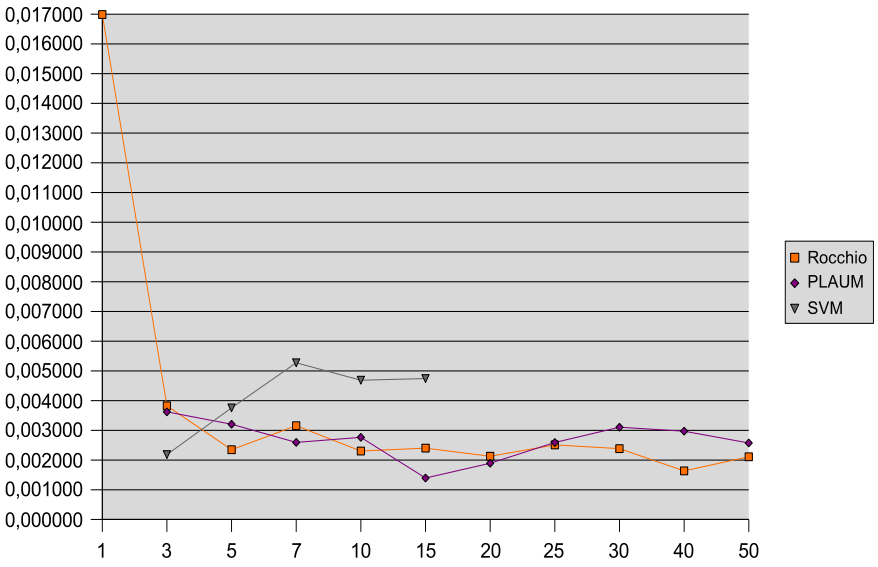


Figure 12.9: Number of folds Vs. standard deviation of recall in cross-validation for Rocchio, PLAUM and SVM algorithms

12.2 EXP 12.2 - Comparing different base classifiers

When constructing a multi-label classifier using the architecture defined in previous chapters, the selection of base binary classifiers is very important. We could say that the performance of the system is mainly dependent on the right use of a good binary classifiers. Binary classifiers with poor performance will lead to a non-performing multi-label classifier. In these experiments we have considered three different base classifiers: Rocchio, PLAUM and SVM (see chapter 6 for a detailed description of these algorithms).

It is our intention here to study the behaviour of such algorithms within our experimental framework and determine which of them should be preferred as base classifiers. TECAT has been designed to allow the use of different classifiers at the same time. These different classifiers can be variations in the configuration for the same learning algorithm or truly different approaches. The system will select the best classifier, by evaluating it over an evaluation subset of the training set, as the base classifier for the class. This is like a voting committee where only one classifier receives all votes for a given class. How this combination of base classifiers affects the final performance is matter of study at this section.

12.2.1 Configuration

In table 12.3 we show the configuration used for this experiment. An explanation of the parameters involved is given in section 10.6. For this set of experiments, again the *hep-ex* partition was used (see chapter 11 for further details). As we can see, now the only difference between configurations is given by the set of base classifiers passed as candidates for the training phase.

In total, seven trained systems have been tested in this set of experiments. First, three multi-label classifiers base in only one base classifier with a fixed configuration: Rocchio, PLAUM and SVM as base classifiers respectively for each multi-label classifier. These trainings correspond to first three entries in table 12.4 (*Rocchio-simple*, *PLAUM-simple* and *SVM-simple*). The next three entries (*Rocchio-multi*, *PLAUM-multi* and *SVM-multi*) are multi-label classifiers trained again using a fixed algorithm as before, but in this case a range of configurations (combinations of parameters) has been provided so that for a class the best configuration of the given classifier is, in principle, selected. For example, the *Rocchio-multi* entry shows that values for α and β parameters are set from a given set of possible values. We have five possible values for each parameter, thus, 25 different combinations have been passed to TECAT as candidates in the training phase. It is the same for the rest of entries whenever a set of possible values is provided, all the possible combinations of these values were tested. As said, 25 configurations of Rocchio were used in the *Rocchio-multi*

indexing		folding	
stop-words	*yes*	DF term filter	*1*
stemming	*yes*	no. folds	*10*
min. term freq.	*0*	IG filter	*50000*
min. term length	*0*	DF class filter	*1*
max. term length	*40*	normalization	*yes*
		term weighting	*TF.IDF*

learning		testing	
S-cut	*see table 12.4*	mode	*by class*
measure	*F1*		
threshold	*0.1*		
methods	*see table 12.4*		

Table 12.3: Parametrization of TECAT for experiment 12.2

experiment, 16 for *PLAUM-multi*, 5 for *SVM-multi* and 41 for the *Mixed* experiments, where both Rocchio and PLAUM are provided under all the variants proposed before.

12.2.2 Results

Using 10-fold cross-validation, performance measures have been macro-averaged at each of the 10 rounds in a per-document basis. Final measures shown in table 12.5 are computed by averaging rounds results. The column titled *% of classes* tells us about the percentage of classes that found a valid classifier to represent them. We can see, for instance, that SVM is the most discriminative classifier, since few classes get successfully trained under this algorithm. In figure 12.10 a graphical comparison of obtained performances is shown.

About the speed, we have timed[1] how long it takes for the system to classify a single document. TECAT is optimized to fast testing, since we assign documents to classes instead of classes to documents. Thus, for timing the real label assignment to documents, a different experiments has been carried out. Using again the *hep-ex* partition, each single abstract was passed to TECAT to obtain the list of predicted classes associated. Each assignment involves loading class models, vectorization of the plain-text file and running classifiers for every trained class. This experiment was executed on a DELL dual Intel Xeon machine at 2.8 GHz per processor, equipped with 1 GB of RAM memory.

As we can see, TECAT assigns about 5 classes per document in far less than 2 seconds. This can be considered a *real time* classifier. For the interests

[1]The `time` UNIX command was used for this purpose.

experiment	base classifier(s)	threshold	parameters
Rocchio-simple	Rocchio	S-Cut	$\alpha = 1$ $\beta = 1$
PLAUM-simple	PLAUM	0	$T = 100$ $\tau_+ = 1$ $\tau_- = -1$ $\eta = 1$ $\lambda = 0$
SVM-simple	SVM	0	*cost-factor* $= 1$
Rocchio-multi	Rocchio	S-Cut	$\alpha = \{1, 2, 5, 10, 50\}$ $\beta = \{1, 2, 5, 10, 50\}$
PLAUM-multi	PLAUM	0	$T = 100$ $\tau_+ = \{0, 1, 10, 100\}$ $\tau_- = \{-10, -1, 0, -1\}$ $\eta = 1$ $\lambda = 0$
SVM-multi	SVM	0	*cost-factor* $= \{1, 2, 5, 10, 50\}$
Mixed	Rocchio + PLAUM	S-Cut	For Rocchio as for *Rocchio-multi* For PLAUM as for *PLAUM-multi*

Table 12.4: Base classifiers configuration

precision	recall	F1	accuracy	error	% of classes	experiment
0.46970	0.54166	0.45971	0.97208	0.02791	87.90	*Rocchio-simple*
0.69610	0.41221	0.49107	0.98190	0.01809	53.28	*PLAUM-simple*
0.73508	0.33396	0.43308	0.98208	0.01791	**31.57**	*SVM-simple*
0.50024	**0.56878**	0.50977	0.97605	0.02394	91.12	*Rocchio-multi*
0.52552	0.52902	0.50734	0.97753	0.02246	68.40	*PLAUM-multi*
0.70321	0.44066	0.51287	0.98256	0.01744	55.20	*SVM-multi*
0.51357	0.56465	**0.51482**	0.97648	0.02351	92.13	*Mixed*

Table 12.5: Results of experiments with different configurations

measure	no. labels	real time	user time	system time
mean	5,29	1,13	1,09	0,04
maximum	17	2,13	2,09	0,07
minimum	0	1,06	1,01	0,03
mode	4	1,1	1,07	0,04
variance	8,2	0,01	0,01	0
standard deviation	2,86	0,11	0,11	0,01

Table 12.6: Classification speeds (in seconds) for TECAT using PLAUM on the *hep-ex* partition

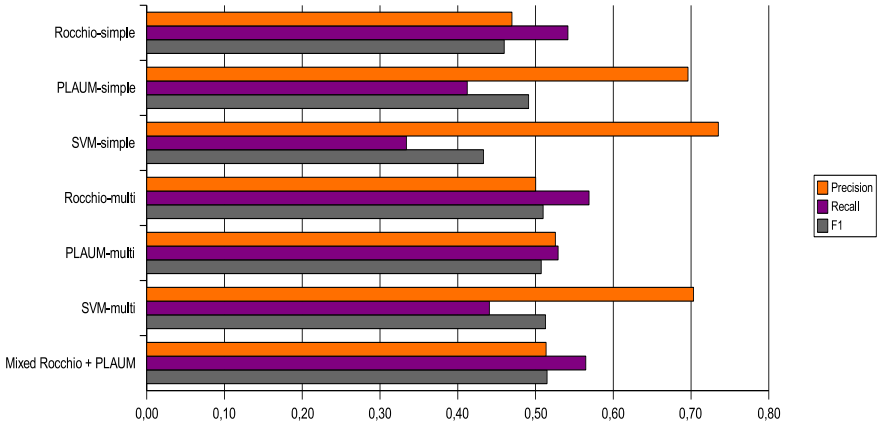


Figure 12.10: Performance of different strategies for base classifier candidates

of CERN, the capability of TECAT for proposing labels in that sort time allows researches to generated on the fly classes for their working documents so they can find related documents already classified and stored in the database. Experiment 12.11 explores real time classification in a more subject centered manner.

12.2.3 Analysis of results

The goal of these experiments is to determine whether the configuration of the base classifier should be different for each class or not. Intuitively, the answer is *yes*, since each class may need a different parametrization of the base classifier. An easy way to allow that is to operate as done by TECAT: by selecting the classifier from a range of variants according to the observed performance on a set of samples. The cost of the training is higher, but the final multi-label classifier is not affected by the number of candidates, since we still have one classifier per class.

From results we can underline the following points:

1. The Support Vector Machines algorithm is usually not trainable on a large set of classes, producing a low percentage of trained classes as observed in the results, and a lower recall compared to other algorithms. The use of additional configurations for candidates results in a higher recall (from 33% up to 44%) without affecting too much the precision. The value observed for F1 measure is significant.

2. The approach of adaptive selection of the best classification configuration for each class results is an improvement of performance for any base

classifier over an unique parametrization. That is, for a fixed learning algorithm, it is good to use a validation subset to select best parameters in an automated mode.

3. In basis of F1 measure as comparison value, the mixed strategy is the best choice. It is even better than the one with PLAUM with single configuration, but only slightly.

4. Comparing SVM with multiple configurations versus the mixed strategy, it is impossible to assert that one strategy outperforms the other one just by looking at the F1 measure. Nevertheless, it is clear that recall and precision are more balanced when following mixed strategy, since SVM tends to prime precision over recall.

In general, we conclude that allowing different algorithms and parameter adjusting depending on the class leads to better classification results. In multi-label problems the large number of classes makes it prohibitive to combine several algorithms for every class, but by applying the described adaptive selection we are not increasing computational cost, because the total number of classifiers is at most equal to the number of classes. In fact, even by setting the validation threshold constraint to a low value (0.1 in our experiments) the number of discarded classifiers and, consequently, classes, can be important without a significant loss in performance but a significant gain in speed.

12.3 EXP 12.3 - Use of bigrams detection

Normally, the meaning of a word has no sense without the companion of adjacent words. In some cases the concept is represented by a combination of the so called *multi-words* (see subsection 4.1.1). Feature identification can be enhanced by detecting such cases and forming an unique term from them. We have taken the whole collection and computed the *mutual information* between each possible pair of terms. These pairs have been ranked in two lists: one by this mutual information value, and another by its frequency of occurrence. When a pair is very frequent and is "well formed" (without containing stop words and following a valid *part-of-speech*, POS, combination) it could be also worth considering. That is the reason for also experimenting with this second simply method of multi-word list generation. Thus, for our experiments we have considered only multi-words of size "two", that is, bigrams.

12.3.1 Configuration

For these experiments also the Rocchio and PLAUM methods have been tried out. In table 12.7 we show the standard configuration applied. See section 10.6

to interpret correctly the meaning of each parameter. For this set of experiments, again the *hep-ex* partition was used (see chapter 11). The variability is now given not only by the type of base classifier applied, but also for the number of multi-words detected on the corpus.

indexing		**folding**	
stop-words	*yes*	DF term filter	*1*
stemming	*yes*	no. folds	*10*
min. term freq.	*0*	IG filter	*50000*
min. term length	*0*	DF class filter	*1*
max. term length	*40*	normalization	*yes*
		term weighting	*TF.IDF*

learning		**testing**	
S-cut	*only for Rocchio*	mode	*by class*
measure	*F1*		
threshold	*0.1*		
methods	*Rocchio or PLAUM*		

Table 12.7: Parametrization of TECAT for experiment 12.3

As we have generated two lists of ranked bigrams (one by mutual information and another one by frequency) with all possible well formed pairs appearing in the corpus. It is needed to determine which ones of those bigrams will be detected as multi-words in the corpus. For that, only the top ranked ones have been considered relevant bigrams. The number of top ranked bigrams tried are 50, 100, 500 and 1000. We have, therefore, 16 different runs as a combination of base classifiers (Rocchio or PLAUM), list of candidate bigrams (by frequency of by mutual information) and the number of bigrams considered from the selected list.

The larger the size of the corpues, the more representative the mutual information becomes. For that we have created another sequence of runs based also on the *hep-ex* partition, but using full-text documents instead of just abstracts. It is not the goal of these experiments to establish a judgement between the use of abstracts or the use of full-text, but to determine whether the identification of multi-words is worth considering. As final remark, note that bigrams ranked list are generated from the whole corpus, not only the training partition. This does not bias results, as mutual information or paired frequencies are measures independent from classes associated to documents.

Precision	Recall	F1	BEP	Accuracy	% classes	n-top	By	Algorithm
0.6961	0.4122	0.4910	0.5541	0.9819	53.28	0	nothing	PLAUM
0.6921	0.4167	0.4931	0.5544	0.9820	52.58	50	MI	PLAUM
0.6925	0.4111	0.4880	0.5518	0.9818	52.50	100	MI	PLAUM
0.6860	0.4121	0.4883	0.5490	0.9817	52.15	500	MI	PLAUM
0.6890	0.4082	0.4862	0.5486	0.9818	51.50	1000	MI	PLAUM
0.6850	0.4101	0.4869	0.5475	0.9818	52.89	50	Freq.	PLAUM
0.6911	0.4173	0.4937	0.5542	0.9819	51.94	100	Freq.	PLAUM
0.6935	0.4028	0.4821	0.5481	0.9819	51.16	500	Freq.	PLAUM
0.6923	0.3952	0.4762	0.5437	0.9818	52.41	1000	Freq.	PLAUM
0.4668	0.5410	0.4570	0.5039	0.9720	88.15	50	MI	Rocchio
0.4683	0.5420	0.4581	0.5051	0.9719	87.30	100	MI	Rocchio
0.4656	0.5367	0.4545	0.5012	0.9717	87.53	500	MI	Rocchio
0.4647	0.5390	0.4543	0.5018	0.9716	87.60	1000	MI	Rocchio
0.4764	0.5454	0.4648	0.5109	0.9727	88.19	50	Freq.	Rocchio
0.4767	0.5487	0.4683	0.5127	0.9727	88.12	100	Freq.	Rocchio
0.4721	0.5484	0.4678	0.5103	0.9728	87.84	500	Freq.	Rocchio
0.4813	0.5469	0.4706	0.5141	0.9730	89.12	1000	Freq.	Rocchio

Table 12.8: Results in experiment 12.3 using **abstracts** corpus

Precision	Recall	F1	BEP	Accuracy	% classes	n-top	By	Algorithm
0.6961	0.4122	0.4910	0.5541	0.9819	53.28	0	nothing	PLAUM
0.7114	0.4387	0.5147	0.5751	0.9828	58.90	50	MI	PLAUM
0.7081	0.4382	0.5140	0.5732	0.9827	59.80	100	MI	PLAUM
0.5565	0.1857	0.2633	0.3711	0.9261	31.35	500	MI	PLAUM
0.5723	0.2045	0.2820	0.3884	0.9261	34.14	1000	MI	PLAUM
0.7122	0.4395	0.5172	0.5759	0.9828	59.04	50	Freq.	PLAUM
0.7051	0.4325	0.5091	0.5688	0.9826	57.82	100	Freq.	PLAUM
0.7081	0.4365	0.5123	0.5723	0.9828	58.47	500	Freq.	PLAUM
0.7028	0.4347	0.5099	0.5687	0.9825	56.75	1000	Freq.	PLAUM
0.4242	0.5223	0.4268	0.4733	0.9699	86.92	50	MI	Rocchio
0.4216	0.5199	0.4236	0.4708	0.9693	86.92	100	MI	Rocchio
0.3257	0.5128	0.3695	0.4192	0.8750	91.04	500	MI	Rocchio
0.3464	0.5143	0.3853	0.4304	0.8783	91.78	1000	MI	Rocchio
0.4256	0.5204	0.4261	0.4730	0.9698	85.74	50	Freq.	Rocchio
0.4232	0.5188	0.4244	0.4710	0.9696	86.89	100	Freq.	Rocchio
0.4310	0.5240	0.4321	0.4775	0.9703	87.44	500	Freq.	Rocchio
0.4227	0.5235	0.4307	0.4731	0.9700	86.55	1000	Freq.	Rocchio

Table 12.9: Results in experiment 12.3 using **full-text** corpus

12.3.2 Results

In table 12.8 the measurements for each possible configuration are listed when using abstracts as corpus. Results for full-text corpus are in table 12.9. These values are obtained by macro-averaging per document a stratified ten-fold run (see chapter 7 for a detailed description about it).

For both tables, the measures given are

- precision,
- recall,
- F1,
- break-even-point simply computed as the mean between precision and recall,
- accuracy,
- the percentage of classes successfully trained,
- the number of top bigrams considered,
- the base of the ranking (*MI* for mutual information and *Freq.* for raw frequency), and
- the base algorithm used.

There are two rows at each table with the measures obtained when no bigrams are considered, that is, not multi-word detection is performed. These rows allow the analysis of the whole multi-word detection technique for our HEP collection.

12.3.3 Analysis of results

A graphical representation of precision, recall and F1 results is given in figures 12.11a and 12.11b for the Rocchio algorithm on the abstracts corpus, 12.12a and 12.12b for the PLAUM on the same corpus, and 12.13a and 12.13b for the PLAUM algorithm on full-text corpus.

From these graphs a simple conclusion is reached: the use of the multi-word detection technique described above does not report significant improvements. Moreover, on the full-text corpus it can eventually leads to worse results, as seen on figure 12.13. These results are coherent with the conclusions reached by Lewis [65] on his study about feature selection. The use phrases and the selection features with higher rated mutual information values is not strong enough to justify its use. Therefore, for text categorization in HEP domain, no multi-word detection will be used.

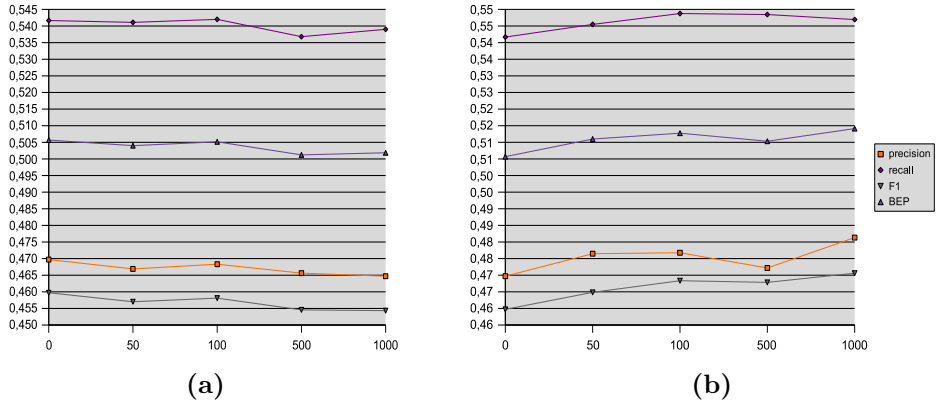


Figure 12.11: Influence of multi-word detection on abstracts corpus for the **Rocchio** algorithm using (a) mutual information ranked list, or (b) frequency ranked list

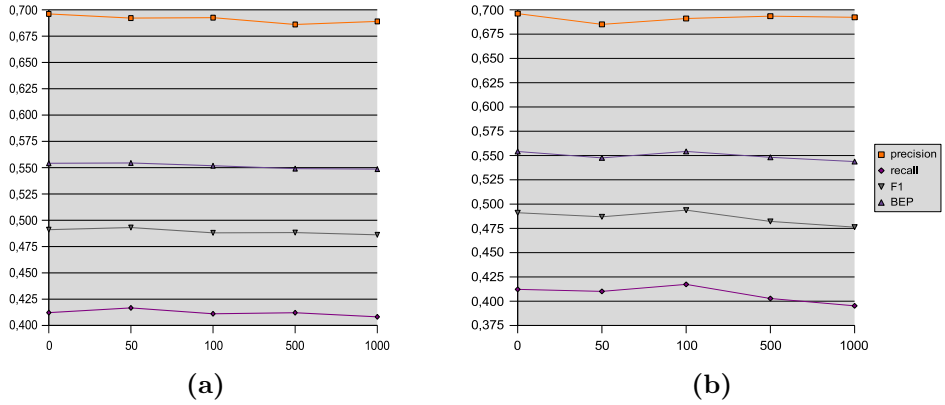


Figure 12.12: Influence of multi-word detection on abstracts corpus for the **PLAUM** algorithm using (a) mutual information ranked list, or (b) frequency ranked list

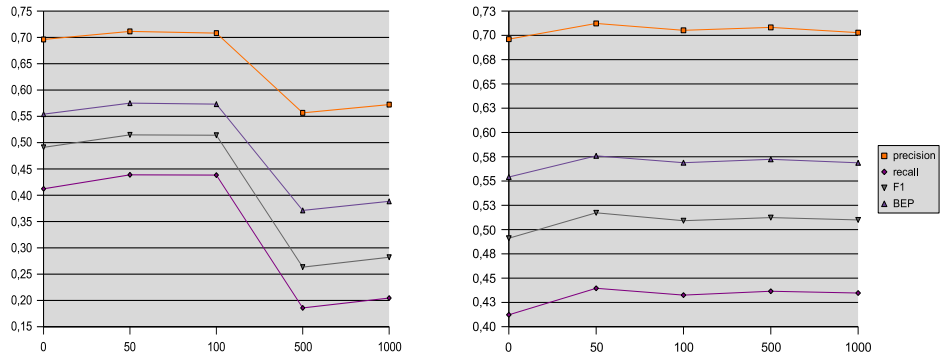


Figure 12.13: Influence of multi-word detection on full-text corpus for the **PLAUM** algorithm using (a) mutual information ranked list, or (b) frequency ranked list

12.4 EXP 12.4 - Stemming and stop words removal

Stemming and stop words removal are two usual steps for feature reduction in text processing for information retrieval (see section 4.1 for more details). Stemming is a fast way to strip out the meaningless part of a word (containing morphological information like gender, number, person, etc.) and obtain the root of it, which is supposed to represent the core meaning of the word. On the other hand, stop words removal consists of discarding all those words appearing in a given list. The words in the list are usually words like determinants, conjunctions or auxiliary verbs, assuming that they lack of relevance for the document. As we can see, this simplification of the language richness down-to simple stems is a legacy from the vector space model, where a document is modeled to be a list of weighted terms. In this case, any semantic contained in the text is reduced to independent stems.

Despite its drastic simplicity, these two techniques have proven their effectiveness as feature reduction techniques. Our goal in the current set of experiments is to determine whether such hypothesis holds in our problem domain.

12.4.1 Configuration

In table 12.10 parametrization of TECAT is detailed. It is shown how stemming and stop words removal will be enabled alternatively to explore all possible combinations. Also we can notice the base binary classifiers that will be used.

indexing		folding	
stop-words	*yes/no*	DF term filter	*1*
stemming	*yes/no*	no. folds	*10*
min. term freq.	*0*	IG filter	*50000*
min. term length	*0*	DF class filter	*1*
max. term length	*40*	normalization	*yes*
		term weighting	*TF.IDF*

learning		testing	
S-cut	*for Rocchio and Widrow-Hoff*	mode	*by class*
measure	*F1*		
threshold	*0.1*		
methods	*Rocchio, PLAUM or Widrow-Hoff*		

Table 12.10: Parametrization of TECAT for experiment 12.4

Again, the *hep-ex* partition has been used as base corpus for experimentation on the effect of applying or not applying stemmer and/or stop words removal.

Three sets of experiments have been carried out, each one with a different base classifier (to ensure results do not depend on such choice). Algorithms used have been *Rocchio*, *Widrow-Hoff* and *PLAUM*, all of them described in previous chapters.

12.4.2 Results

In tables 12.11, 12.12 and 12.11 the measures for *precision*, *recall* and *F1* are given for the algorithms considered. We can see how the combination of use of the two techniques discussed in this experiment provides different performances. The highest value for each of the three measures is marked in bold.

Widrow-Hoff algorithm		**Stop words removal**	
		ON	OFF
Stemmer	ON	Precision 0.446720 Recall **0.536308** F1 **0.454100**	Precision 0.436516 Recall 0.531219 F1 0.446665
	OFF	Precision 0.443430 Recall 0.529131 F1 0.448989	Precision **0.450532** Recall 0.530579 F1 0.452681

Table 12.11: Results in experiment 12.4 using Widrow-Hoff algorithm

We can see in table 12.11, that the highest value of precision is obtained when no stop words removal nor stemming is applied.

Rocchio algorithm		**Stop words removal**	
		ON	OFF
Stemmer	ON	Precision 0.463353 Recall 0.539928 F1 0.454927	Precision 0.472351 Recall 0.536897 F1 0.458039
	OFF	Precision 0.472654 Recall **0.540725** F1 **0.460108**	Precision **0.476874** Recall 0.537542 F1 0.458329

Table 12.12: Results in experiment 12.4 using Rocchio algorithm

PLAUM algorithm		**Stop words removal**	
		ON	OFF
Stemmer	ON	Precision 0.689221 Recall **0.413704** F1 **0.490228**	Precision **0.706438** Recall 0.387936 F1 0.472011
	OFF	Precision 0.700645 Recall 0.390677 F1 0.473383	Precision 0.700898 Recall 0.390271 F1 0.472247

Table 12.13: Results in experiment 12.4 using PLAUM algorithm

12.4.3 Analysis of results

Results shown above may seem to not conclude any final affirmation. It seems that the system is not very sensitive to the use or not of these two feature reduction approaches. Anyhow, despite the behaviour of the Rocchio algorithm, it could be considered that the combination of both techniques leads to a better overall performance (in terms of the F1 measure). But the increment in performance is in most cases very tight. Though this fact may make us think of not using stemming or stop words removal, we have to take into account the benefits of it: we are drastically reducing the feature space dimensionality. In our experiments we have not shown how the speed of the system is improved and how the size of the database is lowered by the combination of both methods.

We conclude that stemming and stop words removal are **worth using** for our text classification tasks because they not only tend to produce improvements in performance, but also an obvious optimization in disk usage and computation time is experienced.

12.5 EXP 12.5 - Dimensionality reduction

When dealing with text documents, the vector space model offers a framework able to convert them into manageable set of data. This implies the identification and filtering of features, that are generated from the words found in the text. As we have seen in the two previous sections, the process which generates these list of terms, which are weighted afterwards, has mainly two common goals: 1) select those terms that better represent the content of the text, and 2) filter out as much as possible in order to reduce the high dimensionality of original feature space (see chapter 5). That is one of the main problems in text based machine learning: learning algorithms are not very fast when working on vectors with thousands of possible features (even when they are very sparse). These two concepts have the same goal: by filtering we want to keep only the most meaningful words.

Therefore, additional effort has to be done at this level. In the present set of experiments, the *information gain* and *document frequency* filters are tested to prove their suitability not only to reduce the feature space but to select most informational terms.

12.5.1 Configuration

indexing		**folding**	
stop-words	*yes*	DF term filter	{*1, 5, 10, 50, 100*}
stemming	*yes*	no. folds	*10*
min. term freq.	*0*	IG filter	{*1000, 2000, 5000, 10000, 20000, 50000*}
min. term length	*0*	DF class filter	*1*
max. term length	*40*	normalization	*yes*
		term weighting	*TF.IDF*

learning		**testing**	
S-cut	*for Rocchio*	mode	*by class*
measure	*F1*		
threshold	*0.1*		
methods	*Rocchio or PLAUM*		

Table 12.14: Parametrization of TECAT for experiment 12.5

Experimental documents about High Energy Physics have been used from

two sources: fulltext generated from PDF files and abstracts from record metadata, both from the *hep-ex* partition. In this way we can study how the reduction of features affects to large and short documents. Fulltext and abstracts corpora differ in the tried values for each filter. Table 12.15 show the combinations of values for both filters and on which corpus a certain combination has been evaluated. We can see that for abstracts corpus the values used to filter out those features not appearing in more of a certain number of documents are {0, 1, 5, 10, 50, 100}, whether when using the fulltext corpus the last two values are not tested (to reduce the number of runs). The maximum dimensionality of the vector space allowed when applying the selection of features by their information gain value are {1000, 2000, 5000, 10000, 20000} for both corpora.

		Max. DF allowed					
		0	**1**	**5**	**10**	**50**	**100**
	1,000	abstracts	abstracts fulltext	abstracts fulltext	abstracts fulltext	abstracts	abstracts
Top	**2,000**	abstracts	abstracts fulltext	abstracts fulltext	abstracts fulltext	abstracts	abstracts
by	**5,000**	abstracts	abstracts fulltext	abstracts fulltext	abstracts fulltext	abstracts	abstracts
IG	**10,000**	abstracts	abstracts fulltext	abstracts fulltext	abstracts fulltext	abstracts	abstracts
	20,000	abstracts	abstracts fulltext	abstracts fulltext	abstracts fulltext	abstracts	abstracts

Table 12.15: Different values tried for both abstracts and fulltext corpora

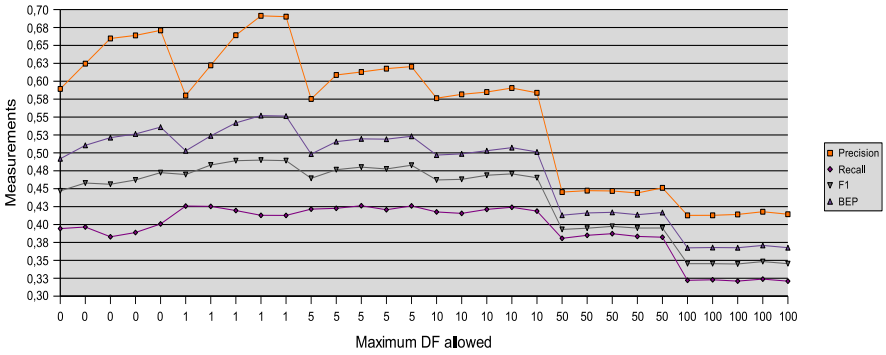
In total, 30 different combinations of DF and IG filters have been performed on the abstracts corpus, and 15 combinations for the fulltext corpus. Thus, exactly 90 (45 combinations $\times$ 2 possible learning algorithms) configurations have been tried to determine how these two parameters responsible of a feature reduction may affect the performance of the classification system.

12.5.2 Results

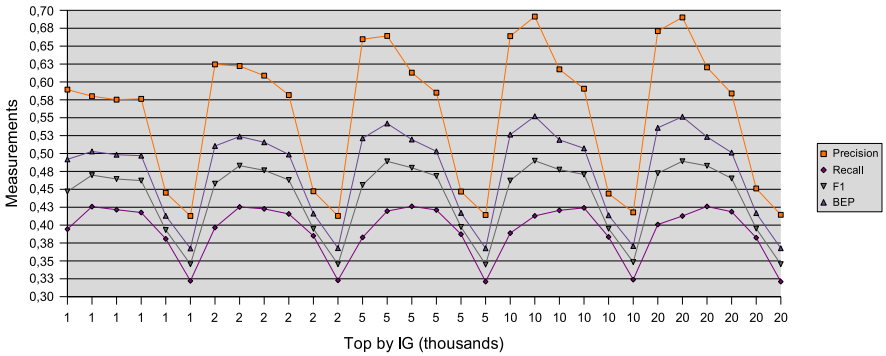
We are not interested in the specific value of performance reached for any of these combinations, but rather in the improvement expressed by some combinations over the rest. Four measures have been analyzed: *precision*, *recall*, *F1* and *break-even point* (BEP). These four measures are described in detail in chapter 7.

For each corpus, two pairs of diagrams are drawn, each pair with a different learning algorithm: Rocchio (diagrams 12.15a, 12.15b, 12.17a, 12.17b) and PLAUM (diagrams 12.14a, 12.14b, 12.16a and 12.16b). Each pair is composed by one diagram showing the effect of the document frequency based filter (diagrams 12.15a, 12.14a, 12.17a and 12.16a) and another one exposing the effect of the filter by information gain (diagrams 12.15b, 12.14b, 12.17b and 12.16b). In the case of frequency based filter, the X axis indicates the number of documents a feature must appear to pass the filter. For the information gain filter, this

axis shows the number (in thousands) of top ranked features by information gain value that are preserved by the filter. In this way we can study how each filter tuning affects performance measures to both proposed corpora (abstracts and fulltext) on two alternative learning algorithms.



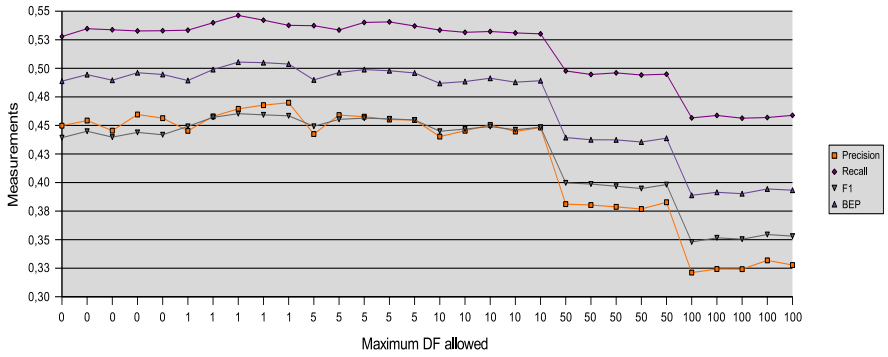
(a)



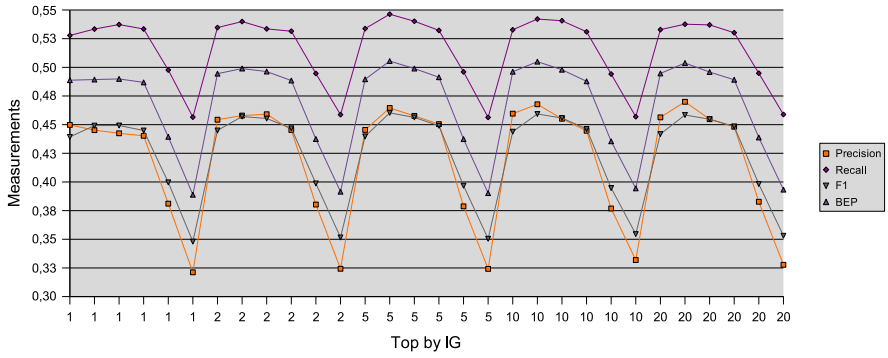
(b)

Figure 12.14: Influence of (a) document frequency and (b) information gain filters on **abstracts** corpus using **PLAUM** learning algorithm

When using corpus of documents with a reduced set of features (as in abstracts corpus, where we have about 100 words per document) we find that the filter by document frequency is more or less stable from a value of 1 (that is, the document frequency filter is worth applying). Nevertheless, starting from a value of 10 documents, i.e. when the feature has to appear in 10 or more documents to avoid being discarded, the filtering results in a significant decrease in performance, as we can see at diagrams 12.15a and 12.14a. The reduction based on information gain does not affect noticeably the performance of the classifier, since for the abstract corpus the number of features after *stemming* and *stop words removal* (see experiment 12.4) is just about 1500 features, so only the



(a)



(b)

Figure 12.15: Influence of (a) document frequency and (b) information gain filters on **abstracts** corpus using **Rocchio** learning algorithm

value 1000 for the filter may affect it. It is important to note that even when applying the highest reduction (keeping only 66% of original features) by information gain selection the performance is quite stable yet, as shown by diagrams 12.15b and 12.14b.

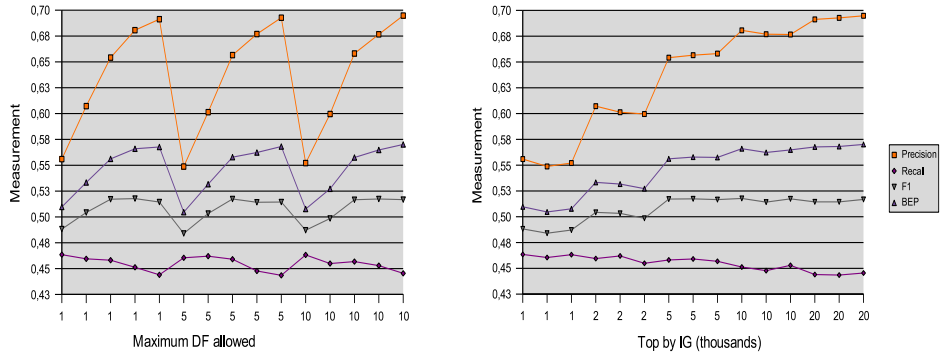


Figure 12.16: Influence of (a) document frequency and (b) information gain filters on **fulltext** corpus using **PLAUM** learning algorithm

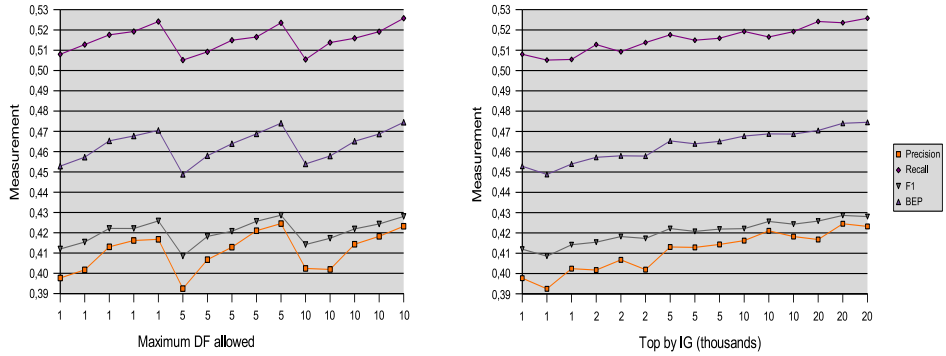


Figure 12.17: Influence of (a) document frequency and (b) information gain filters on **fulltext** corpus using **Rocchio** learning algorithm

For documents with large content, which is the case for our fulltext corpus where a document can contain from 5 to 50 pages, the influence of these two reduction techniques are somewhat different. Document frequency filter does not involve a decay in performance, as shown by figures 12.16a and 12.17a. As for abstracts, we may suppose that at a certain value the classifier should report

worse results, but since the number of features so high (hundreds of thousands of terms) no influence can be noticed using the values studied.

On the other hand, information gain filtering is noticeable does affect when applied on fulltext corpus. For the PLAUM algorithm it reports a constant increase in precision (see figure 12.16b) when more features are kept, while recall decreases and F1 gets stable starting from 5,000 features. For Rocchio, the information gain filter affects negatively the classification at any level (figure 12.17b).

12.5.3 Analysis of results

Summarizing results given above, we can conclude that:

1. For short documents document frequency filter should be set, at most, to 10, since higher values produce bad results.
2. When using PLAUM as classifier, information gain should keep no less than 10,000 features, while for Rocchio 20,000 is the minimum suggested for this filter.
3. In general, the more features we keep, the better the classifier results. But since this is prohibitive in many cases, we should set this parameter just to the point where the computational cost is affordable.

Information gain and *CHI* values were found the best by Yang and Pedersen[130] for dimensionality reduction. By using another mathematical approach based on Information Theory, Koller [60] finds that effectively it is possible to reduce the dimensionality of the feature space without loosing, or even increasing, the accuracy of a classification system.

Information gain filter can drastically reduce the feature space (from hundred of thousands to few thousands) without a big loss in performance. The document frequency based filter is a cheap solution to reduce the dimensionality and, therefore, is also worth considering.

After this study, TECAT has been set to discard all those terms not appearing in, at least, one document at each set (training, evaluation and testing). For information gain, we have set the maximum number of features to 50,000, since for our classification purposes the computational cost is within CERN Document Server constraints.

12.6 EXP 12.6 Evaluating weighting schemes

As described in section 4.2, several weighting schemes have been proposed along time in order to measure the relevance of a feature for a given document. Global,

local and normalization calculations are combined for the final feature weight. The purpose of the present set of experiments is to evaluate such combinations within a limited range (since evaluating all possible formulae would be outside the scope of this work).

As combination of local and global weighting we have tested *TF.IDF* and *entropy* weighting schemes. For the normalization, the *cosine* normalization has been applied. As an additional global factor, the *inverse class frequency* is introduced, to penalize those terms related to too many classes.

12.6.1 Configuration

Five different weighting schemes have been used in these experiments, with three different learning algorithms: *Rocchio*, *Widrow-Hoff* and *PLAUM*. The five schemes have been designed using the following formulae:

1. **TF.IDF.no-cosine** as previous one, but without applying the cosine normalization:

   Inverse document frequency (*IDF*):

   $$idf_i = \log(N/n_i)$$

   TF.IDF:

   $$w_{ij} = f_{ij} \cdot idf_i \tag{12.1}$$

   where

   w_{ij} is the weight of term i in document j
   f_ij is the frequency of term i in document j
   N is the total number of documents in the collection
   n_i is the number of documents in the collection that contain term i

2. **TF.IDF.cosine** as in previous scheme, but without *inverse class frequency* correction; it is one of the most used approaches by the information retrieval community:

   $$w_{ij} = \frac{f_{ij} \cdot idf_i}{\sqrt{\sum_{k=1}^{T} (f_{kj} \cdot idf_k)^2}} \tag{12.2}$$

   where

   T is the number of terms in the collection

3. **TF.IDF.ICF.cosine** The TF.IDF combined with the inverse class frequency correction and the final cosine normalization.

Inverse class frequency (*ICF*):

$$icf_i = \log(M/m_i)$$

TF.IDF.ICF.cosine:

$$w_{ij} = \frac{f_{ij} \cdot idf_i \cdot icf_i}{\sqrt{\sum_{k=1}^{T} (f_{kj} \cdot idf_k \cdot icf_k)^2}} \tag{12.3}$$

where

M is the total number of classes
$m_i, (m_k)$ is the number of classes related to term i, (k)

4. **entropy** has the same formula as above but discarding the inverse class frequency:

$$w_{ij} = \frac{(1 + \log f_{ij}) \left(1 + \sum_{k=1}^{N} \frac{\frac{f_{ik}}{F_i} \log_2 \frac{f_{ik}}{F_i}}{\log_2(N)}\right)}{\sqrt{\sum_{l=1}^{T} \left((1 + \log f_{lj}) \left(1 + \sum_{k=1}^{N} \frac{\frac{f_{lk}}{F_l} \log_2 \frac{f_{lk}}{F_l}}{\log_2(N)}\right)\right)^2}} \tag{12.4}$$

where

N is the total number of documents in the collection
F_i is the total frequency of term i through the entire collection: $F_i = \sum_{j=1}^{N} f_{ij}$
f_{ij} is the frequency of term i in document j
T is the total number of distinctive terms in the corpus

5. **entropy.ICF** is the entropy weighting scheme but with the additional factor of inverse class frequency:

$$w_{ij} = \frac{(1 + \log f_{ij}) \left(1 + \sum_{k=1}^{N} \frac{\frac{f_{ik}}{F_i} \log_2 \frac{f_{ik}}{F_i}}{\log_2(N)}\right) icf_i}{\sqrt{\sum_{l=1}^{T} \left((1 + \log f_{lj}) \left(1 + \sum_{k=1}^{N} \frac{\frac{f_{lk}}{F_l} \log_2 \frac{f_{lk}}{F_l}}{\log_2(N)}\right) icf_l\right)^2}} \tag{12.5}$$

In table 12.16 we can see how TECAT has been parametrized for this set of experiments. Learning algorithm and weighting scheme are modified for each set, selecting one of equations described above. It will produce a total of 15 runs. As usual within TECAT framework, each run consists in a 10-fold cross validation to produce more stable measurements.

indexing		folding	
stop-words	*yes*	DF term filter	*1*
stemming	*yes*	no. folds	*10*
min. term freq.	*0*	IG filter	*50000*
min. term length	*0*	DF class filter	*1*
max. term length	*40*	normalization	*(see weighting)*
		term weighting: *TF.IDF.ICF.cosine, TF.IDF.cosine, TF.IDF.no-cosine, entropy.ICF or entropy*	

learning		testing	
S-cut	*for Rocchio and Widrow-Hoff*	mode	*by class*
measure	*F1*		
threshold	*0.1*		
methods	*Rocchio, PLAUM or Widrow-Hoff*		

Table 12.16: Parametrization of TECAT for experiment 12.6

12.6.2 Results

In tables 12.17, 12.18 and 12.19 results for each learning algorithm are shown. Within each table measurements of precision, recall and F1 are given for the five different weighting schemes under study. The highest values found for these measures are in bold text to ease the analysis.

weighting scheme	**measurements**
TF.IDF.ICF.cosine equation 12.3	Precision 0.418535 Recall 0.522262 F1 0.427662
TF.IDF.cosine equation 12.2	Precision 0.446747 Recall 0.532222 F1 0.452211
TF.IDF.no-cosine equation 12.1	Precision 0.420832 Recall 0.453551 F1 0.343644
entropy.ICF equation 12.5	Precision **0.463947** Recall **0.550939** F1 **0.467335**
entropy equation 12.4	Precision 0.441417 Recall 0.531456 F1 0.459666

Table 12.17: Results for various weighting schemes in experiment 12.6 using **Widrow-Hoff** algorithm

In table 12.18 it may be surprising to observe that the two highest values for both precision and recall are not for the same weighting scheme as the one registering the top F1 measurement. This is due to the fact that F1 is not computed over the two former values, but rather the result of the averaged F1 values found for each document. It is easy to prove that an averaged F1 value is not equal to the F1 measure computed from the averaged values of precision and recall. The reason for not computing a final F1 measure based on the final averaged measures of precision and recall is because we want to provide a global measure for the behavior of a multi-label classifier at *document level*. If for a document either the precision or the recall are zero, then the F1 value will be zero for that document labeling. That is, through all the labellings, precision will be only affected by the values of precision, same for recall; but for F1 a low (and eventually zero) value for any of the two leads to low F1 measurement. This is the reason why averaged F1 values use to be lower than if we computed it with final precision and recall values.

weighting scheme	measurements
TF.IDF.ICF.cosine equation 12.3	Precision 0.468962 Recall 0.553082 F1 **0.470537**
TF.IDF.cosine equation 12.2	Precision 0.469488 Recall 0.546725 F1 0.461763
TF.IDF.no-cosine equation 12.1	Precision 0.450929 Recall 0.488167 F1 0.391756
entropy.ICF equation 12.5	Precision **0.481777** Recall **0.554461** F1 0.469308
entropy equation 12.4	Precision 0.435835 Recall 0.505913 F1 0.424746

Table 12.18: Results for various weighting schemes in experiment 12.6 using **Rocchio** algorithm

weighting scheme	measurements
TF.IDF.ICF.cosine equation 12.3	Precision 0.671883 Recall 0.383264 F1 0.459496
TF.IDF.cosine equation 12.2	Precision 0.692792 Recall 0.413311 F1 0.490548
TF.IDF.no-cosine equation 12.1	Precision 0.408928 Recall 0.349825 F1 0.363149
entropy.ICF equation 12.5	Precision **0.708065** Recall 0.418331 F1 **0.498932**
entropy equation 12.4	Precision 0.652432 Recall **0.419064** F1 0.485063

Table 12.19: Results for various weighting schemes in experiment 12.6 using **PLAUM** algorithm

12.6.3 Analysis of results

As for experiment 12.4 differences in results cannot conclude with a solid preference of one scheme over another. Depending on the learning algorithm used, the best scheme varies. Though we may think of *entropy.ICF* as the better scheme, the difference is less and 1% so such conclusion should be considered carefully. Entropy involves a more complex computation so its use over TF.IDF it is not so clear for us.

From all the experiments carried out, only one definitive conclusion arises: the use of the **cosine normalization produces always better results** no matter the learning algorithm chosen. This is graphically shown on figure 12.18. On the other hand, correction factors like *ICF* seem no to be so effective for TECAT as expected, although in other classification approaches it has represented a significant improvement (like in the *HEPindexer* classifier, see [86]).

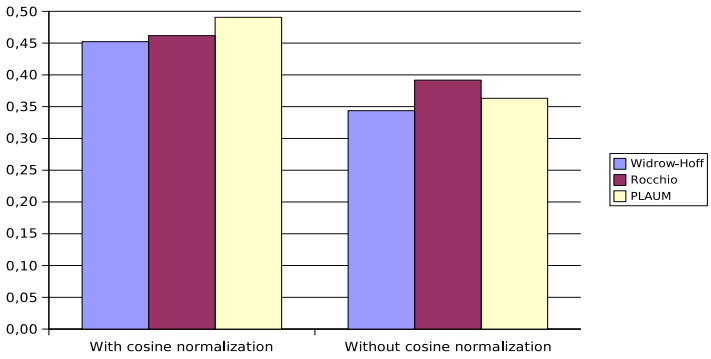


Figure 12.18: Influence of cosine normalization on performance (F1 measure)

As conclusion, the low cost TF.IDF weighting scheme with cosine normalization should be preferred in TECAT for High Energy Physics papers classification.

12.7 EXP 12.7 - Dealing with Imbalance

12.7.1 The class imbalance problem

Usually, multi-labeled collections make use of a wide variety of classes, resulting in an unequal distribution of classes throughout the collection and a high number of rare classes. This means that there is not only a strong imbalance between positive and negative samples, but also that some classes are used much more frequently than other classes. This phenomenon, known as the *class imbalance*

problem, is especially relevant for algorithms like the C4.5 classification tree [22, 35] and margin-based classifiers like SVM [50, 101, 126].

Extensive studies have been carried out on this subject as reported by Japkowicz [50], identifying three major issues in the class imbalance problem: *concept complexity*, *training set size* and *degree of imbalance*. Concept complexity refers to the degree of "sparsity" of a certain class in the feature space (the space where document vectors are represented). This means that a hypothetical clustering algorithm acting on a class with high concept complexity would establish many small clusters for the same class. Regarding the second issue, i.e. the lack of a significantly large training sets, the only possible remedy is the usage of over-sampling when the amount of available samples is insufficient, and under-sampling techniques for classes with too many samples, e.g. just using a limited number of samples for training a SVM, by selecting those positive and negative samples that are close to each other in the feature space. The validity of these techniques is also subject to debate [35]. Finally, Japkowicz defines the degree of imbalance as an index to indicate how much a class is more represented over another, including both the degree of imbalance between classes (what we call *inter-class imbalance*) and between its positive and negative samples (what we call the *inner imbalance degree*). Unfortunately, Japkowicz defined these values for her work towards the generation of an artificial collection and rewrote them later to fit specific problems regarding fixed parameters and the C5.0 algorithm, which make them difficult to manipulate. For these reasons, we cannot reuse her equations and propose here a variant focusing on the multi-label case.

We define the *inner imbalance degree* of a certain class i as a measure of the positive samples over the total of samples:

$$i_i = |1 - 2n_i/n| \tag{12.6}$$

where

n is the total number of samples and

n_i is the total number of samples having the class i in their labels.

Japkowicz' definition of imbalance degree helps in the generation of artificial distributions of documents to classes. Its value does not lie within a defined range, which makes it difficult to manipulate and compare with the degree of other classes in different partitions. On the contrary, the value proposed in equation 12.6 is zero for perfectly balanced classes, i.e. when the number of positive and negative samples are the same. It has a value of 1 when all samples are either positive or negative for that class. Its linear behavior is shown in figure 12.19 and, as we can see, it varies within the range [0,1].

We can now study this collection applying the inner imbalance degree measure defined in equation 12.6. The two graphs in figures 12.20a and 12.20b show the inner imbalance degree for the main key words in the `hep-ex` partition. We

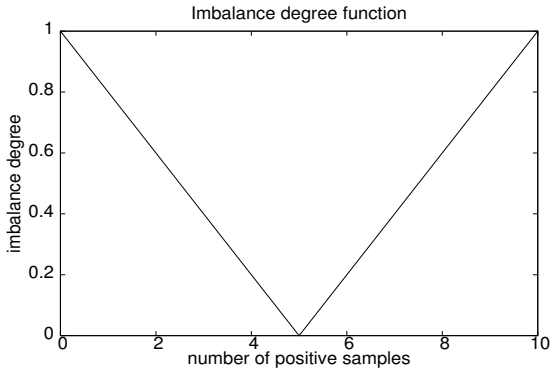
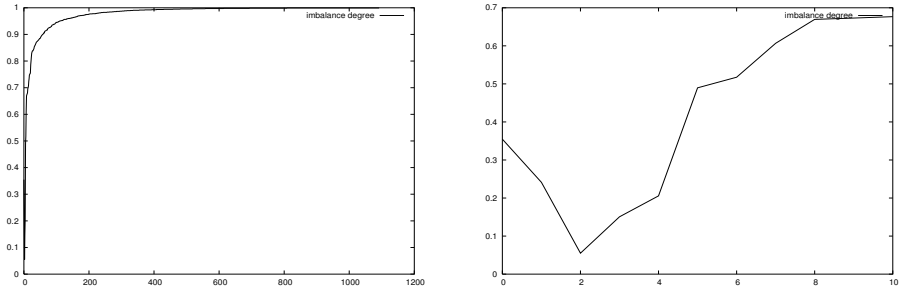


Figure 12.19: The linear 'imbalance degree' function

can notice how fast the imbalance grows to a total imbalance degree of almost 1. When looking at the ten most frequent classes, we can see the effect of our degree estimation: classes 0 and 1 are more imbalanced than class 2, which gets the lowest degree of imbalance in the whole set of classes. It is due to the fact that, as shown by table 11.6, this class has almost the same number of positive and negative samples. From class 3 onwards, the imbalance then grows dramatically.

When training binary classifiers for these key words, we realized that the performance decreases strongly with growing imbalance degree. To correct document distribution across classes, we can use over-sampling (or under-sampling) or tune our classifiers accordingly. For example, for SVM we can set a cost factor, by which training errors on positive samples out-weights errors on negative samples [89]. We will use this in our experiments.



(a) All classes (b) Ten most frequent

Figure 12.20: Imbalance degree of classes in the `hep-ex` partition

12.7.2 Balance weighting and classifier filtering

Some algorithms work better when, in the one-against-all approach, the number of positive samples is similar to the number of negative ones, i.e. when the class is balanced across the collection. However, multi-label collections are typically highly imbalanced. This is true for the HEP collection, but also for other known document sets like the OHSUMED medical collection used in the filtering track of TREC [48], and for the document collection of the European Institutions classified according to the EUROVOC thesaurus. This latter collection has been studied extensively for automatic indexing by Pouliquen et. al. (e.g. [18]), who exploit a variety of parameters in their attempt to determine whether some terms refer to one class or to another in the multi-labeled set.

The question now is how to deal with these collections when trying to apply binary learners that are sensitive to high imbalance degrees. We can use techniques like over-sampling and under-sampling, as pointed out earlier, but this would lead to an overload of non-informational samples in the former case, and to the loss of information in the second case. Furthermore, concept complexity has also its effects on binary classifiers. We have not paid attention to this fact since it is out of the scope of the present study, but we should consider this to be yet another drawback for collections indexed with a large number of non-balanced classes.

In our experiments we basically train a system using the battery strategy, but *(a)*, we allow tuning the binary classifier for a given class by a balance factor, and *(b)* we provide the possibility of choosing the best of a given set of binary classifiers. At CERN, we intend to apply our classification system to *real time* environments so that a gain in classification speed is very important. Therefore, we have introduced a parameter α in the algorithm, resulting in the algorithm given in figure 12.21. This value is a threshold for the minimum performance of a binary classifier during the validation phase in the learning process. If the performance of a certain classifier is below the value α, meaning that the classifier performs badly, we discard the classifier and the class completely. By doing this, we may decrease the recall slightly (since less classes get trained and assigned), but the advantages of increased computational performance and of higher precision compensate for it. The effect is similar to that of the *SCutFBR* proposed by Yang [128]. We never attempt to return a positive answer for rare classes. In the following, we show how this filtering saves us considering many classes without significant loss in performance.

We allow over-weighting of positive samples using the actual fraction of negative samples over positive ones, that is, the weight for positive samples (w_+) is:

$$w_+ = C_-/C_+$$

where

Input:
  a set of multi-labeled training documents D_t
  a set of validation documents D_v
  a threshold α on the evaluation measure
  a set of possible label (classes) L,
  a set of candidate binary classifiers C
Output :
  a set $C' = \{c_1, ..., c_k, ..., c_{|L|}\}$ of trained binary classifiers
Pseudo code:

```
C' ← ∅
for-each l_i in L do
    T ← ∅
    for-each c_j in C do
        train-classifier(c_j, l_i, D_t)
        T ← T ∪ {c_j}
    end-for-each
    c_best ← best-classifier(T, D_v)
    if evaluate-classifier(c_best) > α
        C' ← C' ∪ {c_best}
    end-if
end-for-each
```

Figure 12.21: The one-against-all learning algorithm with classifier filtering

C_- is the total number of negative samples for the class

C_+ is the total number of positive samples for the class

As we can see, the more positive documents we have for a given class, the lower the over-weight is, which makes sense in order to give more weight only when few positive samples are found. This method was used by Morik et al. [89] but they did not report how much it improved the performance of the classifier over the non-weighted scheme. As we said, this w_+ factor was used in our experiments to over-weight positive samples over negative ones, i.e. the classification error on a positive sample is higher than that of a negative one.

We also considered the *S-cut* approach. The assignment of a sample as positive can be tuned by specifying the decision border. By default it is zero, but it can be set using the S-Cut algorithm [128]. This algorithm uses as threshold the one that gives the best performance on an evaluation set. That is, once the classifier has been trained, we apply it against an evaluation set using as possible thresholds the classification values (the margin for SVM). The threshold that reported the best performance (the highest F1 in our case) will be used.

12.7.3 Configuration

The collection consists of 2967 full-text abstracts linked to 1103 main key words. Each abstract was processed as follows:

- Punctuation was removed
- Every character was lower-cased
- Stop words were removed
- The Porter stemming algorithm [93] was applied
- Resulting stems were weighted according to the TF.IDF scheme [108]

After processing the collection in this way, we trained the system applying each strategy using the *SVM-Light*[2] package as the base binary classifier. We also filtered out classes not appearing in any document either in the training, validation or test sets, reducing the number of classes to 443.8 on average. Results are shown at the end of this section. The summary for the TECAT configuration is given in table 12.20.

indexing		**folding**	
stop-words	*yes*	DF term filter	*1*
stemming	*yes*	no. folds	*10*
min. term freq.	*0*	IG filter	*50000*
min. term length	*0*	DF class filter	*1*
max. term length	*40*	normalization	*yes*
		term weighting	*TF.IDF*

learning		**testing**	
S-cut	*depends on run*	mode	*by class*
measure	*F1*		
threshold	*0.1*		
methods	*SVM*		

Table 12.20: Parametrization of TECAT for experiment 12.4

As usual, for the evaluation of experiments, *ten-fold cross validation* [58] was used in order to produce statistically relevant results that do not depend on the partitioning of the collection into training, evaluation and test sets. Extensive experiments have shown that this is the best choice to get an accurate estimate. The measures computed are *precision* and *recall*. The F_1 measure (introduced

[2]SVM-Light is available at `http://svmlight.joachims.org/`

by Rijsbergen [118] a long time ago) is used as an overall indicator based on the two former ones and is the reference when filtering is applied. Also *accuracy* and *error* measurements are given for later discussion. The final values are computed using macro-averaging on a per-document basis, rather than the usual micro-averaging over classes. The reason is, again, the high imbalance in the collection. If we average by class, rare classes will influence the result as much as the most frequent ones, which will not provide a good estimate of the performance of the multi-label classifier over documents. Since the goal of this system is to be used for automated classification of individual documents, we considered to be far more useful to concentrate on these measurements for our evaluation of the system. More details about these concepts can be found in [112], [63] and [129].

12.7.4 Results

Table 12.21 shows the results of ten runs of our multi-label classifier with different configurations. The highest values of F_1 are reached when the system chose among fixed values for over-weighting positive samples (2, 5, 10 and 20). These are the results when applying the algorithm of figure 12.21 with $\alpha = 0.0$, i.e. no filtering over classifiers is done.

Experiment	Precision	Recall	F1	Accuracy	Error	% of classes
No weight	74.07	33.96	43.92	98.23	1.77	33.96
No weight / Scut	**74.26**	34.44	44.38	98.24	1.76	99.95
Overweight 20	51.47	**45.84**	46.50	97.71	2.29	57.32
Auto weight	58.10	44.39	48.09	97.94	2.06	58.09
Overw. 2,5,10,20 / Scut	71.74	39.92	48.47	**98.25**	**1.75**	**100.00**
Auto weight / Scut	58.03	45.30	48.56	97.89	2.11	99.82
Overweight 2	70.74	40.45	48.78	98.21	1.79	53.36
Overweight 5	64.56	43.57	49.40	98.11	1.89	57.19
Overweight 10	62.30	45.22	50.14	98.08	1.92	57.30
Overw. 2,5,10,20	65.89	44.59	**50.53**	98.17	1.83	57.53

Table 12.21: Results of experiments using SVM **without filtering**

We see that the top recall reached does not imply having more classes trained. Therefore we may want to study how we can reduce the number of classes trained to speed up the classification process without loosing too much in performance. For that purpose, we experimented with different values of α, as shown in tables 12.22 and 12.23.

α	0.0	0.1	0.2	0.3	0.4	0.5	0.6	0.7
Precision	65.89	70.04	70.41	70.88	71.90	71.96	71.02	67.96
Recall	44.59	44.49	43.95	42.95	40.54	36.65	31.80	23.02
F_1	50.53	**51.59**	51.32	50.77	49.21	46.11	41.70	32.83
Accuracy	98.17	98.25	98.25	98.25	98.24	98.21	98.15	98.03
Error	1.83	1.75	1.75	1.75	1.76	1.79	1.85	1.97
% classes trained	57.53	56.49	50.81	43.20	32.73	23.23	16.00	8.58

Table 12.22: Results of experiments using **multi-weighted** SVM with filtering

α	0.0	0.1	0.2	0.3	0.4	0.5	0.6	0.7
Precision	58.03	62.47	64.84	67.45	69.47	71.19	71.14	68.24
Recall	45.30	45.04	44.83	44.24	42.76	39.59	34.43	24.88
F_1	48.56	49.93	50.47	50.75	50.27	48.37	44.10	34.76
Accuracy	97.89	98.06	98.14	98.20	98.23	98.22	98.17	98.05
Error	2.11	1.94	1.86	1.80	1.77	1.78	1.83	1.95
% classes trained	99.82	85.30	77.10	68.47	55.74	42.34	30.82	16.72

Table 12.23: Results of experiments using **auto-weighted S-Cut** thresholding SVM with filtering

12.7.5 Analysis

Interesting conclusions can be drawn from the tables above. The first thing we notice is that recall is low compared to precision. This is normal if we consider the existence of rare and, therefore, difficult-to-train classes. When tuning our multi-label classifier, we see that variations in precision are more representative than for recall. The F_1 measure remains quite stable: throughout all the experiments with different configurations, the most we gain is 6.61%. However, a very important result is that, even when some configurations are able to train up to 100% of the total of classes involved (we can see how the percentage of classes successfully trained varies widely), it does not influence that much the overall performance of the classifier. We can conclude that *rare classes are not worth training*. This is the reason for the design of our filtering algorithm. Furthermore, it is not clear that S-Cut and auto-weighting strategies are so relevant for our data. As we can also notice, accuracy and error are not very sensitive to the variations of our parameters, but this is again due to imbalance: most of the classes are rare and for the most frequent ones we get high precision and recall, even with not very sophisticated configurations.

When discarding classes, we obviously gain in precision and, despite more classes not being trained, we do not lose that much in recall. The result is a better F_1 than without discarding, as shown by F1 values in 12.21 compared to those of tables 12.22 and 12.23. We can see how strongly we can reduce the number of classes without affecting significantly the overall performance of the multi-label classifier. Figures 12.22 and 12.23 visualize the behavior described. The bigger our α is, the more classes are discarded. From all the test runs, the best value of F_1 was obtained with an α value of 0.1 and using candidate classifiers with over-weights 2, 5, 10 and 20 for positive classes. From the graphs we can see that increasing α yields to a higher precision up to a maximum from which the threshold will be so restrictive that even good classifiers are discarding and, therefore, the precision starts to decrease accordingly. Thus, our choice of α will depend on our preference of precision over recall and our need of reducing classes for faster classification. If we are able to discard non-relevant (rarely used) classes, we can almost maintain our performance even classifying against a lower number of classes.

The frequency of use of a key word determines the capability of the system

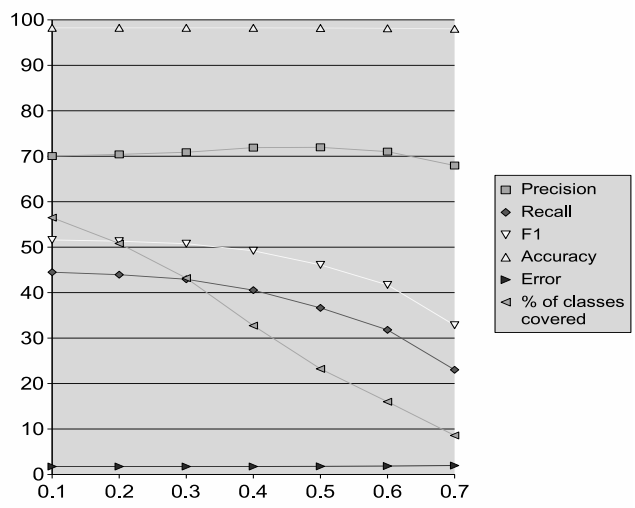


Figure 12.22: Influence of filtering on **multi-weighted** SVM with S-cut thresholding

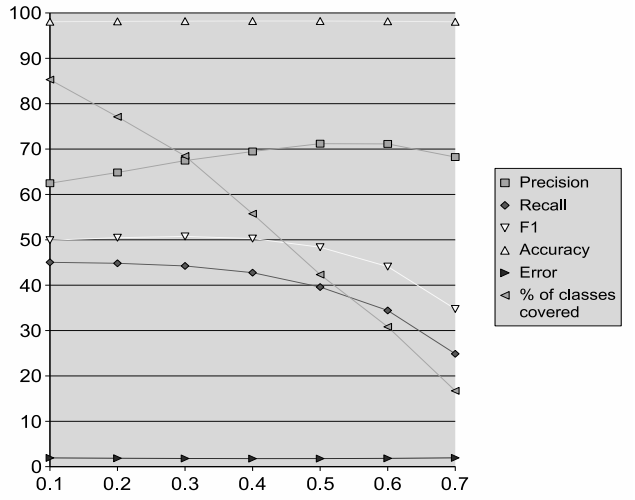


Figure 12.23: Influence of filtering on **auto-weighted** with S-cut thresholding

to learn for it. When a class is balanced, that is, when it appears in about 50 % of the documents, then our training data should be enough for the learning algorithm to produce good classifiers. Of course, the entropy of the class itself has strong influence on the final classification capability. We have run TECAT

using the PLAUM classifier and calculated average measurements for 100 most frequent classes (the most used classes for labeling documents). A graph showing the different performance measures depending on the number of documents a class belongs to is shown in figure 12.24. Classes has been sorted by number of documents, so the more we move to the right, the more frequent the class is.

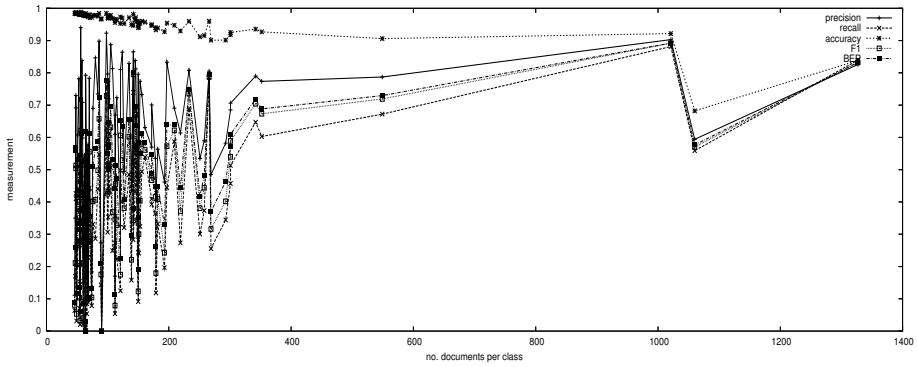


Figure 12.24: Influence of the frequency of use of a class on the performance obtained for that class

From this diagram, whose results were obtained on the *hep-ex* partition, with a total number of documents of 2780 after filtering those with non trainable labels, many interesting aspects can be pointed out:

1. For very infrequent classes, accuracy is always very high, since they are so rare that not considering them do not report any important loss.

2. When there are few document per class, performances obtained are quite random: sometimes we can train with good results, sometimes results are really bad. In general, we cannot ensure a lower bound for performance when the frequency of use of a class is less than 200 documents. On the other hand, from a frequency of about 350 documents, all the measures stay above 50%. This is an obvious observation: we obtain better performance when we have more documents labeled for every class, and these oscillations get more stable, with a higher lower bound when the number of documents associated to the class grows.

3. In the diagram we can notice an estrange behavior for the class with a document frequency of about 1050: performance suddenly falls down (though not dramatically). There is a very easy explanation for such phenomenon, since it is related to the nature of the class itself. The main key-word associated is `experimental results`, and such key-word is very complex to determine. Human experts assign it depending on the purpose of the related document. Documents showing experimental results in HEP

for the first time to the scientific community (not for later discussion) are labeled with such key-word. Therefore, other documents may show the same number of diagrams, numerical explanations and so on, but they may not get that label. In this case, another approach could be proposed, like studying the number of diagrams in a paper or the discourse structure, for example.

12.7.6 Conclusions and future work

A calculus for measuring the imbalance degree has been proposed, along with a study of the overweight of positive classes on this collection using SVM and the application of S-Cut. The results show that this is a relevant issue, and that an imbalance study of any multi-label collection should be carried out in order to properly select the base binary classifiers. Another promising issue would be to work on other aspects of imbalance like *concept complexity* [51]. We have started studying this topic by working with "concepts" rather than with terms in order to reduce the term space. By doing this, we would cover the main drawbacks of imbalanced collections. S-cuts on SVM increases drastically the number of classes trained by the system without relevant improvement on performance, bu a consequent penalization on classification speed and storage requirements, so it is not recommendable for SVM.

Filtering by classification thresholding is very effective to reduce the number of classes involved in multi-label classification. Without forcing expensive tuning of the threshold, we propose to provide a range of α values and let the algorithm choose the classifier with the best behavior.

One of the disadvantages using the battery approach is its computational cost, since we have to launch every classifier for a sample. However, SVM is quite selective, not being trainable in many cases, discarding in this way many conflicting classes. This reduces the computation without loosing too much in performance. We have shown that, by increasing the selectivity, we can even gain significantly in precision without loosing too much in recall.

One multi-label collection issue we have not considered is inter-class dependency. In some preliminary analysis we found that the correlation among classes is relevant enough to be considered. We could actually benefit from such a correlation to speed up the classification process, by discarding those classes not correlated to the ones we have already found relevant. This relation could probably be used to fight one of the drawbacks found: our recall is very low compared to the precision. If we were able to select those classes that are highly correlated with classes assigned with high precision, we might gain in recall. This will need further investigation.

All these interesting conclusions have been reported to the scientific community in a publication by Montejo-Ráez, Steinberger and Ureña-López [88].

12.8 EXP 12.8 - Integrating meta-data information

Corpora available tend to be not as accessible and complete as the research community wishes. Well known collections as *Reuters-21578*[3], *OHSUMED* [48] (used in TREC evaluation forum) or *20 Newsgroups*[4] show, for each sample, sort fragments of text. Instead, within EUROVOC related experiments ([18, 99, 100]) full text documents are used as sample data for classifier training. But, in most of the experiments arranged based on these collections, just plain text data from main content of the document is used. Words are then stemmed or lemmatized, counted and weighted following a defined indexing scheme.

But data from digital libraries is much richer than all that: digital libraries contain *metadata*, i.e. additional information about every stored document. Metadata is *data about data*: author of a document, date of publication, storing format, identifier in the database, publisher, length and so on. The *Dublin Core Metadata Initiative*[5] (DCMI) is a open standard for adding information to documents in digital libraries. The *Open Archive Initiative*[6] (OAI) aims to establish a standard for document exchange between different digital libraries and a protocol for harvesting and retrieving of documents from database supporting it. OAI also support MARC[7] format for metadata. DCMI is also used as a source of entities for the *Semantic Web*[8] project [15]. As we can see, metadata is something to care about.

For HEP papers stored at CERN, the MARC format is used (although full accessing through OAI has been recently integrated). A document record sample is shown in figure 12.25. This sample has been obtained from the CERN Document Server database. For more information about the HEP collection see chapter 11.

We can see in those figures the amount of additional data available. The target of this experiment is to test whether the use of additional data (apart from content) can improve classification performances. Due to the profusion of meta-data entries in the records of current digital libraries, a positive answer is an important clue to build enriched classifiers.

12.8.1 Configuration

In table 12.24 we show the configuration used for this experiment. An explanation of the parameters involved is given in section 10.6. For this set of

[3] Prepared by David D. Lewis. The collection is freely available from the web page `http://www.research.att.com/~lewis/reuters21578.html`

[4] Available at `http://kdd.ics.uci.edu/databases/20newsgroups/20_newsgroups.tar.gz`

[5] `http://dublincore.org/`

[6] `http://www.openarchives.org`

[7] `http://www.loc.gov/marc/`

[8] `http://www.w3.org/2001/sw/`

```
<?xml version="1.0" encoding="UTF-8"?>
<collection>
    <dc xmlns="http://purl.org/dc/elements/1.1/"
          xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
            xsi:schemaLocation="http://purl.org/dc/elements/1.1/
                                http://www.openarchives.org/OAI/1.1/dc.xsd">

        <language>eng</language>
        <creator>Kitamura, T</creator>
        <title>Theory of liquid-glass transition in water</title>
        <subject>Condensed Matter</subject>
        <identifier>http://documents.cern.ch/[...]id=0503152</identifier>

        <description>A quantum field theory of the liquid-glass
        transition in water based on the two band model in the
        harmonic potential approximation is presented by taking into
        account of the hydrogen bonding effect and the polarization
        effect. The sound and diffusion associated with intra-band
        density fluctuations, and the phonons and viscocity associated
        with inter-band density fluctuations are calculated. The
        Kauzmann paradox on the Kauzmann's entropy crisis and the
        Vogel-Tamman-Fulcher (VTF) law on the relaxation times and the
        transport coefficients are elucidated from the sound
        instability at a reciprocal particle distance corresponding a
        hydrogen bond length and at the sound instability temperature
        very close to the Kauzmann temperature. The gap of specific
        heat at the glass transition temperature and the boson peaks
        are also presented.</description>

         <date>2005-03-09</date>
    </dc>
</collection>
```

Figure 12.25: Sample for metadata information in XML DC format.

experiments the *hep-ex* partition was used.

The configuration passed to TECAT for this experiment is defined in table 12.24. Four different algorithms have been taken as base classifiers (but not simultaneously): *Widrow-Hoff*, *Rocchio*, *PLAUM* and *SVM*. At each of these algorithms, a group of corpora of data has been used for learning and testing. These corpora are a combination of full-text information with available metadata:

- **Source A: Abstracts**. Experiments with given algorithms have been carried out on abstracts (the `description` field in XML DC form).
- **Source M: Metadata**. Each document is composed by a combination of plain-text data (`abstract` and `title`) with fixed values (`date`, `subject`, `creator` and `language`).
- **Source F: Full-text**. The corpus is composed by the full-text version of documents (extracted from PDF versions of each paper).

indexing		folding	
stop-words	*yes*	DF term filter	*1*
stemming	*yes*	no. folds	*10*
min. term freq.	*0*	IG filter	*50000*
min. term length	*0*	DF class filter	*1*
max. term length	*40*	normalization	*yes*
		term weighting	*TF.IDF*

learning		testing	
S-cut	*yes/no* [a]	mode	*by class*
measure	*F1*		
threshold	*0.1*		
methods	*Rocchio, Widrow-Hoff, SVM or PLAUM*		

[a] Scut is applied on Widrow-Hoff and Rocchio algorithms, but not active on PLAUM nor SVM

Table 12.24: Parametrization of TECAT for experiment 12.8

- **Source F+M: Full-text and metadata**. This is the most complete corpus, where plain-text data is built up from full-text version, `title` and `abstract`, and combined with fixed fields like `date`, `subject`, `creator` and `language`.

It is important to point out that the processing of plain-text data differs from that applied on fixed fields. These fields have been protected to not be affected by stopping and stemming procedures. The rest of operations are performed on these fields, considering each entry as an additional feature.

Four measures have been studied to determine whether a source outperforms another. These measures are *precision*, *recall*, *F1* and *accuracy*. F1 provides a more general view than precision and recall separately, and the accuracy provides a even more general measure of the goodness of the method (though not very preferred in text processing tasks). However, precision and recall has been studied alone to inspect how a source affects each one of these two important indicators.

12.8.2 Results

The description of experiments is described as follows: for each source (corpus), we have run four different experiments. Each of such experiments consists in using a different unique base classifier as detailed above. As product, we will have sixteen sets of results. The reason to use different algorithms is to avoid dependencies on the type of learning method applied. Therefore, results from different algorithms will be considered together and compared to those for experiments with different corpus used. At each run the selected four measures have been obtained for each one of the ten most frequent key words (see table 12.25).

experimental results
magnetic detector
talk
electron positron
quark
CERN LEP Stor
anti-p p
Z0
Batavia TEVATRON Coll
mass spectrum

Table 12.25: Ten most frequent categories

We will perform a Wilcoxon Signed Ranked test. This is a non-parametric test that let us know if two distributions of the same variable are statistically different (a description of this test is given at appendix C). We have used *Octave*[9] to compute the *p-values* returned by the test. In our comparison, we will consider only p-values under 0.05, that is, the probability for the two given distributions to differ not by chance is higher than 95%. Since we have run a 10-fold cross-validation mechanism within TECAT, we have averaged the measured values of precision, recall, F1 and accuracy for the ten most frequent key words. Therefore, to compare source A (abstracts) against source F (full-text), we create two distributions with 40 values each (4 algorithms $\times$ 10 measured key words). Then, we can construct the comparison matrices (one per measure) with Wilcoxon p-values shown at tables 12.26, 12.27, 12.28 and 12.29.

over precision	**A**	**M**	**F**	**F+M**
A	0.50000000	0.06272324	**0.00014218**	**0.00004588**
M	0.93727676	0.50000000	0.07907275	**0.02646161**
F	0.99985782	0.92092725	0.50000000	0.09387790
F+M	0.99995412	0.97353839	0.90612210	0.50000000

Table 12.26: Tailored Wilcoxon test over precision

[9]Octave is a high-level language, primarily intended for numerical computation that is available under the GPL. `http://www.octave.org`

over recall	**A**	**M**	**F**	**F+M**
A	0.50000000	**0.00000002**	**0.00184614**	**0.00000104**
M	0.99999998	0.50000000	0.99976274	0.73181109
F	0.99815386	**0.00023726**	0.50000000	**0.00000136**
F+M	0.99999896	0.26818891	0.99999864	0.50000000

Table 12.27: Tailored Wilcoxon test over recall

over F1	**A**	**M**	**F**	**F+M**
A	0.50000000	**0.00000011**	**0.00001842**	**0.00000008**
M	0.99999989	0.50000000	0.97666661	0.22179235
F	0.99998158	**0.02333339**	0.50000000	**0.00000007**
F+M	0.99999992	0.77820765	0.99999993	0.50000000

Table 12.28: Tailored Wilcoxon test over F1

Wilcoxon can be one-tailored or two-tailored. For one-tailored test we can conclude if a difference exists, but we cannot say anything about the *direction* of it. It is our interested to find if a certain source of data provides better results *over* another source of data. Thus, we are interested in a one-tailored analysis that will let us know if one source outperforms another source. This is how it has been computed in the given tables.

The macro-averaged values by document obtained are given in table 12.30, from which we can see the measured values over all trained classes. At this table, each row shows the measures of performance for a given algorithm by using a given source. For example, the entry *Rocchio M* specifies measured macro-averaged values by document through classes and cross-validated folds using as source the metadata based corpus and applying Rocchio algorithm as base classifier (with S-cut thresholding as detailed previously in configuration description).

Many classes are just discarded in the filtering process or due to fact that no possible algorithm can be trained with a minimum performance (the percentage column provides useful information about it). It is interesting to note how different algorithms are more "trainable" than others at this point.

over accuracy	**A**	**M**	**F**	**F+M**
A	0.50000000	**0.00007519**	**0.00012807**	**0.00000665**
M	0.99992481	0.50000000	0.66154384	**0.04778496**
F	0.99987193	0.33845616	0.50000000	**0.00000665**
F+M	0.99999335	0.95221504	0.99999335	0.50000000

Table 12.29: Tailored Wilcoxon test over accuracy

Precision	Recall	F1	Accuracy	Error	% classes	Experiment
0,442927	0,540300	0,455324	0,972086	0,027914	84,43	Widrow-Hoff A
0,441282	0,555588	0,464150	0,972381	0,027619	85,00	Widrow-Hoff F
0,455313	0,553409	0,466323	0,972133	0,027867	84,54	Widrow-Hoff M
0,463957	0,571303	0,480547	0,973005	0,026995	87,27	Widrow-Hoff F+M
0,471174	0,542124	0,461181	0,972455	0,027545	88,16	Rocchio A
0,427285	0,523589	0,428148	0,969635	0,030365	86,12	Rocchio F
0,452551	0,560559	0,456976	0,970048	0,029952	86,18	Rocchio M
0,442462	0,541912	0,443245	0,970011	0,029989	87,92	Rocchio F+M
0,691184	0,410466	0,489537	0,981950	0,018050	52,48	PLAUM A
0,710694	0,434822	0,511888	0,982663	0,017337	57,67	PLAUM F
0,720159	0,448690	0,526646	0,982828	0,017172	55,67	PLAUM M
0,725469	0,452998	0,531442	0,983174	0,016826	59,27	PLAUM F+M
0,745852	0,333645	0,434222	0,982223	0,017777	31,14	SVM A
0,754986	0,357410	0,459033	0,982669	0,017331	35,03	SVM F
0,773406	0,351618	0,458678	0,982704	0,017296	32,23	SVM M
0,769740	0,373289	0,477560	0,983198	0,016802	36,62	SVM F+M

Table 12.30: Macroaveraged measures for all classes in presented experiments

12.8.3 Analysis of results

First, we remind the target of these experiments: to study how the use of different sources of data affects the performance of a multi-label categorization engine. Not only it is interesting to identify which combination of sources is best, but also to study whether is worth using one over another. For example, the computation cost of using full-text documents is much higher than that of using just small abstracts: the collection is smaller, the trained data demands lower storage space, and the classification process is accelerated. Thus, it is very important in our domain to state such facts.

To understand these results let's take, for example, from table 12.29, the p-value corresponding to row **F** and column **F+M** is very small: 0.00000665. It says that when using the corpus created as combination of full-text and metadata, we can be sure (99.9994% sure!) of obtaining more accurated results than those obtained by feeding the system with corpus based on just full-text data. P-values show a *transitive* behavior: let a, b and c, be distinctive sources, and '>' the binary operator to indicate that a source provides better results than another, then, we can state that:

$$(a \geq b) \wedge (b \geq c) \Longrightarrow a \geq c \tag{12.7}$$

At table 12.29 **M** outperforms **A**, and **F+M** outperforms **M**, therefore, we can observe how **F+M** outperforms widely **A**.

It is possible to illustrate graphically the improvement obtained when using some sources instead of others. Figure 12.26 shows four diagrams, one per measure. At each diagram the considered sources are drawn, with their 40 measured values sorted to ease the visual comparison between sources. In principle, the use of only the abstract of a document seems to be the worst choice. Actually, this is not an easy question, as we will argue from Wilconox test results.

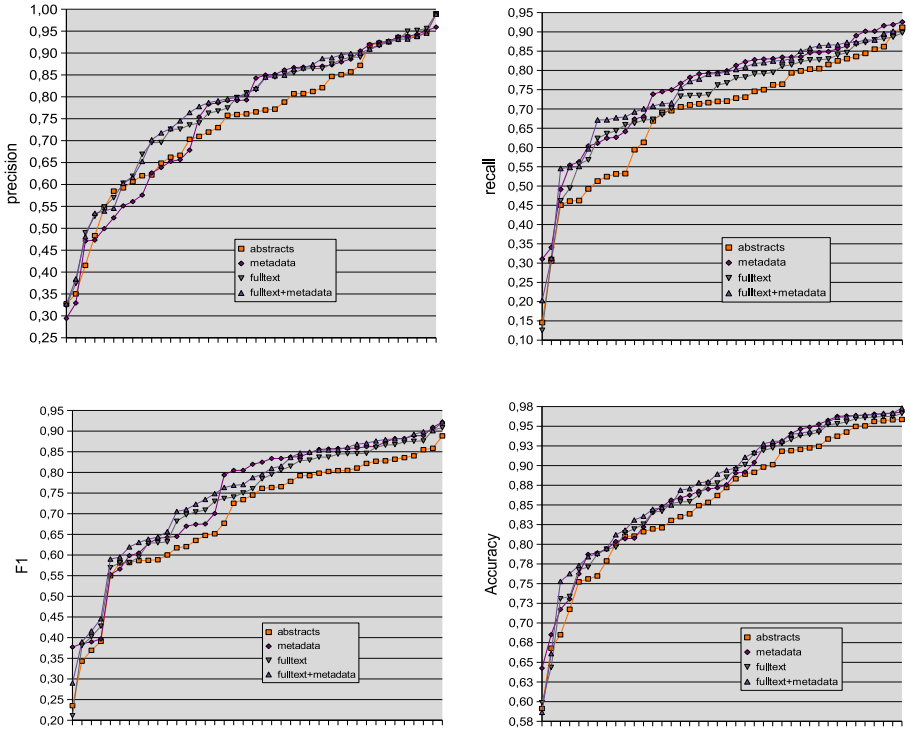


Figure 12.26: Comparison of sorted measures for each source over precision, recall, F1 and accuracy

As said before, we can notice an increase in performance just having a look at figure 12.26, but the statistical values contained in tables 12.26, 12.27, 12.28 and 12.29 provide us solid evidences on the studied subject. After reviewing every table we can conclude with the following points:

1. The combined use of metadata and full-text information is the best choice in any case (though the most costly).
2. Metadata is also good choice in most cases despite the fact is outperformed by the combination of full-text and metadata when we are more interested in precision.
3. It even outperforms corpus based on full-text papers for recall and F1.
4. As main conclusion, **the metadata source should be preferred** due to its reduced computational cost and its good behavior against full-text or the combination of full-text and metadata (except for precision matters).

These interesting results have been recently reported [97]. Therefore, adding a big full-text content of a document (extracting it from the PDF version) to the features that can be promptly extracted from the document record stored in the database is not worthy. This conclusion validates the visual conclusion obtained from inspecting diagrams shown at figure 12.26, where we can see how the curve for the metadata source is usually above the rest of curves. This effect is less clear for precision, as reported by the test.

12.9 EXP 12.9 - Selection: Ranking versus Boolean

In section 6.1 different solutions for multi-labeling by combining base binary classifiers were introduced. TECAT has been programmed to support two possible strategies: by *rank* and by *thresholding*. The former one consists on the selection of the top classes ranked by their classification status value (CSV), i.e. the value returned by the binary classifier for the class. Thresholding, as done by TECAT, delegates to the binary classifier the decision on whether the class should be assigned to the document or not. We may think that this second strategy would provide a high number of labels and force us to loose the control on the number of classes to be attached to a certain document.

In this set of experiments we have studied the performance of both strategies over two possible base classifiers: *PLAUM* and *Rocchio*.

12.9.1 Configuration

As usual for previous experiments, parameters passed to TECAT for present experiments are detailed, see table 12.31.

indexing		folding	
stop-words	*yes*	DF term filter	*1*
stemming	*yes*	no. folds	*10*
min. term freq.	*0*	IG filter	*50000*
min. term length	*0*	DF class filter	*1*
max. term length	*40*	normalization	*yes*
		term weighting	*TF.IDF*

learning		testing	
S-cut	*for Rocchio*	mode	*by class or by rank*
measure	*F1*		
threshold	*0.1*		
methods	*Rocchio or PLAUM*		

Table 12.31: Parametrization of TECAT for experiment 12.9

The corpus used was the *hep-ex* partition of abstracts documents. For the rank strategy the number of top classes selected was modified to produce 5 different runs over each base algorithm: {5, 10, 15, 20, 20} were set as the number of classes for each document. Since we wanted to compare it also against the Boolean strategy where all positive class are returned, we have in total 12 runs of the multi-label classifier over the corpus. Again, 10-fold cross validation framework was applied.

12.9.2 Results

We can briefly summarize the results obtained listed in table 12.32 by graphically presenting them as in figures 12.27 (for PLAUM algorithm) and 12.28 (for Rocchio algorithm). As we can see at first sight, the behavior strongly depends on the algorithm used.

It is, again, noticeable the discriminative behavior of PLAUM algorithm (similar to that of SVM): only 53.74 % of classes were successfully trained by the PLAUM learning algorithm, while 88.74 % of classes were trained by Rocchio algorithm.

In general, an increment in the number of classes obtained reports a higher recall value (obviously), but is punished by a drastic fall of precision. It is curious how all measures converge when the number of classes is 10, due to the fact that precisely 10 is the average number of classes per document in the corpus.

n-top ranked	Precision	Recall	F1	BEP	Accuracy	Algorithm
all positive	0.472300	0.543758	**0.461417**	0.508029	0.972009	Rocchio
5	0.255107	0.130831	0.166047	0.192969	0.970028	Rocchio
10	0.219876	0.221002	0.212103	0.220439	0.962849	Rocchio
15	0.196147	0.290238	0.226044	0.243193	0.954855	Rocchio
20	0.175423	0.342289	0.224755	0.258856	0.946054	Rocchio
50	0.107944	0.510858	0.174812	0.309401	0.886364	Rocchio
all positive	0.691092	0.412173	0.489869	0.551632	0,981761	PLAUM
5	0.703932	0.356124	0.457544	0.530028	0.980245	PLAUM
10	0.539713	0.526894	**0.516098**	0.533304	0.977412	PLAUM
15	0.415165	0.597558	0.475219	0.506362	0.969811	PLAUM
20	0.332678	0.633238	0.424067	0.482958	0.960371	PLAUM
50	0.150648	0.709563	0.244088	0.430106	0.896079	PLAUM

Table 12.32: Performance measures registered for PLAUM and Rocchio algorithms using ranking strategy

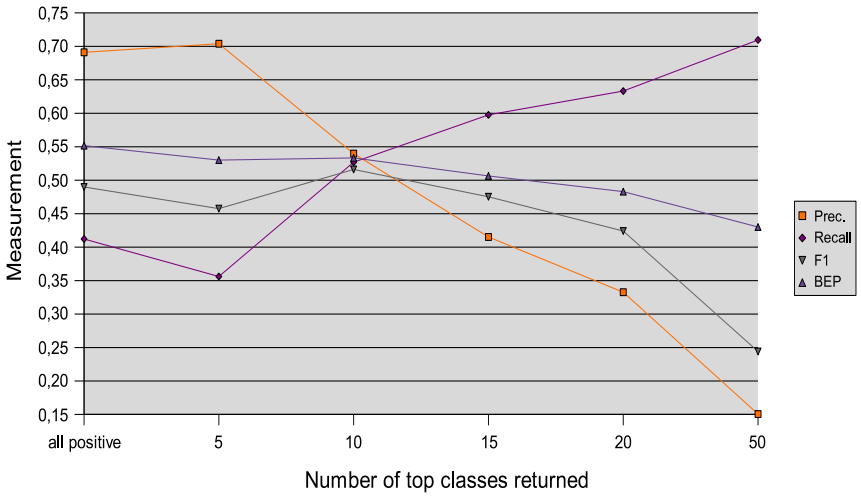


Figure 12.27: Rank strategy results for **PLAUM** algorithm

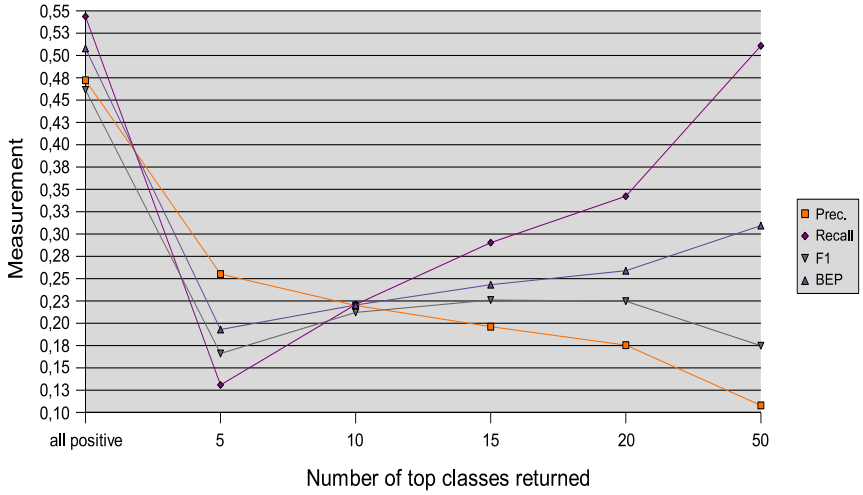


Figure 12.28: Rank strategy results for **Rocchio** algorithm

12.9.3 Analysis of results

The main conclusion is straightforward: the rank strategy is not a good solution for merging classifiers results into final set of labels. Although we can control whether precision should be penalized in favor of a higher recall, specifying a fixed set of final classes is not a recommended option. This is due to the fact that we may select classes that were refused by the associated binary classifier, and symmetrically, some classes found positive by the originator classifier my be discarded. That is, the rank strategy is a "blind" strategy, since it only consider the CSV value returned by the classifier, but not the internal threshold that the classifier may use to determine the suitability of a class for the document.

Moreover, the rank strategy is only applicable when CSV values are comparable, and that is a very difficult question to answer: even when using the same learning algorithm, the classifier obtained after training it for a class may not be comparable with the same algorithm for another class. Also, some algorithms like margin based ones (SVM and PLAUM, for instance) produce CSVs that could not be considered as distance measure of the document to the class. An maximum on F1 measure is observable for PLAUM when taking 1. This higher F1 value over the one registered when all positive ones are returned is consequence of the behavior of the PLAUM algorithm, which reports usually higher values of precision than for recall. By taking top 10 classes always (even when some of them may be negative) we are penalizing precision in favor of the recall index, so we register that overall increment in F1 measure.

Therefore, we recommend to return all classes which are result of a posi-

tive evaluation of the classifier, so the number of classes per document is more dynamic and adjusted to the relative performance of each base algorithm.

12.10 EXP 12.10 - Tests over additional HEP corpora

Mainly in our experiments, we have used the *hep-ex* abstracts partition. In some of them also the full-text file for those documents has been fed into the system for further studies. Once we know that TECAT can perform well for the base line `hep-ex` collection, it is important to report the performance of the tuned system against different corpora. To do so, two additional partitions of the originally provided HEP collection have been source of documents for training and testing: the *astro-ph* and *hep-th*[10] partitions. Details about both collections are given in next section.

If we find similar performance of the TECAT system on these two corpora, we are in a better position to extrapolate the benefits of our system to the full of HEP digital library.

12.10.1 Configuration

In table 12.33 the basic configuration of the TECAT system is detailed. As we can see, most of the parameters are the commonly used values for most of the experiments presented so far. This configuration provides a performing training of the system in order to execute real-time classification. If there is one parameter that may be affected by the size of the collection is the number of features to be filtered by its information gain value. As we saw in experiment 12.5, this number affects performance. For these runs, the number of features to be filtered by this reduction method has been varied according to three possible values: {10000, 20000, 50000}, referred in our experiments as *IG-A*, *IG-B* and *IG-C* respectively.

As proved by experiment 12.8, the use of abstracts combined with additional metadata information significantly improves classification performances. Those directives have been followed here. Therefore, each corpus has been generated not only from abstracts, but also incorporating into the content of the document its title, authors, date of publication, subject and language (though this last one is irrelevant: all documents are in English).

[10]These corpora were also provided by the CERN Document Server for its use in these experiments, so we can use as benchmark document collections related to a different subject from pure HEP experimental papers. More details about them are given later.

indexing		folding	
stop-words	*yes*	DF term filter	*1*
stemming	*yes*	no. folds	*10*
min. term freq.	*0*	IG filter	*10000, 20000 or 50000*
min. term length	*0*	DF class filter	*1*
max. term length	*40*	normalization	*yes*
term weighting	*TF.IDF*		

learning		testing	
S-cut	*no*	mode	*by class*
measure	*F1*		
threshold	*0.1*		
methods	*PLAUM*		

Table 12.33: Parametrization of TECAT for experiment 12.10

12.10.2 Results

Two of the main differences between different corpora are *(a)* the number of documents and *(b)* the relative number of classes covered by those documents. As show by figures 12.29a and 12.29b, consequently the more documents are in a collection, the larger the number of classes covered is. But as we can see from those figures, this relation is not proportional: *hep-th* has eight times more documents than the *hep-ex* partition (figure 12.29a), but they almost cover the same number of classes. In table 12.34 these values are presented: the *number of classes tested* (that is, the number of classes found in the corpus), the *number of classes trained* (classes successfully modeled by a binary classifier), and the number of documents.

collection	classes tested	classes trained	no. docs
astro-ph	296	124	2766
hep-ex	441	258	2839
hep-th	518	313	17270

Table 12.34: Relation of classes and documents by corpus

Figure 12.30 shows performance measures registered by TECAT tests with modified number of features allowed by the information gain filter for each corpus. The most sensitive value is, again, the precision of classification. Also recall, F1, and break-even-point (*BEP*, here just a simple mean for precision and recall) are given to show the effect of varying the filtering parameter.

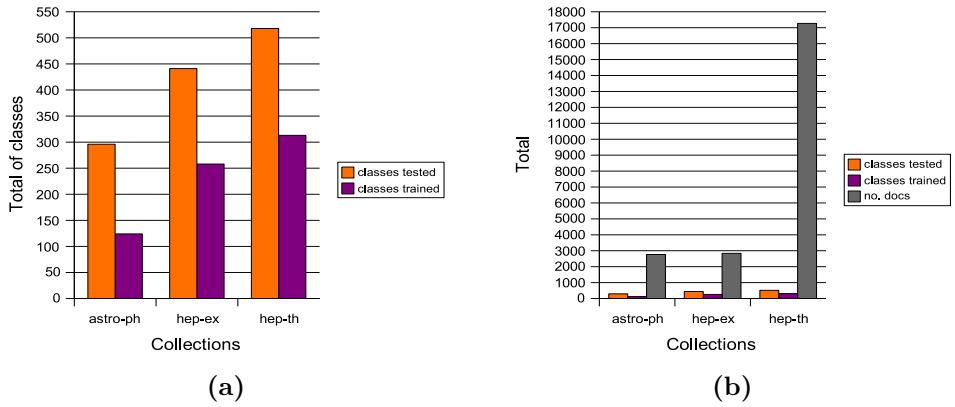


Figure 12.29: (a) Number of classes trained and tested and (b) number of docs for each collection

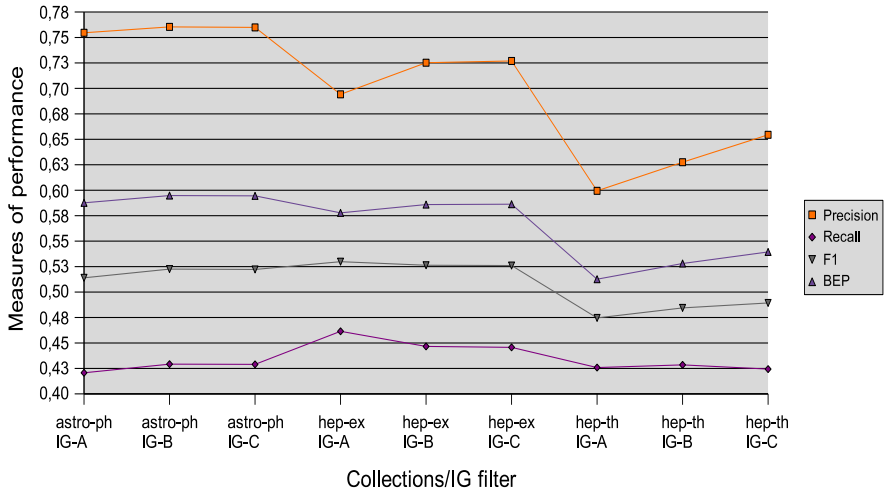


Figure 12.30: Performance measures obtained by TECAT over each collection

12.10.3 Analysis of results

From results described above, we can conclude with three main argumentations:

1. The more classes we have, the more difficult is to find a suitable multi-label classifier. Mainly due to imbalance causes, these data sets are a real challenge for researchers.

2. Depending on the number of documents, we have to adjust filters like the one based on the information gain. As we can see, the performances found on the *hep-th* partition increases according to the number of features allowed by the filter. This obvious relation need further research in order to set a dynamic limit for such values depending on the original dimensionality found in the corpus.

3. Taking into account that most of our experiments have focused on studying best parameters for multi-labeling in the HEP domain, but using almost always the *hep-ex* partition, we find that almost same performances are registered by *astro-ph* partition (being this one, an smaller one, i.e. fewer data available), and also by the *hep-th* partition. The last one is much larger, with more classes, but its F1 value is only 3,7% lower than the one registered for *hep-ex* corpus.

In our opinion, these results provide strong evidence of the robustness of our method. Our approach for multi-labeling of High Energy Physics documents can very well be scaled to the whole of the CERN digital library.

12.11 EXP 12.11 - Real-time multi-labeling

The last set of experiments performed brings up the capabilities of the system proposed as a multi-label classifier able to process documents in plain-text format and produce groups of labels in what we can consider a real-time response.

12.11.1 Configuration

A total of 2779 abstracts have been used to measure the time it takes to TECAT, which implements all our studied approaches, to generate a set of related key words to every document. Each abstract is in a separate file, in plain text (ASCII) format. We have programmed an script that will take, one by one, these files and pass them to TECAT. The TECAT classifier will load, at every classification, its configuration files (the trained data and class models). Then, it will process the document and pass it to the series of classifiers in order to produce final labels. The process is modeled in figure 12.31.

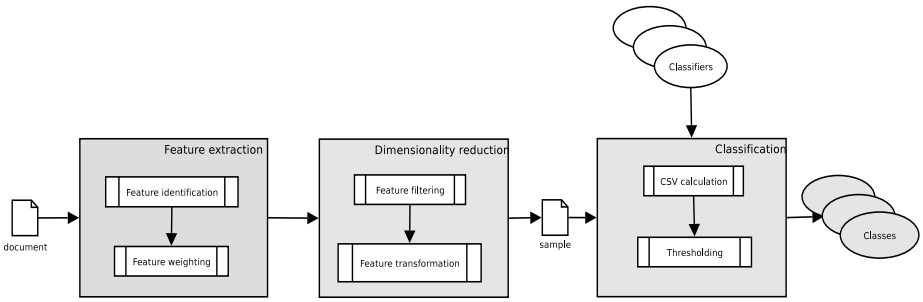


Figure 12.31: Classification steps within TECAT

As we can prove the feasibility of our approach as a real-time classifier, we have chosen a common server platform to run this benchmark. The machine used was a Dual (bi-processor) Intel Xeon at 2.8GHz per processor running SuSE Linux 9.1.

The TECAT configuration used to train the system is as given by table 12.35. The only algorithm used as base binary classifier was *PLAUM*, as it performs reasonably well and is a built-in algorithm in TECAT. The system is intended more as a framework for multi-labeling research than for a classification production resource, but it has been designed to report a good performance in speed, though. Anyhow, since it is trained usually into a set of "folds" for the n-fold cross validation testing, the classification of a single document is resolved by using the learned models from just one single fold, as every fold is ideally trained over the whole set of possible key words[11].

12.11.2 Results

When running the classifier against every abstract to produce their key words, the `time` UNIX command has been used to record the duration of the task into three measurements:

1. **Real**. It reports, in seconds, the elapsed real time between invocation and termination of the command (the invocation of the TECAT classifier).

2. **User**. The user CPU time (the CPU time consumed by the user owner of the process run).

3. **System**. The system CPU time (the CPU time consumed by the operating system to effectively run and complete the process, involving shell operations).

[11] We know this is not true due to the high class imbalance registered across the corpus, but in any case it does not affect the benchmark itself.

indexing		folding	
stop-words	*yes*	DF term filter	*1*
stemming	*yes*	no. folds	*10*
min. term freq.	*0*	IG filter	*50000*
min. term length	*0*	DF class filter	*1*
max. term length	*40*	normalization	*yes*
term weighting:	*TF.IDF*		
learning		**classification approach**	
S-cut	*no*	mode	*by class*
measure	*F1*		
threshold	*0.1*		
methods	*PLAUM*		

Table 12.35: Parametrization of TECAT for experiment 12.11

Timings for every document has been computed at classification. A summary of representative final statistics is detailed in table 12.36. First column, `no. classes`, reports the classes returned by the classifier. Next three columns provide the three measures described above. Last column shows the statistic computed for each of the four previous values over the whole distribution of values obtained for proposed corpus.

no. classes	real	user	system	statistic
5,29	1,13	1,09	0,04	**average**
17	2,13	2,09	0,07	**maximum**
0	1,06	1,01	0,03	**minimum**
4	1,1	1,07	0,04	**mode**
8,2	0,01	0,01	0	**variance**
2,86	0,11	0,11	0,01	**standard deviation**

Table 12.36: Global statistics for `hep-ex` corpus classification

12.11.3 Analysis

As we can see, the average classification time taken by TECAT is few more than a second, so our approach is, indeed, a real time classifier, as reported by an authored paper [84]. That is an important success of the architecture for multi-label classification studied along this work. Considering that we are dealing with a final set containing about 300 classes successfully trained, it is a valid solution for the CERN Document Server and its huge collection of full-text papers. It is also important to note that the number of labels proposed for each document is dynamically controlled by TECAT, returning a reasonable number

within a reasonable range (from nothing to 17 in these experiments).

12.12 EXP 12.12 - TECAT on EUROVOC data

So far, we have applied TECAT just on High Energy Physics corpora. All our experiments have been focused on relevant issues raised during the study of a multi-label text classifier when classifying against thousands of classes, as is in the case of the DESY thesaurus used for HEP papers. But now the question to be answered is whether our conclusions are applicable over other collections showing also high imbalance degree of classes across their multi-labeled documents.

This is the last experiment we have carried out before closing this work, since the access to EUROVOC data has been traditionally restricted. Anyhow, extensive research has been done on the database of documents labeled with EUROVOC descriptors, as described in section 9.1.3. Although the author has been in close contact with people at the Joint Research Centre, this is the first time some data is liberated to the research community.

12.12.1 The corpus

Jan Zizka (research visitor at the Joint Research Centre of the European Commission at Ispra, Italy), has prepared a set of documents already labeled using EUROVOC descriptors. In order to be able to release such data to the public, it has been *anonymized*, so features and classes are replaced by simple numbers. The data we have used in following runs of TECAT is composed by 21,216 documents (117,434 features), labeled with a total of 2,911 classes (from a set of 4,004 possible descriptors described for this collection). 208 megabytes of data have been fed into the system.

Figure 12.32: Inner imbalance degree for 200 most frequent classes

The imbalance truly high, as we can observe at figure 12.32, where imbalance

class	no. docs	IID
1	1259	0,881316
2	1075	0,898661
3	1050	0,901018
4	1017	0,904129
5	967	0,908842
6	893	0,915818
7	763	0,928073
8	705	0,933541
9	671	0,936746
10	665	0,937311

Table 12.37: Inner imbalance degree (IID) for ten most frequent classes in the EUROVOC corpus

degree for the 200 most frequent classes is shown graphically. In table 12.37 imbalance degree for ten most frequent classes is also provided. Note that the classes, since the corpus is anonymized, are named with numbers. None of the classes included appears in more than 12% of the released documents. When classifying using one-against-all approach, it implies that the learning algorithm will have very few positive samples, against a very large set of negative ones. Thus, our expectations for setting a robust margin decrease as the imbalance for the class the system is being trained grows.

12.12.2 Configuration

For these runs, we have configured TECAT as shown in table 12.38. This time we have allowed TECAT to select between two possible binary classifiers: PLAUM and Rocchio. The rest of parameters remains to the values we have concluded to be good values for the proposition of a performing multi-label classifier.

This configuration has been, as usual, lead to a 10-fold cross validation. The training process (distribution of documents across folds, stemming, stop words removal, feature reduction by document frequency and information gain value, weighting of features, training of base candidates, and evaluation and selection for every class of the best of the two algorithms) took about 28 hours. The testing only took 4 hours. At the end, TECAT was able to classify new incoming documents in less than 20 seconds. If only PLAUM algorithm had been used, it could have been down to 4 or 5 seconds, by using a feature of TECAT able to compact PLAUM models).

indexing		folding	
stop-words	*yes*	DF term filter	*1*
stemming	*yes*	no. folds	*10*
min. term freq.	*0*	IG filter	*50000*
min. term length	*0*	DF class filter	*1*
max. term length	*40*	normalization	*yes*
term weighting:	*TF.IDF*		
learning		**classification approach**	
S-cut	*only for Rocchio*	mode	*by class*
measure	*F1*		
threshold	*0.1*		
methods	*PLAUM and Rocchio together as candidates*		

Table 12.38: Parametrization of TECAT for experiment 12.12

12.12.3 Results

Results obtained are promising. We have tuned the system for optimal performance of TECAT, since as we have justified across the present work, that would imply an in depth study of the corpus and a costly analysis to set best parameters. Results for top ten classes are detailed in table 12.39. We can see that, despite the increasing inner imbalance degree of these top classes, the quality of results seems to not be according to that fact. In the whole collection, it is, but it is also true (as pointed by previous experimental results on the HEP collection), that each class may show different difficulties for its correct training. Maybe, the problem is to establish which is the source of data a certain key word needs for its proper modeling.

Global results have been obtained in a per-document basis (see chapter 7 for more details). They indicate that, after filtering too low frequent classes, 1424,10 key words where passed to the training phase, and that 1281,0 where successfully trained (which is a pretty high degree of capacity of learning). Such global values obtained were 0.511867 of precision, 0.509602 of recall, 0.997018 of accuracy, 0.002982 of error, 0.476656 for F1, and 0.510734 for the break-even-point.

12.12.4 Analysis

We have been in contact with researchers at the Joint Research Centre (JRC) to share our experiences with them. They have found our results very good. The way we have computed our measures is by only counting exact matched.

class	precision	recall	accuracy	error	F1	BEP
1	0.713993	0.702386	0.965497	0.034503	0.705030	0.708189
2	0.639921	0.434601	0.956285	0.043715	0.503703	0.537261
3	0.399299	0.708935	0.930454	0.069546	0.506372	0.554117
4	0.735356	0.686286	0.972810	0.027190	0.706778	0.710821
5	0.475359	0.732636	0.949727	0.050273	0.566215	0.603998
6	0.904384	0.851261	0.989913	0.010087	0.876128	0.877823
7	0.381226	0.212821	0.957133	0.042868	0.258883	0.297024
8	0.653248	0.576640	0.975791	0.024209	0.611103	0.614944
9	0.528606	0.643517	0.970162	0.029838	0.576800	0.586062
10	0.856026	0.741460	0.987846	0.012155	0.792116	0.798743

Table 12.39: Classification results for ten most frequent key words using EUROVOC collection

The system running at the JRC is a ranking based one, so different measures of precision and recall are computed depending on the number of terms considered from the final output of the system. In our experiments (though TECAT can operate in a ranking manner), the list of key words returned is a closed list, the user does not need to specify how many terms should be taken. When working on fully automatized environment, the latter is preferred, whether for machined aided indexing we may want to be provided with a list of candidates ranked by the confidence found by the system (though TECAT can also work in such manner). Anyhow, this final returning strategy (ranked or not) will depend on the final use the system will have. For example, the JRC team found that for the NewsExplorer application, the ranked strategy was preferable, as it turned to produce better results (see [98]).

By exact matching the highest value obtained by their system is 0.63 (63%) for F1 measure, with a precision of 0.58 and a recall of 0.68 (when taking the 10 top ranked descriptors, see [18]). Though our values are lower than these ones, they report that their system has been optimized through years in order to get such results, so ours are more than worth considering.

As conclusion, we can fully validate our multi-label classification framework over the EUROVOC corpus. Therefore, we expect that our conclusions found for the HEP collection are extensible to other multi-labeled collections of documents showing a high class imbalance degree.

Chapter 13

Conclusions and future work

Once all our experiments have been exposed and their results analyzed, and due mainly to the large variety of them, it is time to summarize key points found during this research, underlining major issues covered by present work. It is, of course, agreed that many open issues remain in the horizon, and we will mention them in the last section of this chapter. Text Categorization research is a richness area where several computer based techniques meet to propose a wide variety of solutions. We believe that the success of a system resides not only on the excellence of proven learning algorithms, or specific natural language based methods, but also on a closer observation of the matter under study: the collection of documents. Depending on the characteristics a given corpus may present, our system will be, of course, differ from that focused on another document collection with its own particular properties, although some experiments have reported promising results (see experiment 12.12).

TECAT is not the expected final result of this thesis, but rather an intermediate product that has provided us with a consistent framework able to face every single subject of study during the definition and implementation of a real-time multi-label classifier. Previous implementations using *Perl*, *Java* and *Python* programming languages have been also fruitful as experimental playgrounds, but always with the same focus: automatize the classification of documents in the High Energy Physics field.

The three major points covered in this research are summarized in following sections as **main scientific results** of the present work:

1. the new corpus introduced to the research community,

2. the new approach for multi-label classification,

3. and the importance of additional sources of data for effective class assignment.

Finally, as not a scientific result, the TECAT tool is reported aside as a product of this work, and further topics that remain opened and that demand additional research effort are listed at the end of this chapter.

13.1 HEP collection: a new and challenging corpus for multi-labeled text categorization research

In chapter 11 a new collection of documents annotated in a multi-labeling fashion is introduced. This collection provides a challenging amount of data with many properties that make it very interesting for the research community in Text Categorization. So far, there is no such a rich collection of papers manually indexed with predefined topics. Its high class imbalance has made us to consider certain design alternatives seldom covered in previous works. This work has studied this, a priory, disadvantage to propose a solution that has been extensively analyzed in performed experiments. Thanks to this special distribution of classes across documents we have faced the imbalance and produced following conclusions:

1. We have defined a new indicator for classes imbalance: the *inner imbalance degree* (equation 12.6), which improves previous propositions by providing a more consistent factor allowing comparisons at this level between non-related collections.

2. We have detailed this imbalance found in this collection, and analyzed how we could deal with such characteristic, as result we have proposed the method described in next section, proven to be valid and consistent solution to this problem.

3. We have detailed the richness of data available from this collection. In section 13.2 further details about it are provided.

4. The variability of experiments performed has covered many of the aspects to be considered in the categorization of text coming from this collection and offers a base line for future experiments. Processing phases like *feature selection*, *dimensionality reduction* or *feature weighting* has been extensively examined, justifying the tuning of the set of parameters involved in the aim of defining a final classification method.

Access to the data is permitted and researchers from all around the world can test their methods and compare their results with ones encountered by us.

As pointed several times, the success of a classification system depends on the capability of the method to fit on the specific nature of the corpus treated. It has been our contribution to touch many of the variables involved in such "fit". Now, coming investigations may explore certain possibilities not covered in the present work and that may result in an improvement of the classification system.

When a new collection is offered to the research community, it is a double grace to serve it along with the experimental machinery this thesis may have offered. It aids in the understanding of the properties of the data and determines straightforward issues that are worth studying.

13.2 Adaptive selection of base classifiers

The method for dealing with a highly imbalanced multi-labeling framework is discussed in experiment 12.7. This method consists in the automatic generation, by supervised learning, of a multi-label classifier that proposes just one classifier for each class. Classes yielding to non-performing classifiers are discarded. But the system allows the possibility of training among a given list of candidate algorithms. These candidates may show wide differences among them in the interpretation of value returned, the approach followed, the mathematical complexity of the algorithm, or the parametrization passed. After the training and evaluation of such candidates, the one generating best results is selected as the classification scheme for the class. At the end, for every class, either a unique classification model, or the determination of never proposing that class, is stored in the global classifier model. It has been proven during our experiments that such approach generates a fast multi-label classifier, able to generate labels in real-time executions, able to integrate in an easy manner additional base classifiers, and able to be parametrized, through the α threshold, to satisfy specific requirements.

This method works very well with collections with a large set of possible classes and it is not difficult to be implemented, due to its conceptual simplicity. It also allows the integration of many possible resources, since the classification process is clearly divided into defined processing: feature identification, feature selection, dimensionality reduction, feature weighting, training of binary candidate algorithms, and evaluation of them per class.

13.3 Metadata records: an informationally rich source

Text Categorization research tends to focus on Machine Learning algorithms. The lack of corpora available generally involves a recursive study of methods applied to the same collections. It is important to state common sets of data,

since that is the way to establish a proper benchmark, but when such data show a lack on certain aspects demanding to be closely study, then additional collections should be provided. But sometimes we do not consider in depth all the information provided from a given source, and that may imply ignore some benefits that were already there. That is the case of the metadata information related to every document in the digital library of High Energy Physics documents at CERN. Former research on such documents has either focused on citations (see subsection 9.2.1 for more details), or on the full-text version of papers provided by CDS. Human indexers agglomerate a vast knowledge on the subjects treated on reviewed papers, and there are certain key words that cannot be assigned without additional information usually out of the scope of the document content. How can an indexer, by reading an scientific article, know that the text is the result of a talk, or a survey about a collaboration between research centers? The metadata attached to records stored in the database may provide simple answers to that apparently difficult questions.

Some approaches like the AIR/PHYS (see subsection 9.2), are based on the study of underlying structures inferred from simple document segmentation. The assignment of a key word is result of the combined computation of features weights appearing in the title, the abstracts, and the rest of the paper.

In our experiment 12.8 we have analyzed how certain combinations of available sources effectively yield to performance melioration. That should motivate the debate of first stating, when constructing a new classifier for a certain domain, which is the nature of the data acquired, and find out if further sources of information could be fed into the system. It is a basic premise: we classify by analyzing certain information, the richer such information is, the more accurate our decision should be in return.

13.4 Practical product result: TECAT software

The CERN Document Server has determined the crucial role a system like the one unveiled during the reading of these pages. This work has, therefore, covered a dilated spectrum of methods and algorithms, and it could have been even much bigger. It is so immense the number of possibilities, that we have encountered many difficult decisions to be taken, like not to explore certain aspects suitable of been included in the present research. But, in our opinion, this work has successfully integrated many relevant topics and discovered relevant aspects as confirmed by our discussions at every work shop and lecture given using results found in our experiments. Proudly, we have a classification system which is a production system, a real application now being integrated into the CDS software. The sample document shown in figure 13.1 has been manually labeled with key words listed in figure 13.1. TECAT automatically proposed the key word list of figure 13.3. As we can see, just three key words are selected by the classifier, with 100% of precision and 50% of recall, though.

```
<?xml version="1.0" encoding="UTF-8"?>
<collection>

 <dc xmlns="http://purl.org/dc/elements/1.1/"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://purl.org/dc/elements/1.1/
 http://www.openarchives.org/OAI/1.1/dc.xsd">

  <language>eng</language>

  <creator>Albrecht, H</creator>

  <title>
   Determination of the Michel Parameters $\rho$, $\xi$,
   and $\delta$ in $\tau$-Lepton Decays with $\tau \to
   \rho \nu$ Tags
  </title>

  <subject>
   Particle Physics - Experimental Results
  </subject>

  <identifier>
   http://preprints.cern.ch/ ... id=9711022
  </identifier>

  <description>Using the ARGUS detector at the $e^+
  e^-$ storage ring DORIS II, we have measured the
  Michel parameters $\rho$, $\xi$, and $\xi\delta$
  for center of mass energies in the region of the
  $\Upsilon$ resonances. Using 0.04 \pm 0.08$,
  $\xi_{e}= 1.12 \pm 0.20 \pm 0.09$, $\xi\delta_{e}
  = 0.57 \pm 0.14 \pm 0.07$, $\rho_{\mu}= 0.69 \pm
  0.06 \pm 0.08$, $\xi_{\mu}= 1.25 \pm 0.27 the
  combined ARGUS results on $\rho$, $\xi$, and
  $\xi\delta$ using this work on previous
  measurements.</description>

  <date>1997-12-01</date>

 </dc>

</collection>
```

Figure 13.1: Sample for metadata information in XML DC format.

```
DESY HERA Stor
electron positron
experimental results
magnetic detector
Michel parameter
tau
```

Figure 13.2: Main DESY key words manually assigned to sample at figure 13.1

```
1.233175 tau
1.533310 electron positron
0.327726 magnetic detector
```

Figure 13.3: Main DESY key words automatically assigned by TECAT to sample at figure 12.25. They are preceded by their associated *Classification Status Value* (in this sample, returned by the PLAUM algorithm).

13.5 Open issues and future research

As commented before, many aspects of the system conform a ground for further analysis. The amount of parameters and methods involved in a complete solution would require much longer time to be covered in detail. We list now some of the main open issues that may drive further research from present work.

- Extending method capabilities to other domains is a research task that has been briefly unveiled by experiments presented in section 12.12. Results obtained suggest the robustness and flexibility of the method introduced.
- It is important to recall that we have covered only the first level of the DESY thesaurus: no secondary key words have been proposed. In our opinion, it is needed a vast amount of information to consider such study, since just facing this first level, the imbalance registered let few chances for success. Imbalance at second level is, consequently, higher and, therefore, currently untreatable. Though, future approach may bring light to this difficult problem.
- Physics documents are plentiful of formulae, diagrams and tables. We have not paid special attention to these important items. The number of tables or graphs found in document content could represent an relevant aid when deciding which classes have to be assigned. The nature of certain formulae, and the values found in them could be used to clarify obviated classes in this work like energy or reaction related ones. Thus, and wider study on document content along with its segmentation for a more accurate weighting of features is imposed in further developments.
- As corollary of previous issue, a deeper analysis of Natural Language Techniques may lead to improvements in system performance. Our light study of some of these techniques (like multi-word detection) remains untouched many interesting approaches as document summarization (for condensing document content and reducing its initial length to speed up later processing), word sense disambiguation (despite HEP language technicality and consequent possible lack of such problem), semantic indexing (to reduce the feature space), and so on. In any case, these possible task of research have been discarded by obvious and still valid reasons: the weight of response-time constraints was high.
- New Machine Learning algorithms appear continuously. Some of the current proposals may be promising within our domain. Test boosting algorithms like AdaBoost and other methods like K-NN, Naïve Bayes, EG, LVQ and Logistic Regression on the HEP corpus could lead to significant improvements, although one of our main conclusions is the suggestion of looking for additional sources of data, such as meta-data in document records or other information outside the content scope. rather than trying yet another base classification algorithm.

As we can assert from results found in experiment 12.12, TECAT is a flexible system and general enough to be considered as an useful resource for document classification research. Anyhow, we maintain the hope on new software to come improving our results for HEP documents.

Benefits coming from the application of automated classifiers have been reported in chapter 8. From that we can do nothing but expect a promising future for that kind of systems. This work has proven the viability of automatic multi-labeling for collections with a large associated thesaurus. It has covered many problems found during the construction of a productive solution and proposed a coherent architecture through an empirical analysis. The execution of thousands of experiments during the last three years has given in return a complete analysis to the multi-labeling problem, but has also identified additional topics as product of the use of a novel corpus for the research community.

Bibliography

[1] ASIS Thesaurus of Information Science. http://www.asis.org/Publications/Thesaurus/isframe.htm.

[2] EUROVOC Thesaurus. http://europa.eu.int/celex/eurovoc/.

[3] The freedesktop project. URL: http://freedesktop.org.

[4] The gnome website. URL: http://www.gnome.org.

[5] The grace engine. URL: http://www.grace-ist.org.

[6] INSPEC Thesaurus. http://www.iee.org.uk/publish/inspec/.

[7] Erin L. Allwein, Robert E. Schapire, and Yoram Singer. Reducing multiclass to binary: A unifying approach for margin classifiers. In *Proc. 17th International Conf. on Machine Learning*, pages 9–16. Morgan Kaufmann, San Francisco, CA, 2000. URL `citeseer.ist.psu.edu/allwein00reducing.html`.

[8] Ethem Alpaydin. Combined 5 x 2 cv f test for comparing supervised classification learning algorithms. *Neural Computation*, 11(8):1885–1892, 1999.

[9] A. Arampatzis, T. van der Weide, C. Koster, and P. van Bommel. An evaluation of linguistically-motivated indexing schemes, 2000. URL `citeseer.ist.psu.edu/article/arampatzis00evaluation.html`.

[10] L. Douglas Baker and Andrew Kachites McCallum. Distributional clustering of words for text classification. In *Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 96–103. ACM Press, 1998. ISBN 1-58113-015-5.

[11] Brain T. Bartell, Garrison W. Cottrell, and Richard K. Belew. Latent Semantic Indexing is an Optimal Special Case of Multidimensional Scaling, 1990.

[12] Eric Bauer and Ron Kohavi. An empirical comparison of voting classification algorithms: Bagging, boosting, and variants. *Machine Learning*, 36(1-2):105–13, August 1999.

[13] P. Baxendale. Machine-made index for technical literature — an experiment. *IBM Journal, pages 354–361, October*, 1958.

[14] Ron Bekkerman, Ran El-Yaniv, Naftali Tishby, and Yoad Winter. Distributional word clusters vs. words for text categorization. *J. Mach. Learn. Res.*, 3:1183–1208, 2003. ISSN 1533-7928.

[15] Tim Berners-Lee, James Hendler, and Ora Lassila. The Semantic Web. *Scientific American*, 284(5):34–43, May 2001. ISSN 0036-8733. URL `http://www.sciam.com/2001/0501issue/0501berners-lee.html`.

[16] P. Biebricher, N. Fuhr, G. Lustig, M. Schwantner, and G. Knorz. The automatic indexing system air/phys - from research to applications. In *Proceedings of the 11th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 333–342. ACM Press, 1988. ISBN 2-7061-0309-4.

[17] Leo Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. *Classification and Regression Trees.* Wadsworth, 1984. ISBN 0-534-98053-8.

[18] Bruno Pouliquen, Ralf Steinberger, and Camelia Ignat. Automatic Annotation of Multilingual Text Collections with a Conceptual Thesaurus. In Amalia Todirascu, editor, *Proceedings of the workshop 'Ontologies and Information Extraction' at the EuroLan Summer School 'The Semantic Web and Language Technology'(EUROLAN'2003)*, page 8 pages, Bucharest (Romania), 2003.

[19] M. Buenaga, J.M. Gómez, and B. Díaz. Using WORDNET to complement training information in text categorization. In *Proceedings of Second International Conference on Recent Advances in Natural Language Processing (RANLP)*, 1997.

[20] Christopher J. C. Burges. A Tutorial on Support Vector Machines for Pattern Recognition. *Data Mining and Knowledge Discovery*, 2(2):121–167, 1998. URL `citeseer.nj.nec.com/burges98tutorial.html`.

[21] Maria Fernanda Caropreso, Stan Matwin, and Fabrizio Sebastiani. A learner-independent evaluation of the usefulness of statistical phrases for automated text categorization. pages 78–102, 2001.

[22] Nitesh V. Chawla. C4.5 and imbalanced data sets: Investigating the effect of sampling method, probabilistic estimate and decision tree structure. In *Workshop on Learning from Imbalanced Datasets II, ICML*, Washington DC, 2003.

[23] E. Chisholm and T. G. Kolda. New term weighting formulas for the vector space method in information retrieval. Technical report, Oak Ridge National Laboratory, 1998.

[24] Erica Chisholm and Tamara G. Kolda. New term weighting formulas for the vector space method in information retrieval. URL `citeseer.nj.nec.com/198082.html`.

[25] K. W. Church and P. Hanks. Word association norms, mutual information and lexicography. *Computational Linguistics*, 16(1):22–29, 1990.

[26] Jean-Blaise Claivaz, Jean-Yves Le Meur, and Nicholas Robinson. From fulltext documents to structured citations: Cern's automated solution. *High Energy Physics Library Webzine*, (5), November 2001.

[27] William W. Cohen. Fast effective rule induction. In Armand Prieditis and Stuart Russell, editors, *Proc. of the 12th International Conference on Machine Learning*, pages 115–123, Tahoe City, CA, July 9–12, 1995. Morgan Kaufmann. ISBN 1-55860-377-8. URL `citeseer.ist.psu.edu/cohen95fast.html`.

[28] Christopher Culy. An extension of phrase structure rules and its application to natural language. Master's thesis, Stanford University, 1983.

[29] M.A. García Cumbreras, L.A. Ureña López, A. Montejo Ráez, and F. Martínez Santiago. Búsqueda de respuestas multilingüe: Clasificación de preguntas en español basada en aprendizaje. *Sociedad Española para el Procesamiento del Lenguaje Natural*, 34:31–40, 2005.

[30] David Dallman and Jean-Yves Le Meur. Automatic keywording of high energy physics. Technical report, European Laboratory for Particle Physics, Geneva, Switzerland, October 1999.

[31] Scott C. Deerwester, Susan T. Dumais, Thomas K. Landauer, George W. Furnas, and Richard A. Harshman. Indexing by Latent Semantic Analysis. *Journal of the American Society of Information Science*, 41(6):391–407, 1990. URL `citeseer.nj.nec.com/deerwester90indexing.html`.

[32] DESY. The High Energy Physics Index Keywords, 1996. http://www-library.desy.de/schlagw2.html.

[33] Thomas G. Dietterich. Approximate statistical tests for comparing supervised classification algorithms. *Neural Computation*, 10(7):1895–1923, 1998.

[34] Sándor Dominich. Formal foundation of information retrieval. In *Proceedings of the ACM SIGIR MF/IR*, pages 8–15, 2000.

[35] Chris Drummond and Robert C. Holte. C4.5, class imbalance, and cost sensitivity: Why under-sampling beats over-sampling. In *Workshop on Learning from Imbalanced Datasets II, ICML*, Washington DC, 2003.

[36] V.V. Ezhela et al. Citations as a mean for discovery and automatic indexing of the scientific texts with new knowledge for a given subject, 2001.

[37] C. J. Fall, A. Törcsvári, K. Benzineb, and G. Karetka. Automated categorization in the international patent classification. *ACM SIGIR Forum*, 37(1):10–25, 2003. ISSN 0163-5840.

[38] Reginald Ferber. Automated indexing with thesaurus descriptors: a cooccurrence-based approach to multilingual retrieval. In Carol Peters and Costantino Thanos, editors, *Proceedings of ECDL-97, 1st European Conference on Research and Advanced Technology for Digital Libraries*, pages 233–251, Pisa, IT, 1997. Lecture Notes in Computer Science, number 1324, Springer Verlag, Heidelberg, DE.

[39] Tony Hey Fran Berman, Geoffrey Fox, editor. *Grid Computing: Making the Global Infrastructure a Reality*. Wiley, 2003.

[40] Yoav Freund. An adaptive version of the boost by majority algorithm. *Machine Learning*, 3(43):293–318, June 2001.

[41] Nir Friedman, Dan Geiger, and Moises Goldszmidt. Bayesian network classifiers. *Mach. Learn.*, 29(2-3):131–163, 1997. ISSN 0885-6125.

[42] N. Fuhr and G. E. Knorz. Retrieval test evaluation of a rule based automatic indexing (AIR/PHYS). In C. J. van Rijsbergen, editor, *Proceedings of the Third Joint BCS and ACM Symposium*. Cambridge University Press, 1984.

[43] 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.

[44] George A. Miller, Richard Beckwith, Christiane Fellbaum, Derek Gross, and Katherine Miller. Introduction to WordNet: An On-line Lexical Database, 1993.

[45] Fredric C. Gey. Inferring probability of relevance using the method of logistic regression. In *SIGIR '94: Proceedings of the 17th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 222–231, New York, NY, USA, 1994. Springer-Verlag New York, Inc. ISBN 0-387-19889-X.

[46] Warren R. Greiff and Jay M. Ponte. The maximum entropy approach and probabilistic ir models. *ACM Trans. Inf. Syst.*, 18(3):246–287, 2000. ISSN 1046-8188.

[47] Jiawei Han and Xiaoxin Yin. CPAR: Classification based on predictive association rules. 2003.

[48] William Hersh, Chris Buckley, T. J. Leone, and David Hickam. Ohsumed: an interactive retrieval evaluation and new large test collection for research. In *Proceedings of the 17th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 192–201. Springer-Verlag New York, Inc., 1994. ISBN 0-387-19889-X.

[49] David A. Hull. Stemming algorithms: A case study for detailed evaluation. *Journal of the American Society of Information Science*, 47(1): 70–84, 1996. URL `citeseer.ist.psu.edu/hull96stemming.html`.

[50] N. Japkowicz and S. Stephen. The class imbalance problem: A systematic study. *Intelligent Data Analysis Journal*, 6(5), November 2002.

[51] Nathalie Japkowicz. Class imbalances: Are we focusing on the right issue? In *Workshop on Learning from Imbalanced Datasets II, ICML*, Washington DC, 2003.

[52] 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*, number 1398, pages 137–142, Chemnitz, DE, 1998. Springer Verlag, Heidelberg, DE. URL `citeseer.ist.psu.edu/joachims97text.html`.

[53] Paul H. Klingbie June P. Silvester, Michael T. Genuardi. Machine-Aided Indexing at NASA, 1994.

[54] Adam Kilgariff. Which words are particularly characteristic of a text? a survey of statistical approaches. In *Proceedings of Language Engineering for Document Analysis and Recognition, AISB Workshop*. Falmer, Sussex, 1996.

[55] Adam Kilgarriff and David Tugwell. WORD SKETCH: Extraction and display of significant collocations for lexicography. In *Proc. Collocations Workshop, ACL 2001*, pages 32–38, 2001.

[56] Jyrki Kivinen and Manfred Warmuth. Exponentiated gradient versus gradient descent for linear predictors. Technical Report UCSC-CRL-94-16, University of California, Santa Cruz, Jack Baskin School of Engineering, June 1994.

[57] Gerhard Knorz, Gerhard Lustig, Tzeras Tzeras, Michael Schwantner, Norbert Fuhr, and Stephan Hartmann. Automatic indexing in operation: The rule-based system AIR/X for large subject fields. Technical report, June 24 1993.

[58] Ron Kohavi. A study of cross-validation and bootstrap for accuracy estimation and model selection. In *Proc. of the Fourteenth International Joint Conference on Artificial Intelligence*, pages

1137–1145. Morgan Kaufmann, San Mateo, CA, 1995. URL citeseer.ist.psu.edu/kohavi95study.html.

[59] Aleksander Kolcz, Vidya Prabakarmurthi, and Jugal Kalita. Summarization as feature selection for text categorization. In *Proceedings of the Tenth International Conference on Information and Knowledge Management (CIKM-01)*, 2001. URL citeseer.nj.nec.com/kolcz01summarization.html.

[60] Daphne Koller and Mehran Sahami. Toward optimal feature selection. In *International Conference on Machine Learning*, pages 284–292, 1996. URL citeseer.ist.psu.edu/koller96toward.html.

[61] Boris Lauser and Andreas Hotho. Automatic multi-label subject indexing in a multilingual environment. In *Proc. of the 7th European Conference in Research and Advanced Technology for Digital Libraries, ECDL 2003*, volume 2769, pages 140–151. Springer.

[62] Alan R. Aronson Lawrence W. Wright, Holly K. Grossetta Nardini and Thomas C. Rindflesch. Hierarchical concept indexing of full-text documents in the unified medical language system information sources map. *Journal of the American Society for Information Science*, 50(6):514–523, 1999.

[63] D. D. Lewis. Evaluating Text Categorization. In *Proceedings of Speech and Natural Language Workshop*, pages 312–318. Morgan Kaufmann, 1991. URL citeseer.nj.nec.com/lewis91evaluating.html.

[64] D. D. Lewis. An evaluation of phrasal and clustered representations on a text categorization task. In *Proceedings of the 15th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 37–50. ACM Press, 1992. ISBN 0-89791-523-2.

[65] D. D. Lewis. Feature Selection and Feature Extraction for Text Categorization. In *Proceedings of Speech and Natural Language Workshop*, pages 212–217, San Mateo, California, 1992. Morgan Kaufmann. URL citeseer.ist.psu.edu/lewis92feature.html.

[66] D. D. Lewis. *Representation and Learning in Information Retrieval*. PhD thesis, Department of Computer and Information Science, University of Massachusetts, 1992.

[67] D. D. Lewis, Robert E. Schapire, James P. Callan, and Ron Papka. Training algorithms for linear text classifiers. 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 298–306, Zürich, CH, 1996. ACM Press, New York, US. URL citeseer.ist.psu.edu/lewis96training.html.

[68] Cong Li, Ji-Rong Wen, and Hang Li. Text classification using stochastic keyword generation. In *Proceedings of ICML'03*, pages 464–471, 2003.

[69] Y.H. Li and A.K. Jain. Classification of text documents. *Computer Journal*, 41(8):537–546, 1998.

[70] Susan M. Lloyd, editor. *Roget's Thesaurus*. Longman, 1982.

[71] K. E. Lochbaum and L. Streeter. Comparing and combining the effectiveness of latent semantic indexing and the ordinary vector space model for information retrieval. *Information Processing and Management*, 25(6): 665–676, 1989.

[72] H. Luhn. A Statistical Approach to Mechanized Encoding and Searching of Literary Information. *IBM Journal of Research and Development 1 (4) 309-317*, 1957.

[73] Inderjeet Mani and Mark T. Maybury, editors. *Advances in Automatic Text Summarization*. MIT Press, July 1999.

[74] C.D. Manning and H. Schtze. *Foundations of Statistical Natural Language Processing*. The MIT Press. Cambridge, Massachusetts, 1999.

[75] Daniel Marcu. Discourse trees are good indicators of importance in text. Technical report, Information Science Institute, University of Southern California, 1997.

[76] M. Teresa Martín-Valdivia, Miguel A. García Cumbreras, Manuel C. Díaz-Galiano, L. Alfonso Ureña López, and Arturo Montejo Ráez. Sinai at image clef 2005: Adhoc and medical tasks. *Lecture Notes on Computer Science (LNCS Series)*, 2005.

[77] M.T. Martín-Valdivia, M. García-Vega, and L.A. Ureña-López. LVQ for text categorization using multilingual linguistic resource. *Neurocomputing*, 55:665'–679, 2003.

[78] A. McCallum and K. Nigam. A comparison of event models for naive Bayes text classification. In *AAAI/ICML-98 Workshop on Learning for Text Categorization*, pages 41–48. AAAI Press, 1998.

[79] Andrew K. McCallum, Ronald Rosenfeld, Tom M. Mitchell, and Andrew Y. Ng. Improving text classification by shrinkage in a hierarchy of classes. In Jude W. Shavlik, editor, *Proceedings of ICML-98, 15th International Conference on Machine Learning*, pages 359–367, Madison, US, 1998. Morgan Kaufmann Publishers, San Francisco, US. URL `citeseer.ist.psu.edu/mccallum98improving.html`.

[80] R. Meir and G. Rtsch. *An introduction to boosting and leveraging*, pages 119–184. Advanced Lectures on Machine Learning, LNCS. Springer Verlag, 2003.

[81] Tom M. Mitchell. *Machine Learning*, chapter Decision tree learning, pages 55–59. McGraw-Hill, New York, 1997.

[82] A. Montejo-Ráez. Proyecto de indexado automático en el campo de la física de altas energías. *Sociedad Española para el Procesamiento del Lenguaje Natural*, 27(295-296), 2001.

[83] A. Montejo-Ráez. Towards conceptual indexing using automatic assignment of descriptors. Workshop in Personalization Techniques in Electronic Publishing on the Web: Trends and Perspectives, Málaga, Spain, May 2002.

[84] A. Montejo-Ráez. Asignación automática de palabras clave en tiempo real. In *II Jornadas de Tratamiento y Recuperación de la Información (JOTRI 2003)*, pages 289–291, Leganés, Madrid (Spain), September 2003. Universidad Carlos III de Madrid.

[85] A. Montejo-Ráez. Formal models for information retrieval. a review and a proposal for keyword assignment. In *Mathematical/Formal Methods in IR, Workshop in SIGIR 2003*, 2003.

[86] A. Montejo-Ráez and D. Dallman. Experiences in automatic keywording of particle physics literature. *High Energy Physics Libraries Webzine*, (issue 5), November 2001. URL: http://library.cern.ch/HEPLW/5/papers/3/.

[87] A Montejo-Ráez and R. Steinberger. Why keywording matters. *High Energy Physics Libraries Webzine*, (Issue 10), December 2004. URL `http://library.cern.ch/HEPLW/10/papers/2/`.

[88] A. Montejo-Ráez, R. Steinberger, and L. A. Ureña-López. Adaptive selection of base classifiers in one-against-all learning for large multi-labeled collections. (3230):1–12, 2004.

[89] Katharina Morik, Peter Brockhausen, and Thorsten Joachims. Combining statistical learning with a knowledge-based approach - a case study in intensive care monitoring. In *Proc. 16th International Conf. on Machine Learning*, pages 268–277. Morgan Kaufmann, San Francisco, CA, 1999. URL `citeseer.ist.psu.edu/morik99combining.html`.

[90] *Medical Subject Headings (MeSH).* National Library of Medicine, Bethesda, US, 1993.

[91] J. R. Pierce. *Introduction to Information Theory: Symbols, Signals, and Noise*, pages 86–87 and 238–239. New York: Dover, 2nd edition, 1980.

[92] Olga Pombo. *Leibniz and the Problem of a Universal Language.* Nodus Publikationen, 1987.

[93] M. F. Porter. *An algorithm for suffix stripping*, pages 313–316. Morgan Kaufmann Publishers Inc., 1997. ISBN 1-55860-454-5.

[94] Bruno Pouliquen. *Indexation de textes mdicaux par extraction de concepts.* PhD thesis, Facult de Mdecine, Rennes, France, 2002.

[95] J. Ross Quinlan. *C4.5: Programs for Machine Learning.* Morgan Kaufmann, 1993. ISBN 1-55860-238-0.

[96] J. Ross Quinlan and R. Mike Cameron-Jones. FOIL: A midterm report. In *Machine Learning: ECML-93, European Conference on Machine Learning, Proceedings*, volume 667, pages 3–20. Springer-Verlag, 1993. URL `citeseer.ist.psu.edu/quinlan93foil.html`.

[97] A. Montejo Ráez, L.A. Ureña López, and R. Steinberger. Text categorization using bibliographic records: beyond document content. *Sociedad Española para el Procesamiento del Lenguaje Natural*, 35:119–126, 2005.

[98] Steinberger Ralf, Bruno Pouliquen, and Camelia Ignat. Navigating multilingual news collections using automatically extracted information. In *Proceedings of the 27th International Conference 'Information Technology Interfaces' (ITI'2005)*, Cavtat / Dubrovnik, Croatia, June 20-23 2005.

[99] Ralf Steinberger, Bruno Pouliquen, and Johan Hagman. Cross-lingual Document Similarity Calculation Using the Multilingual Thesaurus EUROVOC. *Third International Conference on Intelligent Text Processing and Computational Linguistics*, 2002.

[100] Ralf Steinberger, Johan Hagman, and Stefan Scheer. Using thesauri for automatic indexing and for the visualisation of multilingual document collections. pages 130–141, 2000.

[101] Bhavani Raskutti and Adam Kowalczyk. Extreme re-balancing for svms: a case study. In *Workshop on Learning from Imbalanced Datasets II, ICML*, Washington DC, 2003.

[102] G. Rätsch, B. Schölkopf, S. Mika, , and K.-R. Müller. Svm and boosting: one class. Technical Report 119, GMD FIRST, Berlin, November 2000.

[103] A. M. Robertson and P. Willett. Evaluation of Techniques for the Conflation of Modern and Seventeenth Century English Spelling, April 13–14 1993.

[104] Stephen E. Robertson, Steve Walker, Micheline Hancock-Beaulieu, Aarron Gull, and Marianna Lau. Okapi at TREC. In *Text REtrieval Conference*, pages 21–30, 1992. URL `citeseer.ist.psu.edu/article/robertson96okapi.html`.

[105] Robertson S., Walker S., and Zaragoza H. Microsoft cambridge at trec-10: filtering and web tracks. In *Text Retrieval Conference (TREC-10)*, 2001.

[106] G. Salton and C. Buckley. Term weigthing approaches in automatic text retrieval. *Information Processing and Managemnet*, 24(5):513–523, 1988.

[107] Gerard Salton. Automatic Text Analysis. Technical Report TR69-36, Cornell University, Computer Science Department, 1969.

[108] Gerard Salton, A. Wong, and C. S. Yang. A Vector Space Model for Automatic Indexing. Technical Report TR74-218, Cornell University, Computer Science Department, July 1974.

[109] Mark Sanderson. Word Sense Disambiguation and Information Retrieval, 1997.

[110] R. Schapire and Y. Singer. BoosTexter: A system for multi-class multi-label text categorization, 1998. URL `citeseer.nj.nec.com/schapire98boostexter.html`.

[111] Robert E. Schapire and Yoram Singer. BoosTexter: A boosting-based system for text categorization. *Machine Learning*, 39(2/3):135–168, 2000. URL `citeseer.ist.psu.edu/schapire00boostexter.html`.

[112] Fabrizio Sebastiani. Machine learning in automated text categorization. *ACM Comput. Surv.*, 34(1):1–47, 2002. ISSN 0360-0300.

[113] R. Servedio. Smooth boosting and learning with malicious noise. *Journal of Machine Learning Research*, pages 633–648, September 2003.

[114] Claude E. Shannon. A mathematical theory of communication. 27(3): 379–423, July 1948. Continued 27(4):623-656, October 1948.

[115] Páraic Sheridan, Martin Braschler, and Peter Schäuble. Cross-language information retrieval in a multilingual legal domain. In Carol Peters and Costantino Thanos, editors, *Proceedings of ECDL-97, 1st European Conference on Research and Advanced Technology for Digital Libraries*, pages 253–268, Pisa, Italy, 1997. URL `citeseer.nj.nec.com/sheridan97crosslanguage.html`.

[116] Ralf Steinberger. Cross-lingual Keyword Assignment. In L. Alfonso Ureña López, editor, *Proceedings of the XVII Conference of the Spanish Society for Natural Language Processing (SEPLN'2001)*, pages 273–280, Jaén (Spain), September 2001.

[117] L. A. Ureña-López, M. Buenaga, and J. M. Gómez. Integrating linguistic resources in tc through wsd. *Computers and the Humanities*, 35(2):215–230, May 2001.

[118] C. J. van Rijsbergen. *Information Retrieval*. London: Butterworths, 1975. http://www.dcs.gla.ac.uk/Keith/Preface.html.

[119] V. N. Vapnik. *The Nature of Statistical Learning Theory*. Springer, New York, 1995.

[120] Lyubov A. Vassilevskaya. An approach to automatic indexing of scientific publications in high energy physics for database spires-hep. Master's thesis, Fachhochsule Potsdam, Institut fr Information und Dokumentation, September 2002.

[121] Natasha Vieduts-Stokolo. Concept recognition in an automatic text-processing system for the life sciences, 1987.

[122] B. C. Vilckery. *Information Systems.* Butterworth & Co., 1973.

[123] P. Vossen. EuroWordNet: a multilingual database for information retrieval, 1997. URL `citeseer.nj.nec.com/vossen97eurowordnet.html`.

[124] S. M. Weiss, C. Apt'e, D. Damerau, D. E. Johnson, F. J. Oles, T. Goetz, and T. Hampp. Maximizing text-mining performance. *IEEE Intelligent Systems*, 14(4):63–69, 1999.

[125] Ian H. Witten and Eibe Frank. *Data Mining.* Morgan Kaufmann Publishers, 2000.

[126] Gang Wu and Edward Y. Chang. Class-boundary alignment for imbalanced dataset learning. In *Workshop on Learning from Imbalanced Datasets II, ICML*, Washington DC, 2003.

[127] Li Y., Zaragoza H., Herbrich R., Shawe-Taylor J., and Kandola J. The perceptron algorithm with uneven margins. In *Proceedings of the International Conference of Machine Learning (ICML'2002)*, 2002.

[128] Yiming Yang. A study on thresholding strategies for text categorization. In W. Bruce Croft, David J. Harper, Donald H. Kraft, and Justin Zobel, editors, *Proceedings of SIGIR-01, 24th ACM International Conference on Research and Development in Information Retrieval*, pages 137–145, New Orleans, US, 2001. ACM Press, New York, US. URL `citeseer.ist.psu.edu/yang01study.html`. Describes RCut, Scut, etc.

[129] 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. URL `citeseer.nj.nec.com/yang99reexamination.html`.

[130] Yiming Yang and Jan O. Pedersen. A comparative study on feature selection in text categorization. In *Proceedings of the Fourteenth International Conference on Machine Learning*, pages 412–420. Morgan Kaufmann Publishers Inc., 1997. ISBN 1-55860-486-3.

[131] Tong Zhang and Frank J. Oles. Text categorization based on regularized linear classification methods. *Inf. Retr.*, 4(1):5–31, 2001. ISSN 1386-4564.

Appendix A

The DESY thesaurus

The terms in this key word list are used by the DESY Documentation Service for the indexing of papers on high energy (beam momentum above 400 MeV (per nucleon)) and particle physics, accelerator and detector technology and quantum field theory.

Purpose of Keywords Assignment

These key words serve the following purposes: they allow the generation of a subject index for the biweekly periodical HIGH ENERGY PHYSICS INDEX (HEPI), they are important for computerized information retrieval and SDI (Selective Dissemination of Information) service at DESY and other high-energy physics centers. The total key words assigned to a paper may also be useful as a sort of abstract. A search by key words is also valuable in SPIRES.

Form of Keyword Assignment

Keywords may be used singly or coupled by comma and blank (for example: 'field theory' (single) and 'field theory, nonabelian' (coupled)). While the first term is generally a regular key word, the second term may be a key word or a non-key word.

Non-key words which are frequently used are standardized; they are contained in the alphabetical list (see also point 10).

Depth of Indexing

Papers on peripheral topics will usually have fewer key words per paper than papers on high energy physics. Examples of peripheral topics are quantum mechanics, statistical mechanics, gravitation, astrophysics, and nuclear physics with beam energy above 400 MeV/nucleon.

Classification

Beside of indexing the selected papers are classified with 16 topical fields, one main and any number of secondary fields. For example: Experimental papers on electroproduction of charmed particles are assigned to the main field ((E)) and the secondary field ((C)); books on field theory are assigned to the main field ((3)) and the secondary field ((Z)).

The 16 topical fields are the following:

```
I.  Experimental Physics

 ((A))    general (also cosmic radiation, nuclear physics,
          and gravitational radiation)
 ((B))    weak interactions
 ((C))    electromagnetic interactions, photoproduction
 ((D))    strong interactions
 ((E))    charm, beauty, truth

II.  Technology and Techniques in High Energy Physics

 ((F))    accelerators
 ((G))    detecting systems, experimental methods and data analysis
          methods

III.  Theoretical Physics

 ((T))    general (also relativistic quantum mechanics, mathematics,
                   statistical mechanics, cosmic radiation,
                   nuclear physics, and gravitational radiation)
 ((U))    weak interactions
 ((V))    electromagnetic interactions, photoproduction
 ((W))    strong interactions
 ((X))    charm, beauty, truth
 ((Y))    symmetry principles (also current algebra)
 ((Z))    quantum field theory

IV.  Monographs and Conference Proceedings
```

```
((3))    books
((4))    conferences
```

Two-Particle Initial State

Most of the combinations of any two particles (but not all) in the list are single regular key words. They are to be used for the description of the initial state of interactions. The particles are arranged in order of rising masses, in case of same masses in order of charges: positive particle before negative particle (except 'electron positron' and 'anti-p p').

Particle Spectra and Other Particle Combinations

Particles or particle combinations in final or intermediate states in conjunction with the key words: `angular correlation, angular distribution, bound state, correlation, coupling, coupling constant, double-beta decay, energy spectrum, final state, interference, mass difference, mass ratio, mass spectrum, mixing angle, momentum spectrum, particle identification, universality, vertex function, yield` follow the key word and are listed in parentheses in the order of decreasing masses, in case of same masses in the order charge (+ -).

Examples:

```
angular distribution, (photon)
 final state, (n p 0lepton)
 bound state, (nucleon 2pi)
 mass spectrum, (pi+ pi- pi0)
```

but:

```
 K0 anti-K0, interference
 D0 anti-D0, interference
 B0 anti-B0, interference
```

Reaction Equations

Reactions of two particles or decay modes of a particle are given as in the following examples:

```
anti-p p  -->  K0 K- pi+
p p  -->  p anything
Delta(1232)0  -->  p pi-
photon deuteron  -->  2p pi- (n)pi0 anything0
```

Particles on the left-hand side are arranged in the order of beam and target, particles on the right-hand side are arranged in the order of decreasing masses, in case of equal masses in the order of charge (+ -).

Resonances

Meson and baryon resonances are generally named as in the Particle Data Group Tables; charge states are indicated only for the rho(770) and the Delta(1232). Heavy-quark particles are commonly not called resonances.

For new and yet unnamed resonances the mass (in MeV) may be given in parentheses, e.g. `mass enhancement, (1440)` or `postulated particle, (1440)`.

Energy Declarations

Energy resp. momentum is given in the same way as in the paper, but always in GeV, above 10**5 in exponential form. Additionally papers are assigned to energy-ranges.

		E(beam) [GeV]		(target: nucleon)
Range	E(cms) [GeV]	beam: e-, photon, pi	beam: K	beam: p
((1))	0.0 - 3.0	0.0 - 4.32	0.0 - 4.20	0.0 - 3.85
((2))	- 10.0	- 52.8	- 52.7	- 52.3
((3))	- 30.0	- 479.	- 479.	- 478.
((4))	- 100.0	....	- 5325.	....
((5))	- 300.0	....	- 47900.	....
((6))	- 1000.0	....	- 532500.	....
((7))	- 10000.0	....	- 53250000.	....
((8))	> 10000.0	....	> 53250000.	....

For asymmetric colliders the centre-of-mass energy is E(cms) = 2 sqrt(E(1)E(2)). No energy range is given in case of cosmic radiation (when no interactions are

discussed) and for nucleus nucleus interactions.

Additional information on momentum transfer, limited angular range, etc. may be included. The general rules are illustrated by the following examples :

```
1.5-2.7 GeV-cms, ((1))
1.75, 3.00, 4.50 GeV/c, ((1)) ((2))
351 GeV (pi), 280 GeV (p), ((3))
27.7 GeV/c/nucleon, 8.4 GeV-cms/nucleon
> 5*10**5 GeV, 2-5 degrees, ((6)) ((7)) ((8))
approx. 200 GeV/c, 0.5 < |t| < 2.5 GeV**2, ((3))
```

Alphabetical Keyword List

There are three kinds of entries in the alphabetical list:

regular key words boldface and blank space in column 1.

standardized non-key words "*" in column 1; these terms are generally coupled to regular key words. There are also non-key words which have not been standardized; they are not contained in this key word list

terms which are not used "-" in column 1.

Comments or rules of use are given in parentheses. "Restricted use" means that a key word is used only in cases where it is of central importance in the paper considered.

Entries are ordered in the following sorting sequence:

```
blank . ( + | * ) ; - / , < > : ' = 0....9 aA....zZ
```

Keywords list (fragment for Y and Z)

```
                                --- Y ---

-Y* (baryon resonance, hyperon)
-y-dependence (rapidity dependence)
*Yang-Baxter (algebra, Yang-Baxter)
 Yang-Baxter equation
*Yang-Mills (gauge field theory, Yang-Mills)
-Yerevan ES (Erevan ES)
 yield (usually with particles in parentheses)
 ytterbium
```

```
 yttrium
*Yukawa ('potential, Yukawa' or 'coupling, Yukawa')

                                  --- Z ---

*Z(2) (e.g. 'symmetry, Z(2)')
*Z(3) (e.g. 'symmetry, Z(3)')
*Z(N) (e.g. 'symmetry, Z(N)')
*Z(N) x Z(M) (e.g. 'symmetry, Z(N) x Z(M)')
*Z' (postulated particle, Z')
 Z0
*Z0 Z0 (e.g. 'scattering, Z0 Z0')
*Z0(1780) (partial wave P01; 'postulated particle, Z0(1780)')
*Z0(1865) (partial wave D03; 'postulated particle, Z0(1865)')
*Z1(1725) (partial wave P11; 'postulated particle, Z1(1725)')
*Z1(1900) (partial wave P13; 'postulated particle, Z1(1900)')
*Z1(2150) (postulated particle, Z1(2150))
*Z1(2500) (postulated particle, Z1(2500))
 zero mode
*zeta function (e.g. 'regularization, zeta function')
*ZEUS (at HERA; 'magnetic detector, ZEUS')
 zinc
*Zino (postulated particle, Zino)
 zirconium
-Zweig rule (Iizuka-Okubo-Zweig rule)
```

Appendix B

TECAT command line usage

We include here the output of the command `tecat --help` to show how TECAT program is invoked and how different variables involved are parametrized and tuned. We consider it clarifies how experiments have been configured and executed.

```
Usage: tecat [OPTION...] DATADIR
TECAT -- The TExt CATegorization framework
Arturo Montejo Raez - CERN 2004

  -i, --index-collection     Index full text documents (see INDEX OPTIONS)
  -k, --make-folds           Prepares folds for training classifiersm (see
                             FOLDING OPTIONS)
  -K, --test-tree            Tests the trained system using the MLTree
                             algorithm.
  -R, --test-ranked          Tests the trained system of a serial of classes
                             getting top ranked classes using classification
                             score.
  -t, --train-by-class       Train the system using given classifiers against
                             each class (see TRAIN BY CLASS OPTIONS)
  -T, --train-tree           Train the system with a method which consider all
                             the classes at once (the MLTree algorithm).
  -X, --classify             Classifies a document.
  -z, --test-classes         Tests the trained system of a serial of classes
                             returning only classes ranked with positive or
                             over threshold score.

 DATADIR is the directory where the model is or will be stored

 INDEX OPTIONS ------------------------------------------------

  -C, --classes-dir=DIR      Directory where classes for each file are stored
  -E, --no-stemming          Disable perform Porter stemming (default enabled)
  -F, --full-text-dir=DIR    Directory where full text files are stored
```

```
 -m, --min-term-length=N    Minimal length for a term allowed (default 0,
                            means no filtering)
 -r, --min-freq=N           Minimal frequency for a term allowed (default 0, 0
                            means no filtering)
 -s, --stop-words=FILE      File containing stop words for removal (default
                            'stop.txt' at current directory
 -S, --no-stop-words-removal   Disable perform stop words removal (default
                            enabled)
 -x, --max-term-length=N    Maximal length for a term allowed (default 40, 0
                            means no filtering)

FOLDING OPTIONS -----------------------------------------

 -d, --df-feature-selection=N   Selects only features appearing in at least N
                           documents of the training set (default 1, 0 means
                           all selected)
 -f, --folds=N             Number of folds for cross-validation (default 10)
 -g, --ig-feature-selection=N   Selects only N features with the highest
                           information gain (default 50000, 0 means all
                           selected)
 -j, --df-class-selection=N Selects only classes appearing in training,
                           evaluation and test sets and with at least N
                           documents for training (default 1, 0 means select
                           all)
 -N, --no-cosine-normalization   Term weights will not be normalized by the
                           cosine factor
 -w, --weighting-entropy    Weighting scheme using entropy (default TF.IDF)
 -W, --weighting-tfidf      Weighting scheme using TF.IDF (default)

TeCaT works in two modes: training a binary classifier for each class or
training a tree based on binary classifiers. If you specify '--train-by-class'
option a binary classifier for each class will be trained in a one-agains-all
basis. If '--train-tree' is specified, then a ML-Tree will be constructed. For
both approaches, we can specify as many binary calssifiers as wanted (see
BINARY CLASSIFIERS).


CLASSIFIERS OPTIONS --------------------------------

 -b, --beta-F=VALUE         Beta value for F measure (default 1.0).
 -c, --classifier=OPTIONS   A binary classifier to be used by TECAT. See
                            options below.
 -Q, --compute-scut         The S-Cut approach will be apply to compute, using
                            the validation samples, an automatic threshold on
                            the decision value (default off, the decision
                            value is then 0.0).
 -u, --selection-measure=N  Measure to use for selecting best method. Values
                            of N: 0 (by precision), 1 (by recall), 2 (by
                            fallout), 3 (by accuracy), 4 (by error), 5 (by
                            distance to optimal point), 6 (by F measure), 7
                            (by F1 measure, default)
 -U, --selection-threshold=N   Threshold to be used for the selection measure.
                            This value is 0.0 by default. Please, not that
                            this value use to be in [0,1] interval and depends
                            in the selection measure used.

ML-TREE OPTIONS -----------------------------------
```

```
Above options are valid here. Additional, to control tree generation and
behaviour you have:

 -B, --tree-balance=N       Method to be used for determining the balance of a
                            class: 0 (id, default), 1 (1/ig), 2 (id/ig), 3
                            (Sum_children(id)), 4 (Sum_children(id)/ig); where
                            id is the imbalance degree, ig the information
                            gain and Sum_children the summatory of imbalance
                            degrees for potential children.
 -H, --tree-max-children=N  Maximum number of children per node when training
                            a MLTree (default 4, 0 means all children will be
                            accepted until one of them fails in the
                            evaluation). Be carefull when setting this
                            parameter, since the complexity of the MLTree
                            training algorithm is of order O(n^log2(m)) being
                            m the number of classes and n the number of
                            children per level (the tree-max-children
                            parameter).
 -L, --tree-max-depth=N     Maximum depth of the MLTree allowed when
                            generating it (default 7, 0 means no limit).
 -Z, --tree-allow-duplicates   Determines whether classes already trained in a
                            level may appear in lower levels (default off, add
                            this parameter to set it on).

BINARY CLASSIFIERS


There are two types of classifiers: external and built-in. External
classifiers are external programs which will be called by TECAT following. To
call them properly some options has to be passed. Built-in classifiers are
classification algorithms implemented internally by TECAT, so you don't need
to specify anything but certain parameters that the classification method may
require.

OPTIONS for built-in classifiers:

method=[plaum|rocchio|wh|eg|lvq|nb|genetic]
 Selects a built-in classifier to be added to the list of trained classifiers.
They are the PLAUM perceptron (plaum), the Rocchio algorithm (rocchio), the
Widrow-Hoff algorithm (wh), the Kivinen-Warmuth algorithm (kw), the LVQ neural
network (lvq), and the Naive-Bayes network (nb). Each built-in classifier has
additional parameters which can be specified using colons, e.g.:

    --classifier=method=plaum:pos_tau=2.0:neg_tau=-1.5:iter=100

** PLAUM (plaum)
pos_tau:    margin for positive samples (default +1.0)
neg_tau:    margin for negative samples (default -1.0)
iterations: number of maximum iterations (default 50)
eta:        value for eta (default 1.0)
For more information about his algorith, see [1].

** Rocchio (rocchio)
alpha:       weight for previous weights (default 0.0)
beta:        weight for positive samples (default 1.0)
```

```
gamma:       weight for negative samples (default 1.0)
nonnegative: 0 (negative weights are preserved) or 1 (negative weights are set
to 0, default)

** Naive-Bayes (nb)
NOT AVAILABLE

** Widrow-Hoff (wh)
eta:     learning rate (default 1/(4*||x||^2), being ||x|| the maximum norm in
the set of samples). This value must be positive.
     average: 0 (use final computed weight) or 1 (use average of all computed
weights, default)

** Exponentiated Gradient (eg)
Also known as ``Kivinen & Warmuth'' or EG algorithm
eta:     learning rate (default 1/(3*R^2) where R is the difference between
the maximum weight minus the minimum weight of a term.

** LVQ (lvq)
iterations: number of maximum iterations of the LVQ loop (default 50*ncbv)
ncbv:       number of codebook vectors for positive and negative sides
(default 5)
alpha:      initial value for learning rate (default 0.005)
u:          initial value for decreasing rate (default 0.001)

** Genetic algorithm (genetic)
NOT AVAILABLE

--- OPTIONS for external classifiers:

learn=<learn command>
This is the command used to train the classifier. It is an string following
the format specified below.

classify=<classify command>
This is the command use for classifying using the computed model in the
learning phase. It is an string following the format specified below.

format=[vectors|arff]
This is the format used create the samples for learning and classification.

Format for 'learn' and 'classify' strings:
Since we need to place model, samples and predictions files properly in the
invocation of the program, we will use [m], [s], [p] as patterns to represent
the location of those files in the invocation of the program. Also, some
algorithms accept a weight factor for positive samples as a measure against
imbalanced classes, for that you can use the [f] pattern, which will be
replaced by the fraction of positive samples over negative samples. See
FORMATS below

FORMATS:

** vectors
The samples file following the 'vectors' format will contain a sample
(document) at each line in ASCII format, with the class label set to +1 or -1
before list of pairs <features>:<weight>; e.g.
```

```
+1 1:3.4 56:23.4 1002:-0.423233
-1 23:0.0001 435:1.3232002 2023:23.34232
-1 532:3.0 5000:-0.34

** arff
The samples file following the 'arff' format will be compliant with standard
ARFF files used by Weka (see http://www.cs.waikato.ac.nz/~ml/weka/arff.html).

You can specify as many pairs classifiers as you want. Examples:

./tecat --train                                          \
      --classifier=method=plaum                          \
      --classifier=learn="svm_learn -j [f] [s] [m]",             \
                  classify="svm_classify [s] [m] [p]",   \
                  format=vectors
      --classifier=learn="svm_learn [s] [m]",            \
                  classify="svm_classify [s] [m] [p]",   \
                  format=vectors

TESTING OPTIONS ----------------------------------------------

 -P, --top-classes=N        When --test-ranked is used, it determines that N
                            classes with the hights ranked status value will
                            returned.

 -Y, --tree-traverse-all    Dives down both positive and negative subtrees, no
                            matter the classification sign returned by a
                            classifier node (disabled by default).


CLASSIFICATION OPTIONS ---------------------------------------

 -D, --document=FULLTEXT    Path to the plaintext document to be classify.

Option -w (entropy weighting) or -W  (TF.IDF weighting) may be specify to
comply with the way documents were processed in training (default TF.IDF).

References ---------------------------------------------------

[1]. Li. Y. and Zaragoza H., Herbrich R., Shawe-Taylor J. and Kandola J. 'The
Perceptron Algorithm with Uneven Margins'. Proceedings of the International
Conference of Machine Learning (ICML'2002), 2002.

Other options ------------------------------------------------

 -1, --class-correlation    Output the correlation matrix class-to-class of
                            every folder
 -2, --export-models=DIR    Models for each classifier from the first fold
                            will be saved to the given directory, using the
                            string of the class as file name.
 -3, --dump-term-ids        Dumps the translation table of terms to integer
                            ids.
 -4, --dump-class-ids       Dumps the translation table of classes to integer
                            ids.
 -q, --quiet                Produces no verbose output on processing

 -?, --help                 Give this help list
```

```
      --usage                Give a short usage message
  -V, --version              Print program version

Mandatory or optional arguments to long options are also mandatory or optional
for any corresponding short options.

You can distribute this software freely.

Report bugs to <arturo.montejo.raez@cern.ch>.
```

Appendix C

The Wilcoxon test

The *paired t-test* is a common procedure to determine whether two distributions or categorized variables are statistically different within a given confidence interval. The test is useful for distributions whose variables are categorized into separated groups and, therefore, only comparable by pairs at same category level. This correlated-samples t-test makes certain assumptions and can be meaningfully applied only insofar as these assumptions are met:

1. that the scale of measurement for distributions X_A and X_B has the properties of an equal-interval scale;
2. that the differences between the paired values of X_A and X_B have been randomly drawn from the source population; and
3. that the source population from which these differences have been drawn can be reasonably supposed to have a normal distribution.

If there is one or more of these assumptions that we cannot reasonably suppose to be satisfied, then the t-test for correlated samples cannot be legitimately applied. In that case we can apply the *Wilcoxon test* (also know as *Wilcoxon Signed Ranked* test). This test is the non-parametric equivalent of the paired samples t-test. This implies the assumption that both distributions are symmetrical, in which case the mean and medians are identical. Thus, the null hypothesis (usually represented by H_0) considers that for the two distributions the median difference is zero.

We way to compute it is by calculating the differences between paired samples and then rank the absolute value of such differences. Then we take the minimum of the signed ranked values (that is the reason for its second name) and compare this value against a table if few labels are considered (< 30) or with the approximated normal distribution produced by all possible combinations of ranked signs.

category	**Alg. A**	**Alg. B**	$A-B$	$\|A-B\|$	sorted	rank	signed
experimental	0,821	0,839	-0,018	0,018	–	–	–
magnetic	0,896	0,883	0,012	0,012	–	–	–
talk	0,662	0,733	-0,071	0,071	0,007	1	-1
electron	0,923	0,931	-0,007	0,007	0,011	2	2
quark	0,769	0,833	-0,064	0,064	0,012	3	3
CERN LEP	0,945	0,934	0,011	0,011	0,018	4	-4
anti-p p	1,000	1,000	0,000	–	0,029	5	5
Z0	0,909	0,880	0,029	0,029	0,064	6	-6
Batavia Coll	1,000	1,000	0,000	–	0,071	7	-7
mass spectrum	0,800	0,666	0,133	0,133	0,133	8	8

Table C.1: Wilcoxon procedure example

To clarify this process, let's consider the example given at table C.1. We want to compare two algorithms: A and B. For that, we have measured the accuracy for the ten categories placed at the first column. From those values, we compute

1. the difference between paired values,
2. then we take the absolute value and discard zeros (they are not representative),
3. now we rank this values, and
4. finally we attach the sign again.

From the last column we calculate W value, as $W = min(W_+, W_-)$. W_+ and W_- are the absolute sum of positive and negative ranks: $W_+ = 18 = 2{+}3{+}5{+}8$, $W_- = 18 = 1 + 4 + 6 + 7$. In this case W is 18 (since there is no minimum, both have been found to have the same value). Now we can use this value to find in table C.2 the maximum allowed W admitted to guarantee a 95% confidence level (i.e. *p-value* of 0.005). In our case, for $n = 10$ we find that, effectively, $18 > 8$. Thus, we have to reject the null hypothesis and admit that there are significant differences between two distributions from a two-tailored analysis. Even more, we can also study the one-tailored case, which is equivalent to analyze the difference but in favor of one of the algorithms. Surprisingly here, cannot admit any overall outperforming of one algorithm over the other since both W_- and W_+ are equal. But, since $18 > 10$ (the one-tailored threshold found in the table), we must admit that they *do* differ in *both* directions. As we can see by having a look at the results, for certain classes one of the two algorithms should be preferred.

The table of critical values is not available for large distributions (usually $n > 30$). In that case, as was said formerly, we can compute a value z:

$$z = \frac{max(W_+, W_-) \pm 0.5 - \mu}{\sigma} \tag{C.1}$$

n	**Two-tailored**	**One-tailored**
6	0	2
7	2	3
8	3	5
9	5	8
10	8	10
11	10	13
12	13	17
13	17	21
14	21	25
15	25	30
16	29	35
17	34	41
18	40	47
19	46	53
20	52	60
21	58	67
22	65	75
23	73	83
24	81	91
25	89	100

Table C.2: Critical values for W statistic for the Wilcoxon Signed-Ranks test for different numbers of subsets n at significance p=0.05. For significance, W must be less than or equal to the critical value.

where $\mu = \frac{n(n+1)}{4}$ and $\sigma = \frac{n(n+1)(2n+1)}{24}$[1]. We have (being now $W = max(W_+, W_-)$:

$$P(W \geq w) \approx P(Z \geq z) \tag{C.2}$$

We can use the computer to calculate the last probability (the p-value) since $Z \sim N(0,1)$ (is approximated by a normal distribution). We recommend the use of any statistical toolkit (*R*, *Octave*, *SPSS*, *Matlab*, etc.) for solving the calculations involved in this test. In our example we would have:

$\mu = 8(8+1)/4 = 18$

$\sigma = \sqrt{\frac{8(8+1)(2\cdot 8+1)}{24}} = 7,14$

$z = \frac{18+0.5}{7,14=2,59}$

Now, we can use a framework like Octave to obtain the final p-value:

```
octave:1> 1-normal_cdf(2.59)
ans = 0.0047988
```

Therefore, the probability of these two distributions to be different *by chance* is less than 99,6 %. So, again, we reach the same conclusion: they are effectively different.

[1] in these formulae, n is usually the number of values per distribution without counting zero differences

Index