Study and Development of Novel Techniques for Hierarchical Text Categorization

Andrea Addis

Department of Electrical and Electronic Engineering
University of Cagliari

A thesis submitted for the degree of

Philosophiae Doctor (PhD) degree in Electronic and Computer Engineering,
Andrea Addis

Tutor Prof. Giuliano Armano
1. Reviewer: Bettina Berendt

2. Reviewer: Agapito Ledezma

Day of the defense: March 05, 2010, Cagliari

Panel Evaluators

- Prof. Danilo Bruschi (University of Milan, Italy)
- Prof. Francesc J. Ferri (University Valencia, Spain)
- Prof. Giorgio Giacinto (University of Cagliari, Italy)

Signature from head of PhD committee:
To my family, my friends, my work colleagues, and myself.
Acknowledgements

Several people have helped me with this work during all these last years of research. I could spend pages talking about them, whereas I just hidden their names and stories into this thesis, as well as some strips I put in this document to relax yourself between theory and formulas (see pages 4, 26, 50, 64, 80, 105).

So, find yourselves my friends. After all, this thesis is also about Information Retrieval.

---

1 Solutions will be published in my next PhD Thesis
Abstract

The world is widely changing. The impact of the technology and communications revolution has grown greater today. We are taking part in a new era in which many of the world criteria, systems and instruments, especially in the fields of information will be crucially modified in order to make a contribution to this incredible age.

In fact, even if for thousands of years people have realized the importance of archiving and finding information, only nowadays with the advent of computers and the progress of information technology it became possible to store and share large amounts of information, and finding useful information from such collections became a necessity.

My mild contribution to this field started before my PhD, when I studied how to exploit the semantic knowledge of a taxonomy in order to improve text categorization performances. The study of that algorithms has found a natural application in bioinformatics; in particular, an infrastructure for the retrieval and classification of scientific publications has been developed.

The research activity that originated from the above preliminary work is concerned with better investigating hierarchical text categorization and defining novel algorithms in this field. In particular, as real text categorization applications are characterized by a huge imbalance between relevant (positive examples) and non-relevant (negative examples) documents, a novel progressive filtering approach deemed at dealing with this issue has been investigated.

Experimental results, compared with cutting-edge systems, show that the proposed approach performs better than the flat approach. Furthermore, a theoretical study on the advantages of the proposed approach has been done.
The hypotheses formulated during the research activity have been experimented throughout the implementation and adoption of the generic multi-agent architecture X.MAS. In particular, X.MAS has been devised to make it easier the implementation of information retrieval applications. The list of applications that have been developed starting from X.MAS follows: (i) NEWS.MAS, aimed at classifying news articles belonging to online newspapers; (ii) WIKI.MAS, concerned with the problem of classifying Wikipedia contents according to a predefined set of classes; (iii) MAM.MAS, focused on giving a support to professors and students while interacting with a media asset management system; (iv) SEA.MAS, a MultiAgent System devoted to address the problem of monitoring boats in a marine reserve; (v) SSP (Secondary Structure Predictor), a MultiAgent System aimed at predicting secondary structures of proteins; (vi) PAA (Plan Acquisition Architecture), an Architecture exploiting most of the libraries of X.MAS in order to recover and translate web articles into plans.

A further research activity in the field of text categorization comprised the automatic creation of datasets by means of a semantic approach. Experimental results point out that, on average, the performance of the automatic creation is quite similar to that obtained by hand-made classification of document collections. Moreover, a semantic approach to text categorization has been adopted to study user profiling, with particular focus on the creation of contextual advertising.

This work focuses on the study and development of novel techniques for hierarchical text categorization.
# Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>List of Figures</td>
<td>xi</td>
</tr>
<tr>
<td>List of Tables</td>
<td>xiii</td>
</tr>
<tr>
<td>1 Introduction</td>
<td>1</td>
</tr>
<tr>
<td>2 Information Retrieval</td>
<td>3</td>
</tr>
<tr>
<td>2.1 Introduction</td>
<td>3</td>
</tr>
<tr>
<td>2.2 Models and representation</td>
<td>6</td>
</tr>
<tr>
<td>2.2.1 Vector Models</td>
<td>7</td>
</tr>
<tr>
<td>2.2.2 Probabilistic Models</td>
<td>8</td>
</tr>
<tr>
<td>2.2.3 The inference network model</td>
<td>9</td>
</tr>
<tr>
<td>2.3 Evaluation</td>
<td>11</td>
</tr>
<tr>
<td>2.3.1 Precision and recall</td>
<td>13</td>
</tr>
<tr>
<td>2.3.2 Combining precision and recall</td>
<td>14</td>
</tr>
<tr>
<td>2.4 Techniques for Term Weighting</td>
<td>17</td>
</tr>
<tr>
<td>2.5 Multi Agent Systems for Information Retrieval</td>
<td>19</td>
</tr>
<tr>
<td>2.5.1 Agent Oriented Software Engineering</td>
<td>20</td>
</tr>
<tr>
<td>2.5.2 MAS for Information Retrieval</td>
<td>21</td>
</tr>
<tr>
<td>3 Text Categorization</td>
<td>25</td>
</tr>
<tr>
<td>3.1 Introduction</td>
<td>25</td>
</tr>
<tr>
<td>3.2 A formal definition of Text Categorization</td>
<td>28</td>
</tr>
<tr>
<td>3.2.1 Single-Label versus Multi-Label Text Categorization</td>
<td>29</td>
</tr>
<tr>
<td>3.2.2 Hard Categorization versus Ranking Categorization</td>
<td>30</td>
</tr>
<tr>
<td>3.3 How to represent documents, classes, and corresponding classifiers</td>
<td>31</td>
</tr>
</tbody>
</table>
CONTENTS

3.3.1 Document Indexing .................................................. 31
3.3.2 Dimensionality Reduction ........................................... 32
3.3.3 Classifier learning ................................................... 36
3.3.4 Algorithms for Classifier Learning ................................. 39
3.4 Applications of the Text Categorization .............................. 43
  3.4.1 Automatic Indexing for Boolean Information Retrieval Systems .................................................. 44
  3.4.2 Document organization ............................................. 45
  3.4.3 Document Filtering ................................................. 46
  3.4.4 Word sense disambiguation ....................................... 47

4 Hierarchical Text Categorization ........................................... 49
  4.1 Formal Definitions .................................................... 50
  4.2 Reducing the Dimensionality of the Search Space .................. 51
  4.3 Building the Training Set ............................................. 52
  4.4 The Learning Algorithm ............................................. 55
  4.5 Performance Evaluation ............................................. 57
  4.6 Applications .......................................................... 60

5 Hierarchical Text Categorization through a Progressive Filtering Approach ............................................. 63
  5.1 The Progressive Filtering Approach ................................ 65
    5.1.1 Document representation ..................................... 65
    5.1.2 The learning algorithm ..................................... 67
    5.1.3 The input imbalance issue .................................. 68
  5.2 The Impact of PFA .................................................... 68
    5.2.1 Overall Transformation Performed by a Pipeline of Classifiers ............................................. 68
  5.3 The Threshold-Selection Algorithm ................................ 71
    5.3.1 How Precision, Recall and F1 Change along a Pipeline ............................................. 74
    5.3.2 Comparing PFA with the FLAT Approach ..................... 76
  5.4 Learning Complexity of TSA ....................................... 77
# A Generic MultiAgent Architecture for Information Retrieval

6.1 The Abstract Architecture ........................................ 79

6.2 The Concrete Architecture ........................................ 80

- 6.2.1 X.MAS Macro-Architecture .................................. 82
- 6.2.2 X.MAS Micro-Architecture .................................. 83

- 6.2.3 Comparing X.MAS with other MAS solution to IR ............. 85

6.3 Building Information Retrieval Systems by Using X.MAS ........ 86

- 6.3.1 NEWS.MAS: News Retrieval through X.MAS ................. 86
  - 6.3.1.1 The Scenario. ........................................... 86
  - 6.3.1.2 The Implementation. .................................... 86

- 6.3.2 WIKI.MAS: X.MAS for Classifying Wikipedia Contents ....... 88
  - 6.3.2.1 The Scenario. ........................................... 88
  - 6.3.2.2 The Implementation. .................................... 89

- 6.3.3 MAM.MAS: X.MAS for a Media Asset Management System ..... 91
  - 6.3.3.1 The Scenario. ........................................... 91
  - 6.3.3.2 The Implementation. .................................... 92

- 6.3.4 SEA.MAS: X.MAS for Monitoring Boats in Marine Reserves .. 93
  - 6.3.4.1 The Scenario. ........................................... 93
  - 6.3.4.2 The Implementation. .................................... 94

- 6.3.5 PACMAS/BIO: X.MAS for Classifying Bioinformatic Publications 95
  - 6.3.5.1 The Scenario. ........................................... 96
  - 6.3.5.2 The Implementation. .................................... 96

- 6.3.6 SSP: X.MAS for Protein Secondary Structure Prediction ...... 98
  - 6.3.6.1 The Scenario. ........................................... 98
  - 6.3.6.2 The Implementation. .................................... 99

- 6.3.7 PAA: X.MAS for Recovering Plans from the Web .............. 101
  - 6.3.7.1 The Scenario. ........................................... 101
  - 6.3.7.2 The Implementation. .................................... 102

# Experimental Results

7.1 The Adopted Datasets ............................................ 105

- 7.1.1 Reuters ..................................................... 106
- 7.1.2 DMOZ ..................................................... 107
List of Figures

2.1 Strip ................................................................. 4
2.2 An example of basic model of Inference Network ..................... 10
3.1 Strip ................................................................. 26
3.2 SVM Algorithm, decision surfaces .................................... 39
3.3 The method of the nearest neighbor ................................... 42
4.1 Strip ................................................................. 50
4.2 Categories of training sets. .......................................... 53
5.1 Strip ................................................................. 64
5.2 A taxonomy and its corresponding pipelines. ......................... 65
5.3 An example of pipeline \{C_1 \cdots C_L\} .............................. 69
5.4 Unfolding the threshold-selection procedure for a pipeline composed by three classifiers. ........................................ 74
6.1 Strip ................................................................. 80
6.2 The abstract architecture. ............................................ 81
6.3 The concrete architecture. ............................................ 82
6.4 Agent internals. ..................................................... 84
6.5 The architecture of NEWS.MAS ................................. 87
6.6 WIKI.MAS at a glance ............................................... 88
6.7 Wikipedia Page ...................................................... 89
6.8 A portion of the adopted taxonomy. ................................ 92
6.9 The SEA.MAS System .............................................. 94
6.10 The TAMBIS Ontology ............................................ 97
## LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.11</td>
<td>Typical implementation of the proposed generic multiagent architecture</td>
<td>99</td>
</tr>
<tr>
<td>6.12</td>
<td>Expert Internal Architecture</td>
<td>100</td>
</tr>
<tr>
<td>6.13</td>
<td>The Structure of a WikiHow Page</td>
<td>101</td>
</tr>
<tr>
<td>6.14</td>
<td>The Plan Learn Architecture</td>
<td>102</td>
</tr>
<tr>
<td>7.1</td>
<td>Strip</td>
<td>105</td>
</tr>
<tr>
<td>7.2</td>
<td>An example of pipeline in the Reuters corpus</td>
<td>108</td>
</tr>
<tr>
<td>7.3</td>
<td>DMOZ</td>
<td>109</td>
</tr>
<tr>
<td>7.4</td>
<td>An extract of the DMOZ taxonomy</td>
<td>115</td>
</tr>
<tr>
<td>7.5</td>
<td>Comparison of precision in the Reuters dataset</td>
<td>116</td>
</tr>
<tr>
<td>7.6</td>
<td>Comparison of recall in the Reuters dataset</td>
<td>116</td>
</tr>
<tr>
<td>7.7</td>
<td>Comparison of precision in DMOZ dataset</td>
<td>117</td>
</tr>
<tr>
<td>7.8</td>
<td>Comparison of recall in DMOZ dataset</td>
<td>117</td>
</tr>
<tr>
<td>7.9</td>
<td>Performance improvement on Reuters dataset</td>
<td>118</td>
</tr>
<tr>
<td>7.10</td>
<td>Performance improvement on DMOZ dataset</td>
<td>118</td>
</tr>
<tr>
<td>7.11</td>
<td>Hierarchical measures on Reuters dataset</td>
<td>119</td>
</tr>
<tr>
<td>7.12</td>
<td>Hierarchical measures on DMOZ dataset</td>
<td>119</td>
</tr>
</tbody>
</table>
List of Tables

1.1 Structure of the Thesis ........................................... 2
2.1 Operators supported by the IN model ............................... 11
2.2 Confusion Matrix .................................................... 13
2.3 Agent-based information retrieval systems at a glance. ......... 24
3.1 Main Functions used for Term Space Reduction purposes ........ 34
5.1 An example of Stop Word List ..................................... 66
5.2 Some example of Stemming ......................................... 66
5.3 PFA vs. FLAT approach. ........................................... 76
7.1 Performance improvement on Reuters dataset ...................... 112
7.2 Performance improvement on DMOZ dataset ....................... 113
7.3 TSA vs greedy ....................................................... 114
7.4 $F_1$ in presence of input imbalance ............................... 114
Glossary

**IR** – Information Retrieval (chapter 2).

**IN** – Inference Network, **DN** – Document Network, **QN** – Submitted Query: Acronyms related to Inference Network Model, a technique to model document retrieval as an inference process in an inference network (section 2.2.3).


**tf*idf** – Term Frequency * Inverse of Document Frequency: a technique for weighting terms during a feature selection process (section 2.4).

**MAS** – Multi Agent Systems: A system composed by a pool of Software Agents (section 2.5).

**TC** – Text Categorization (chapter 3).

**DPC** – Document Pivoted Categorization, **CPC** – Category Pivoted Categorization (section 3.2.2).

**DR** – Dimensionality Reduction: phase often applied so as to reduce the size of the document representations from $T$ to a much smaller, predefined number (section 3.3.1).

**TSR** – term space reduction: Reduction of the dimensionality of the space of terms, selected as features for classification purposes (section 3.3.2).

**LSI** – Latent Semantic Indexing: a DR technique developed in IR in order to address the problems deriving from the use of synonymous, near synonymous, and polysemous words as dimensions of document representations (section 3.3.2).

**CSV** – Categorization Status Value: a real-valued functions of the form $CSV : D \times C \rightarrow [0,1]$ generated by the classifier to describe the relevance of the category (section 3.3.3).

**kNN** – k Nearest Neighbour: the k nearest neighbour prototypes that influence the classification process; **wkNN** – the weighted variant of kNN (section 3.3.4).

**SVM** – Support Vector Machines: an effective method in the classifier learning arena (section 3.3.4).

**WSD** – Word Sense Disambiguation: the activity of finding, given the occurrence in a text of an ambiguous (i.e., polysemous or homonymous) word (section 3.4.4).

**HTC** – Hierarchical Text Categorization: The study of the impact of the relation among classes in the Text Categorization field (chapter 4).
Chapter 1

Introduction

A hierarchical organization of entities or notions is very helpful for humans to retain, find and analyze things (20). The main advantage of the hierarchical perspective is that, according to the “divide et impera” philosophy, the problem is partitioned into smaller subproblems, each being effectively and efficiently managed. Therefore, it is not surprising that in the Web 2.0 age people organize large collections of web pages, articles or emails in hierarchies of topics or arrange a large body of knowledge in ontologies. Such organization allows to focus on a specific level of details ignoring specialization at lower levels and generalization at upper levels. In this scenario, the main goal of automatic categorization systems is to deal with reference taxonomies in an effective and efficient way.

Furthermore, due to the increasing importance of term polysemy for large corpora, both precision and recall decrease as the number of categories increases (21) (22), so that considering categories organized in a hierarchy may help to improve the overall performances. Another important issue is concerned with the benefits that a hierarchical approach can give in real-world scenarios, typically characterized by imbalanced data (23). In fact, in these contexts, relevant and non relevant documents (i.e., positive and negative examples, respectively) are typically imbalanced, turning classifiers trained with the same percent of positive and negative examples into not adequate tools.

This thesis presents a novel progressive filtering approach (PFA) method for text categorization, aimed at handling these issues. A theoretical study on the impact of the proposed approach to text categorization is presented with a theoretical and experimental comparison between the performances of the algorithm devised to optimize
1. INTRODUCTION

the performance of categorization vs. a greedy approach. Furthermore the instruments
deeded useful to reach this purpose (in particular software multiagent systems) are an-
alyzed and exploited in order to build a powerful generic infrastructure to prove PFA
algorithm effectiveness.

The scenario that hosts these challenges is complex, and being necessary to make
a general paint of the state of the art, the corresponding fields of interest have been
analyzed in a comprehensive study.

<table>
<thead>
<tr>
<th>Introduction</th>
<th>Chapter 1</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>State of the art</strong></td>
<td></td>
</tr>
<tr>
<td>Information Retrieval</td>
<td>Chapter 2</td>
</tr>
<tr>
<td>Text Categorization</td>
<td>Chapter 3</td>
</tr>
<tr>
<td>Hierarchical Text Categorization</td>
<td>Chapter 4</td>
</tr>
<tr>
<td><strong>The proposed approach</strong></td>
<td></td>
</tr>
<tr>
<td>A Progressive Filtering Algorithm</td>
<td>Chapter 5</td>
</tr>
<tr>
<td>X.MAS, an Architecture for Information Retrieval</td>
<td>Chapter 6</td>
</tr>
<tr>
<td>Experimental Results</td>
<td>Chapter 7</td>
</tr>
<tr>
<td>Conclusions and Future Directions</td>
<td>Chapter 8</td>
</tr>
</tbody>
</table>

Table 1.1: Structure of the Thesis

Table [1.1] shows that the thesis is organized into two main sections over Introduc-
tion and Conclusions: the State of the Art and the Proposed Approach. In the “State
of the art” section the general field of Information Retrieval is analyzed together with
systems aimed at performing information retrieval and Multi Agent Systems. Secondly,
Text Categorization, which is the field of experimentation of this thesis, is studied, in
particular theory and tools are analyzed before being used. Then Hierarchical Text
Categorization research field, to which the PFA algorithm belongs, is presented. In the
“Proposed Approach” section my contribute is presented as a novel way to improve hi-
erarchical categorization systems. A progressive filtering algorithm is described and its
effectiveness and efficiency discussed from a theoretical point of view, whereas experi-
mental results performed with the help of a powerful architecture built for information
retrieval purposes validates the approach from a pragmatic perspective.
Chapter 2

Information Retrieval

For thousands of years people have realized the importance of archiving and finding information. With the advent of computers and the progress of information technology, it became possible to store large amounts of information and finding useful information from such collections became a necessity. The field of Information Retrieval was born in the 1950s out of this necessity. Over the last fifty years, the field has matured considerably. Several Information Retrieval systems are used on an everyday basis by a wide variety of users.

My aim throughout is to give a comprehensive coverage of the most current important ideas in various special areas of information retrieval. In particular, I will be placing emphasis on the use of automatic classification techniques and (in the chapter 3) rigorous methods of measurement of effectiveness in the field of text categorization, which is the field of study of this thesis work.

2.1 Introduction

The world has widely changed in terms of communicating, acquiring and storing information. To handle such amount of information has become a challenging issue: hundreds of millions of people engage in Information Retrieval every day when they use a web search engine or search their email, making such field to become the dominant form of information access, overtaking traditional database-style searching.

As an academic field of study, according to [24], information retrieval might be defined as:
2. INFORMATION RETRIEVAL

Information retrieval (IR) is finding material (usually documents) of an unstructured nature (usually text) that satisfies an information need from within large collections (usually stored on computers).

In fact, generally, IR refers to the retrieval of unstructured data. Most often, it is related to Text Retrieval, i.e., the retrieval of textual documents such as newspaper and magazine articles or Web documents. Other types of retrieval include, for example, WebIR, Image Retrieval, Video Retrieval, and Music Retrieval. The retrieval of information across languages is called Cross-Language IR [25].

The word information can be very misleading. In the context of IR, information, in the technical meaning given in Shannon’s theory of communication, is not readily measured [26]. In fact, in many cases one can adequately describe the kind of retrieval by simply substituting “document” for “information” [27].

More generally, an IR system does not inform (i.e., change the knowledge of) the user on the subject of his inquiry. It merely informs on the existence (or non-existence) and whereabouts of documents relating to his request. This specifically excludes Question-Answering systems as typified by Winograd [28] and those described by Minsky [29]. In IR this may sometimes be of interest but more generally we want to find those items that partially match the request and then select from those a few of the best matching ones.

![Figure 2.1: Strip - No Day Without the Internet](image)

IR can cover various and heterogeneous kinds of data and information problems beyond that specified in the core definition above. The term “unstructured data” refers
to data which does not have clear, semantically overt, easy-for-a-computer structure. It is the opposite of structured data, the canonical example of which is a relational database. In reality, almost no data are truly “structured” or “unstructured”. This is definitely true of all text data if you count the latent linguistic structure of human languages. In fact it is often desirable to facilitate IR using “semistructured” information, as for instance finding a document where the title contains *Java* and the body contains *threading* (24).

It is possible and useful to distinguish IR systems by the scale at which they operate. The prominent ones are three:

1. Web Search;

2. Personal IR;

3. Domain-specific Search.

In **web search**, the system has to provide search over billions of documents stored on millions of computers. Distinctive issues are needing to gather documents for indexing, being able to build systems that work efficiently at this enormous scale, handling particular aspects of the web, such as the exploitation of hypertext, and not being fooled by site providers manipulating page content in an attempt to boost their search engine rankings, given the commercial importance of the web.

At the other extreme is **personal IR**. In the last few years, consumer operating systems have integrated information retrieval (such as Apple’s Mac OS X Spotlight or Windows Vista’s Instant Search). Email programs usually not only provide search but also text classification: they at least provide a spam (junk mail) filter, and commonly also provide either manual or automatic means for classifying mail so that it can be placed directly into particular folders. Distinctive issues here include handling the broad range of document types on a typical personal computer, and making the search system maintenance free and sufficiently lightweight in terms of startup, processing, and disk space usage that it can run on one machine without annoying its owner. In between is the space of enterprise, institutional, and **domain-specific search**, where retrieval might be provided for collections such as a corporation internal documents, a database of patents, or research articles on biochemistry. In this case, the documents will typically be stored on centralized file systems and one or a handful of dedicated machines will provide search over the collection (24).
2. INFORMATION RETRIEVAL

2.2 Models and representation

Boolean systems were the first IR systems allowing users to specify their information need using a complex combination of boolean ANDs, ORs and NOTs. It was quickly clear that boolean systems have several shortcomings, e.g., there is no inherent notion of document ranking, and it is very hard for a user to form a good search request (i.e., a good search request relies on the ability and knowledge of the user). Even though boolean systems usually return matching documents in some order, e.g., ordered by date, or some other document feature, relevance ranking is often not critical in a boolean system. It has been shown by the research community that boolean systems are less effective than ranked retrieval systems, however many power users still use boolean systems as they feel more in control of the retrieval process. However, most everyday users expect IR systems to do ranked retrieval. IR systems rank documents by their estimation of the usefulness of a document for a user query. Most IR systems assign a numeric score to every document and rank documents by this score.

The Boolean retrieval model contrasts with ranked retrieval models such as the vector space model (below discussed), in which users largely use free text queries, that is, just typing one or more words rather than using a precise language with operators for building up query expressions, and the system decides which documents best satisfy the query. Despite decades of academic research on the advantages of ranked retrieval, systems implementing the Boolean retrieval model were the main or only search option provided by large commercial information providers for three decades until the early 1990s (approximately the date of arrival of the World Wide Web). However, these systems did not have just the basic Boolean operations (AND, OR, and NOT) presented so far. A strict Boolean expression over terms with an unordered results set is too limited for many of the information needs that people have, and these systems implemented extended Boolean retrieval models by incorporating additional operators such as term proximity operators. A proximity operator is a way of specifying that two terms in a query must occur close to each other in a document, where closeness may be measured by limiting the allowed number of intervening words or by reference to a structural unit such as a sentence or paragraph (24).

Several more effective models have been proposed for this process. According to (30) the three most used models in IR research are the vector space model, the probabilistic
2.2 Models and representation

models, and the inference network model.

2.2.1 Vector Models

In the vector space model, text is represented by a vector of terms \( (31) \). The representation of a set of documents as vectors in a common vector space is known as the vector space model and is fundamental to a host of information retrieval operations ranging from scoring documents on a query, document classification and document clustering. We first develop the basic ideas underlying vector space scoring; a pivotal step in this development is the view of queries as vectors in the same vector space as the document collection \( (24) \). A document vector captures the relative importance of the terms in a document. The definition of a term is not inherent in the model, but terms are typically words and phrases. If words are chosen as terms, then every word in the vocabulary becomes an independent dimension in a very high dimensional vector space. Any text can then be represented by a vector in this high dimensional space. If a term belongs to a text, it gets a non-zero value in the text-vector along the dimension corresponding to the term. Since any text contains a limited set of terms (the vocabulary can be millions of terms), most text vectors are very sparse. Most vector based systems operate in the positive quadrant of the vector space, i.e., no term is assigned a negative value \( (30) \).

To assign a numeric score to a document for a query, the model measures the similarity between the query vector (since query is also just text and can be converted into a vector) and the document vector. The similarity between two vectors is once again not inherent in the model. Typically, the angle between two vectors is used as a measure of divergence between the vectors, and cosine of the angle is used as the numeric similarity (since cosine has the nice property that it is 1.0 for identical vectors and 0.0 for orthogonal vectors). As an alternative, the inner-product (or dot-product) between two vectors is often used as a similarity measure. If all the vectors are forced to be unit length, then the cosine of the angle between two vectors is the same as their dot-product. If \( \vec{D} \) is the document vector and \( \vec{Q} \) is the query vector, then the similarity of document \( D \) to query \( Q \) (or score of \( D \) for \( Q \)) can be represented as:

\[
Sim(\vec{D}, \vec{Q}) = \sum_{t_i \in Q,D} w_{t_iQ} \cdot w_{t_iD} \tag{2.1}
\]
where \( w_{t,\vec{Q}} \) is the value of \( i \)th component in the query vector \( \vec{Q} \), and \( w_{t,\vec{D}} \) is the \( i \)th component in the document vector \( \vec{D} \).

By denoting by \( \vec{V}(d) \) the vector derived from document \( d \), with one component in the vector for each dictionary term, the set of documents in a collection may be viewed as a set of vectors in a vector space, in which there is one axis for each term. This representation loses the relative ordering of the terms in each document. The challenge is to quantify the similarity between two documents in this vector space. A first attempt might consider the magnitude of the vector difference between two document vectors. This measure suffers from a drawback: two documents with very similar content can have a significant vector difference simply because one is much longer than the other. Thus the relative distributions of terms may be identical in the two documents, but the absolute term frequencies of one may be far larger.

### 2.2.2 Probabilistic Models

This family of IR models is based on the general principle that documents in a collection should be ranked by decreasing probability of their relevance to a query. This is often called the probabilistic ranking principle (PRP) \(^{32}\). Since true probabilities are not available to an IR system, probabilistic IR models estimate the probability of relevance of documents for a query. This estimation is the key part of the model, and this is where most probabilistic models differ from one another. The initial idea of probabilistic retrieval was proposed by Maron and Kuhns in a paper published in 1960 \(^{33}\). Since then, many probabilistic models have been proposed, each based on a different probability estimation technique.

As a description of the basics of the probabilistic model, being \( P(R|D) \) the relevance for a document \( D \), and \( P(\bar{R}|D) \) the probability that the document is non-relevant, we can rank the documents with the formulation: \( \log \frac{P(R|D)}{P(\bar{R}|D)} \).

Under the assumption of independence, generally valid among terms (typically words), we can use Bayesian estimation to find that the probability of presence/absence of a term in relevant/non-relevant documents as \(^{30}\):

\[
P(D|R) = \prod_{t_i \in Q,D} P(t_i|R) \cdot \prod_{t_i \in Q,D} (1 - P(t_j|R)) \tag{2.2}
\]
which uses probability of presence of a term $t_i$ in relevant document for all terms that are common to the query and the document, and the probability of absence of a term $t_j$ from relevant documents for all terms that are present in the query and absent from the document. If $p_i$ denotes $P(t_i|R)$ and $q_i$ denotes $P(t_i|\bar{R})$, the ranking formula reduces to (30)

$$\log \prod_{t_i \in Q,D} \frac{p_i \cdot (1-q_i)}{p_i \cdot (1-p_i)} \text{ or } \sum_{t_i \in Q,D} \log \frac{p_i \cdot (1-q_i)}{p_i \cdot (1-p_i)}$$

(2.3)

Different assumptions for estimation of $p_i$ and $q_i$ yield different document ranking functions.

### 2.2.3 The inference network model

In this model, document retrieval is modeled as an inference process in an inference network (34). Most techniques used by IR systems can be implemented under this model. In the simplest implementation of this model a document instantiates a term with a certain strength, and the credit from multiple terms is accumulated given a query to compute the equivalent of a numeric score for the document. From an operational perspective, the strength of instantiation of a term for a document can be considered as the weight of the term in the document, and document ranking in the simplest form of this model becomes similar to ranking in the vector space model and the probabilistic models described above. The strength of instantiation of a term for a document is not defined by the model, and any formulation can be used.

The Inference Network (IN) model is able to perform a ranking given many sources of evidence by performing a combination of evidence. The IN model is basically a Bayesian Network used to model documents, the document contents, and the query. The IN consists of two sub-networks: the Document Network (DN) produced during indexing and then static during retrieval; the Query Network (QN) produced from the query text during retrieval.

An example of basic model of Inference Network consists of two component network: a document network and a query network.

The DN represents the document collection and consists of nodes for each document (called document nodes) and nodes for each concept with the collection (document concept nodes). The document nodes represent the retrievable units within the network,
2. INFORMATION RETRIEVAL

Figure 2.2: An example of basic model of Inference Network - An example of basic model of Inference Network, taken from http://www.dcs.qmul.ac.uk/andrew/pubs/sigir02/html/node5.htm.

that is, those items we wish to see in the resultant ranking. A causal link (represented as $\rightarrow$) between document node and the document concept node indicates that the document content is represented by the concept. Each link contains a conditional probability, or weight, to indicate the strength of the relationship. The evaluation of a node is done using the value of the parent nodes and the conditional probabilities.

The QN represents the submitted query and consists of a framework of nodes that represent the required concepts (query concept nodes) and the operators (query operator nodes), connected in an inverted tree structure. The QN is constructed with a final leaf (the node I in the figure 2.2) that represents the user Information Need. The framework permits statistical operators and statistical approximations of the Boolean operators, a number of which are given in Table 2.1 (as in the INQUERY implementation).

Two further processes are done to perform retrieval: the attachment process, where by the QN is attached to the DN to form the complete IN and is done where concepts in both networks are the same; the evaluation process, whereby the complete IN is evaluated for each document node to form the probability of the relevance to the query. The evaluation is initialized by setting the output of one document node to 1 and all...
2.3 Evaluation

<table>
<thead>
<tr>
<th>Operator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>and</td>
<td>AND the terms</td>
</tr>
<tr>
<td>or</td>
<td>OR the terms</td>
</tr>
<tr>
<td>not</td>
<td>Negate the term (incoming belief)</td>
</tr>
<tr>
<td>sum</td>
<td>Sum of the incoming beliefs</td>
</tr>
<tr>
<td>wsum</td>
<td>Weighted sum of the incoming beliefs</td>
</tr>
<tr>
<td>max</td>
<td>Maximum of the incoming beliefs</td>
</tr>
</tbody>
</table>

Table 2.1: Operators supported by the IN model

The other document nodes to 0. This is done for each document node in turn and the network is evaluated (see (36) for details and examples on how nodes are evaluated). The probability of document relevance is taken from the final node I and it is used to produce the ranking.

2.3 Evaluation

Objective evaluation of search effectiveness has been a cornerstone of IR. Progress in the field critically depends upon experimenting with new ideas and evaluating the effects of these ideas, especially given the experimental nature of the field. Since the early years, it was evident to researchers in the community that objective evaluation of search techniques would play a key role in the field. The Cranfield tests, conducted in 1960s, established the desired set of characteristics for a retrieval system. Even though there has been some debate over the years, the two desired properties that have been accepted by the research community for measurement of search effectiveness are recall, i.e., the proportion of relevant documents retrieved by the system; and precision, i.e., the proportion of retrieved documents that are relevant (37). Their definition will be completed in section 2.3.1.

The main question “what to evaluate?” boils down to what can we measure that will reflect the ability of the system to satisfy the user. As early as 1966, Cleverdon gave an answer to this. He listed six main measurable quantities (27):

1. the coverage of the collection, that is, the extent to which the system includes relevant matter;

2. the time lag, that is, the average interval between the time the search request is made and the time an answer is given;
2. INFORMATION RETRIEVAL

3. the form of presentation of the output;

4. the effort involved on the part of the user in obtaining answers to his search requests;

5. the recall of the system, that is, the proportion of relevant material actually retrieved in answer to a search request;

6. the precision of the system, that is, the proportion of retrieved material that is actually relevant.

It is claimed that points 1 and 4 are readily assessed. It is recall and precision which attempt to measure what is now known as the effectiveness of the retrieval system. In other words it is a measure of the ability of the system to retrieve relevant documents while at the same time holding back non-relevant one. It is assumed that the more effective the system the more it will satisfy the user. It is also assumed that precision and recall are sufficient for the measurement of effectiveness.

A popular alternative has been recall and fall-out (the proportion of non-relevant documents retrieved). However, all the alternatives still require the determination of relevance in some way. The relationship between the various measures and their dependence on relevance will be made more explicit later.

The final question “How to evaluate?” has a large technical answer. It is interesting to note that the technique of measuring retrieval effectiveness has been largely influenced by the particular retrieval strategy adopted and the form of its output. For example, when the output is a ranking of documents an obvious parameter such as rank position is immediately available for control. Using the rank position as cut-off, a series of precision/recall values could then be calculated, one part for each cut-off value. The results could then be summarized in the form of a set of points joined by a smooth curve. The path along the curve would then have the immediate interpretation of varying effectiveness with the cut-off value. Unfortunately, the kind of question this form of evaluation does not answer is, for example, how many queries did better than average and how many did worse? Nevertheless, we shall need to spend more time explaining this approach to the measurement of effectiveness since it is the most common approach and needs to be understood.
2.3 Evaluation

Before proceeding to the technical details relating to the measurement of effectiveness it is as well to examine more closely the concept of relevance: relevance is a subjective notion. Different users may differ about the relevance or non-relevance of particular documents to given questions. However, the difference is not large enough to invalidate experiments which have been made with document collections for which test questions with corresponding relevance assessments are available. These questions are usually elicited from bona fide users, that is, users in a particular discipline who have an information need. The relevance assessments are made by a panel of experts in that discipline. So we now have the situation where a number of questions exist for which the correct responses are known. It is a general assumption in the field of IR that should a retrieval strategy fare well under a large number of experimental conditions then it is likely to perform well in an operational situation where relevance is not known in advance.

There is a concept of relevance which can be said to be objective and which deserves mention as an interesting source of speculation. This notion of relevance has been explicated by Cooper (38). It is properly termed “logical relevance”. Its usefulness in present day retrieval systems is limited. However, it can be shown to be of some importance when it is related to the development of question-answering systems, such as the one recently designed by T. Winograd at Massachusetts Institute of Technology.

2.3.1 Precision and recall

Effectiveness is purely a measure of the ability of the system to satisfy the user in terms of the relevance of documents retrieved. Initially, effectiveness can be measured exploiting precision and recall; a similar analysis could be given for any pair of equivalent measures.

It is helpful at this point to introduce the famous confusion matrix (also called contingency table in the IR context) depicted in table 2.2.

<table>
<thead>
<tr>
<th>documents</th>
<th>deemed non-relevant</th>
<th>deemed relevant</th>
</tr>
</thead>
<tbody>
<tr>
<td>negative</td>
<td>true negative (TN)</td>
<td>false positive (FP)</td>
</tr>
<tr>
<td>positive</td>
<td>false negative (FN)</td>
<td>true positive (TP)</td>
</tr>
</tbody>
</table>

| Table 2.2: Confusion Matrix |

13
2. INFORMATION RETRIEVAL

Such table is a visualization tool typically used in supervised learning (where it is also called a matching matrix). Each row of the matrix represents the instances in a predicted class, while each column represents the instances in an actual class. One benefit of a confusion matrix is that it is easy to see if the system is confusing two classes (i.e., commonly mislabeling one as another).

In an information retrieval scenario, Precision is defined as the number of relevant documents retrieved by a search divided by the total number of documents retrieved by that search (namely $\pi = TP/(TP + FP)$), and Recall is defined as the number of relevant documents retrieved by a search divided by the total number of existing relevant documents (which should have been retrieved, namely $\rho = TP/(TP + FN)$).

It is well accepted that a good IR system should retrieve as many relevant documents as possible (i.e., have a high recall), and it should retrieve very few non-relevant documents (i.e., have high precision). Unfortunately, these two goals have proved to be quite contradictory over the years. Techniques that tend to improve recall tend to hurt precision and vice-versa; for example, if system designers feel that precision is more important to their users, they can use precision in top ten or twenty documents as the evaluation metric. On the other hand, if recall is more important to users, one could measure precision at (say) 50% recall, which would indicate how many non-relevant documents a user would have to read in order to find half the relevant ones.

2.3.2 Combining precision and recall

The table 2.2 represents the interesting part of the results. Anyway each value could have a different weight depending on the context. There are techniques allowing to combine all these values in order to find an evaluation that wraps all the information about how well a system is performing.

The accuracy issue

An obvious method that may occur to the reader is to judge an information retrieval system by its accuracy, that is, the fraction of its classifications that are correct. In terms of the confusion matrix above, accuracy $= (TP + TN)/(TP + FP + FN + TN)$. This seems plausible, since there are two actual classes, relevant and non-relevant, and an information retrieval system can be thought of as a two-class classifier which attempts to label them as such (it retrieves the subset of documents which it believes
to be relevant). This is precisely the effectiveness measure often used for evaluating machine learning classification problems.

There is a good reason why accuracy is not an appropriate measure for information retrieval problems. In almost all circumstances, the data is extremely skewed: normally over 99.9% of the documents are in the non-relevant category. A system tuned to maximize accuracy can appear to perform well by simply deeming all documents non-relevant to all queries. Even if the system is quite good, trying to label some documents as relevant will almost always lead to a high rate of false positives. However, labeling all documents as non-relevant is completely unsatisfying to an information retrieval system user. Users are always going to want to see some documents, and can be assumed to have a certain tolerance for seeing some false positives providing that they get some useful information. The measures of precision and recall concentrate the evaluation on the return of true positives, asking what percentage of the relevant documents have been found and how many false positives have also been returned.

The advantage of having the two numbers for precision and recall is that one is more important than the other in many circumstances. Typically, web surfers would like every result on the first page to be relevant (high precision) but have not the slightest interest in knowing let alone looking at every document that is relevant. In contrast, various professional searchers such as paralegals and intelligence analysts are very concerned with trying to get as high recall as possible, and will tolerate fairly low precision results in order to get it. Individuals searching their hard disks are also often interested in high recall searches. Nevertheless, the two quantities clearly trade off against one another: you can always get a recall of 1 (but very low precision) by retrieving all documents for all queries! Recall is a non-decreasing function of the number of documents retrieved. On the other hand, in a good system, precision usually decreases as the number of documents retrieved increase. In general, we want to get some amount of recall while tolerating only a certain percentage of false positives.

The F-measure

A single measure that trades off precision versus recall is the F-measure based on van Rijsbergen’s effectiveness measure \( E = 1 - \frac{\alpha}{P} + \frac{1 - \alpha}{R} \) being its relationship \( F_\beta = (1 - E) \) where \( \alpha = 1/(\beta^2 + 1) \).
The F measure is the weighted harmonic mean of precision and recall:

$$F = \frac{1}{\alpha \frac{P}{R} + (1 - \alpha) \frac{R}{P}} = \frac{(1 + \beta^2) \ast (\text{precision} \ast \text{recall})}{(\beta^2 \ast \text{precision} \ast \text{recall})}$$

where \( \alpha \in [0, 1] \) and thus \( \beta^2 \in [0, \infty] \). The default balanced F measure equally weights precision and recall, which means making \( \alpha = \frac{1}{2} \) or \( \beta = 1 \). It is commonly written as \( F_1 \), which is short for \( F_{\beta=1} \), even though the formulation in terms of \( \alpha \) more transparently exhibits the F measure as a weighted harmonic mean. When using \( \beta = 1 \), the formula on the right simplifies to:

$$F_{\beta=1} = \frac{2PR}{P + R}$$

However, using an even weighting is not the only choice. Values of \( \beta < 1 \) emphasize precision, while values of \( \beta > 1 \) emphasize recall. For example, a value of \( \beta = 3 \) or \( \beta = 5 \) might be used if recall is to be emphasized. Recall, precision, and the F measure are inherently measures between 0 and 1, but they are also very commonly written as percentages, on a scale between 0 and 100.

### Averaging techniques

Two conventional metrics for calculating the performances of a text categorization system: micro- and macro-averaging. Micro-averaged values (see equation 2.4) are calculated by constructing a global contingency table and then calculating precision and recall using these sums in order to compute the \( F_1 \) measure (defined in the next section 2.3.2). In contrast, macro-averaged scores (see equation 2.5) are calculated by first calculating precision and recall for each category and then taking their average. The notable difference between these two metrics is that micro-averaging gives equal weight to every document (it is called a document-pivoted measure) while macro-averaging gives equal weight to every category (category-pivoted measure).

$$\pi^\mu = \frac{\sum_{i=1 \ldots m} (TP_i)}{\sum_{i=1 \ldots m} (TP_i + FP_i)} \quad \text{and} \quad \rho^\mu = \frac{\sum_{i=1 \ldots m} (TP_i)}{\sum_{i=1 \ldots m} (TP_i + FN_i)} \quad (2.4)$$

$$\pi^M = \frac{\sum_i^m (\pi_i)}{m} \quad \text{and} \quad \rho^M = \frac{\sum_i^m (\rho_i)}{m} \quad (2.5)$$

The method of pooling or averaging of the individual \( P/R \) curves seems to have depended largely on the retrieval strategy employed. When retrieval is done by co-ordination level, micro-evaluation is adopted.
2.4 Techniques for Term Weighting

On the other hand, the macro-evaluation approach to averaging can be independent of any parameter such as co-ordination level. The average curve is obtained by specifying a set of standard recall values for which average precision values are calculated by averaging over all queries, the individual precision values corresponding to the standard recall values. Often no unique precision value corresponds exactly so it becomes necessary to interpolate.

One measure that deserves special mention is average precision, a single valued measure most commonly used by the IR research community to evaluate ranked retrieval. Average precision is computed by measuring precision at different recall points (say 10%, 20%, and so on) and averaging. Both recall and precision are set oriented measures and have no notion of ranked retrieval. Researchers have used several variants and combinations of recall and precision to evaluate ranked retrieval.

Composite measures

Dissatisfaction in the past with methods of measuring effectiveness by a pair of numbers (just such as precision and recall) which may co-vary in a loosely specified way has led to attempts to invest composite measures. These are still based on the ‘contingency’ table but combine parts of it into a single number measure. Unfortunately, many of these measures are rather ad hoc and cannot be justified in any rational way. The simplest example of this kind of measure is the sum of precision and recall $S = P + R$. This is simply related to a measure suggested by Borko $BK = P + R - 1$.

2.4 Techniques for Term Weighting

The most critical piece of information needed for document ranking in all models is a terms weight in a document. A large body of work has gone into proper estimation of these weights in different models. Another technique that has been shown to be effective in improving document ranking is query modification via relevance feedback. A state-of-the-art ranking system uses an effective weighting scheme in combination with a good query expansion technique.

Various methods for weighting terms have been developed in the field. Weighting methods developed under the probabilistic models rely heavily upon better estimation of various probabilities. Methods developed under the vector space model are often
2. INFORMATION RETRIEVAL

based on researchers experience with systems and large scale experimentation (43). In both models, three main factors come into play in the final term weight formulation (30).

- **Term Frequency (tf):** Words that repeat multiple times in a document are considered salient. Term weights based on tf have been used in the vector space model since the 1960s.

- **Document Frequency (df):** Words that appear in many documents are considered common and are not very indicative of document content. A weighting method based on this, called inverse document frequency (idf) weighting, was proposed by Sparck-Jones early in 1970s (44).

- **Document Length (dl):** When collections have documents of varying lengths, longer documents tend to score higher since they contain more words and word repetitions. This effect is usually compensated by normalizing for document lengths in the term weighting method. Before TREC\(^1\), both the vector space model and the probabilistic models developed term weighting schemes which were shown to be effective on the small test collections available then. Inception of TREC provided IR researchers with very large and varied test collections allowing rapid development of effective weighting schemes.

Soon after first TREC, researchers at Cornell University realized that using raw tf of terms is non-optimal, and a dampened frequency (e.g., a logarithmic tf function) is a better weighting metric (45). In subsequent years, an effective term weighting scheme was developed under a probabilistic model by Steve Robertson and his team at City University, London (46). Motivated in part by Robertsons work, researchers at Cornell University developed better models of how document length should be factored into term weights (47). At the end of this rapid advancement in term weighting, the field had two widely used weighting methods, one (often called Okapi weighting) from Robertsons work, and the second (often called pivoted normalization weighting) from

\(^1\)The Text REtrieval Conference (TREC, [http://trec.nist.gov/](http://trec.nist.gov/)), co-sponsored by the National Institute of Standards and Technology (NIST) and U.S. Department of Defense, was started in 1992 as part of the TIPSTER Text program. Its purpose was to support research within the information retrieval community by providing the infrastructure necessary for large-scale evaluation of text retrieval methodologies.
the work done at Cornell University. Most research groups at TREC currently use some variant of these two weightings.

Many studies have used the phrase $tf \times idf$ weighting to refer to any term weighting method that uses $tf$ and $idf$, and do not differentiate between using a simple document scoring method (like $\sum_{t \in Q, D} tf \cdot ln \frac{N}{df}$) and a state-of-the-art scoring method. Such studies claim that their proposed methods are far superior than $tf \times idf$ weighting, often a wrong conclusion based on the poor weighting formulation used.

One popular class of statistical term weighting functions is $tf \times idf$, where two intuitions are at play: (a) the more frequently $t_k$ occurs in $d_j$, the more important for $d_j$ it is (the term frequency); (b) the more documents $t_k$ occurs in, the less discriminating it is, i.e., the smaller its contribution is in characterizing the semantics of a document in which occurs (the inverse document frequency). Most of the times, the standard $tf \times idf$ function is defined as

$$tfidf(t_k, d_j) = \#(t_k, d_j) \cdot \frac{|T_r|}{\#T_r(t_k)}.$$  (2.6)

Weights computed by $tf \times idf$ are often normalized so as to contrast the tendency of such expression to emphasize long documents. One of the most famous variations to such approach is the Okapi BM25. It is based on the probabilistic retrieval framework developed in the 1970s and 1980s by Stephen E. Robertson, Karen Spärck Jones, and others. BM25, and its newer variants, e.g., BM25F (a version of BM25 that can take document structure and anchor text into account).

### 2.5 Multi Agent Systems for Information Retrieval

IR is one of the application fields in which Multi Agent Systems (MAS) (i.e., a pool of information agents) have been successfully applied. An information agent is an agent that has access to one or more information sources, and is able to store and process information obtained from these sources in order to answer queries posed by users and other information agents. The information sources may be of many types, including web services, web sites, RSS-feeds, and traditional databases.

IR is undoubtedly a really complex domain and agent-based computing is a promising approach for developing applications in complex domain. However, despite the great deal of research during the years, a number of challenges still need to be faced.
make agent-based computing a widely accepted paradigm in software engineering practice, and (ii) to turn agent-oriented software abstractions into practical tools for facing the complexities of modern application areas. In this section, I will shortly introduce the key concepts of agent-based computing (as they pertain to software engineering), and characterize the most important realities of existing MultiAgent Systems.

2.5.1 Agent Oriented Software Engineering

Agents and multiagent systems have emerged in the ’90 as a powerful technology to face the complexity of a variety of today’s ICT scenarios. For instance, several industrial experiences already testify the advantages of using agents in manufacturing processes (50; 51), Web services and Web-based computational markets (52), and distributed network management (53). In addition, several studies advise on the possibility of exploiting agents and multiagent systems as enabling technologies for a variety of present scenarios, i.e., pervasive computing (54; 55), Grid computing (56), Semantic Web (57). However, the emergent general understanding is that multiagent systems, more than an effective technology, represent indeed a general-purpose paradigm for software development (58; 59). Agent-based computing promotes designing and developing applications in terms of autonomous software entities (agents), situated in an environment, and that can flexibly achieve their goals by interacting with one another in terms of high-level protocols and languages.

These features are well suited to tackle the complexity of developing software in modern scenarios: (i) the autonomy of application components reflects the intrinsically decentralized nature of modern distributed systems (55) and can be considered as the natural extension to the notions of modularity and encapsulation for systems that are owned by different stakeholders (60); (ii) the flexible way in which agents operate and interact (both with each other and with the environment) is suited to the dynamic and unpredictable scenarios in which software is expected to operate (61); (iii) the concept of agency provides for an unified view of the artificial intelligence results and achievements, by making agents and multiagent systems act as sound and manageable repositories of intelligent behaviors (62). In the last twenty years, together with the increasing acceptance of agent-based computing as a novel software engineering paradigm, there has been a great deal of research related to the identification and definition of suitable models and techniques to support the development of complex software systems in
2.5 Multi Agent Systems for Information Retrieval

terms of multiagent systems \(63\). These researches, which can be roughly grouped under the term “agent-oriented software engineering” \(59\; 64\), are endlessly proposing a variety of new metaphors, formal modeling approaches, development methodologies and modeling techniques, specifically suited to the agent-oriented paradigm.

2.5.2 MAS for Information Retrieval

As already mentioned, due to the increased availability of documents in digital form and the consequential need to access them in a flexible way, automated content-based document management tasks have gained a main role in the information systems field \(65\). In particular, web information retrieval is highly popular and presents a technical challenge due to the heterogeneity and size of the web, which is continuously growing (see \(66\), for a survey).

Currently, the most overshadowing and noteworthy web information sources are developed according to the collaborative-web paradigm \(67\), also known as Web 2.0. It represents a paradigm shift in the way users approach the web. Users (also called prosumers) are no longer passive consumers of published content, but become involved, implicitly and explicitly, as they cooperate by providing their own content in an “architecture of participation” \(68\).

Let me briefly recall that an agent architecture is essentially a map of the internals of an agent - i.e., its data structures, the operations that may be performed on them, and the corresponding control flows \(69\).

In the literature, several centralized agent-based architectures aimed at performing information retrieval tasks have been proposed. Among others, let me recall NewT \(70\), Letizia \(71\), WebWatcher \(72\), and SoftBots \(73\).

NewT \(70\) has been designed as a society of information-filtering interface agents, which learn user preferences and act on her/his behalf. These information agents use a keyword-based filtering algorithm, whereas adaptive techniques are relevance feedback and genetic algorithms.

Letizia \(71\) is an intelligent user interface agent able to assist a user while browsing the Web. The search for information results as a cooperative venture between the user and the software agent: both browse the same search space of linked web documents, looking for interesting ones.
2. INFORMATION RETRIEVAL

WebWatcher (72) is an information search agent that follows web hyperlinks according to user interests, returning a list of links deemed interesting.

In contrast to systems for assisted browsing or information retrieval, SoftBots (73) accept high-level user goals and dynamically synthesize the appropriate sequence of Internet commands according to a suitable ad-hoc language.

Despite the fact that a centralized approach could have some advantages, in information retrieval tasks it may encompass several problems, in particular how to scale up the architectures to large numbers of users, how to provide high availability in case of constant demand of the involved services, and how to provide high trustability in case of sensitive information, such as personal data.

To this end suitable multiagent systems devoted to perform information retrieval tasks have been proposed. In particular, Sycara et al. (74) proposed Retsina, a multiagent system infrastructure applied in many domains. Retsina is an open MAS infrastructure that supports communities of heterogeneous agents. Three types of agents have been defined: interface agents, able to display the information to the users; task agents, able to assist the user in the management of her/his information; and information agents, able to gather relevant information from the selected sources.

Apart from Retsina, in the literature, several multiagent systems have been proposed and implemented. Among others, let me recall IR agents (75), CEMAS (76), and the cooperative multiagent system for web information retrieval proposed in (77).

IR agents (75) implement an XML-based multiagents model for information retrieval. The corresponding framework is composed of three kinds of agents: (i) managing agents, aimed at extracting the semantics of information and at performing the actual tasks imposed by coordinator agents, (ii) interface agents, devised to interact with the users, and (iii) search agents, aimed at discovering relevant information on the web. IR agents do not take into account personalization, while providing information in a structured form without the adoption of specific classification mechanisms.

In CEMAS (Concept Exchanging Multi-Agent System) (70) the basic idea is to provide specialized agents for each main task, the main tasks being: (i) exchanging concepts and links, (ii) representing the user, (iii) searching for new relevant documents matching existing concepts, and (iv) supporting agent coordination. Although CEMAS provides personalization and classification mechanisms based on a semantic approach,
the main drawback is that it is not generic, being mainly aimed at supporting scientists while looking for comprehensive information about their topic area.

Finally, in [77] the underlying idea is to adopt intelligent agents that mimic everyday-life activities of information seekers. To this end, agents are also able to profile the user in order to anticipate and achieve her/his preferred goals. Although the approach is quite interesting, it is mainly focused on cooperation among agents rather than on information retrieval issues.

Table 2.3 summarizes all these systems putting into evidence both architectural and methodological/algorithmic aspects. In particular, for each system taken into account, the involved agents and the adopted methodology have been shown.

In this thesis work, I am also going to present a generic architecture, designed to support the implementation of applications aimed at managing information among different and heterogeneous sources. Such architecture is called X.MAS (see chapter 6) and has been explicitly created to filter, classify, and organize information. The overall architecture is a support for implementing Personalized, Adaptive, and Cooperative MultiAgent Systems devoted to Information Retrieval Tasks thanks to a pool of autonomous and flexible agents that can be made personal, adaptive and cooperative, depending on the given application.
Table 2.3: Agent-based information retrieval systems at a glance.

<table>
<thead>
<tr>
<th>System</th>
<th>Agent Type</th>
<th>Knowledge Base</th>
</tr>
</thead>
<tbody>
<tr>
<td>NewT</td>
<td>Information Agents</td>
<td>Keyword Based</td>
</tr>
<tr>
<td></td>
<td>Filter Agents</td>
<td>Relevance Feedback</td>
</tr>
<tr>
<td></td>
<td>Interface Agents</td>
<td>Genetic Algorithms</td>
</tr>
<tr>
<td>Letizia</td>
<td>Behavior Based Interface Agent</td>
<td>Query Based</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Relevance Feedback</td>
</tr>
<tr>
<td>WebWatcher</td>
<td>Information search agent</td>
<td>Utility Function Based</td>
</tr>
<tr>
<td>SoftBots</td>
<td>Goal-oriented agents</td>
<td>UCPOP-based Planning</td>
</tr>
<tr>
<td>Retsina</td>
<td>Interface Agents</td>
<td>HTN Planner</td>
</tr>
<tr>
<td></td>
<td>Task Agents</td>
<td>Information Gathering</td>
</tr>
<tr>
<td></td>
<td>Information Agents</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Middle Agents</td>
<td></td>
</tr>
<tr>
<td>CEMAS</td>
<td>Task agents</td>
<td>Semantic Approach</td>
</tr>
<tr>
<td>IR</td>
<td>Interface agents</td>
<td>Semantic Approach</td>
</tr>
<tr>
<td></td>
<td>Search agents</td>
<td></td>
</tr>
<tr>
<td>Team Consensus</td>
<td>User Profiling Agents</td>
<td>Team Consensus</td>
</tr>
</tbody>
</table>
Chapter 3

Text Categorization

Text Categorization (also known as text/document classification) is the task of assigning predefined categories to text documents. It can provide conceptual views of document collections and has many important applications in the real world. Many document collections are useful to be categorized into classes: news stories, are typically organized by subject categories (topics) or geographical codes; academic papers are often classified by technical domains and sub-domains; patient reports in healthcare organizations are often indexed from multiple aspects, using taxonomies of disease categories, types of surgical procedures, insurance reimbursement codes; in spam filtering, email messages are classified into the two categories of spam and non-spam, respectively, and so on.

3.1 Introduction

With the rapid growth of online information, and because of the need for improving the retrieval of such information (as discussed in chapter 2) Text Categorization (TC) has become one of the key techniques for handling and organizing text data. The goal of TC is the classification of documents into a fixed number of predefined categories. Each document can be in multiple, exactly one, or no category at all. Using machine learning, the objective is to learn classifiers from examples which perform the category assignments automatically (158). This is a supervised learning problem. Since categories may overlap, each category is treated as a separate binary classification problem. The first step in TC is to transform documents, which typically are strings of characters,
3. TEXT CATEGORIZATION

into a representation suitable for the learning algorithm and the classification task. IR research suggests that word stems work well as representation units and that their ordering in a document is of minor importance for many tasks. This leads to an attribute-value representation of text. Each distinct word corresponds to a feature, often weighted by its statistical relevance in that context.

This representation scheme leads to very high-dimensional feature spaces containing 10000 dimensions and more. Many researchers have noted the need for feature selection to make the use of conventional learning methods possible, to improve generalization accuracy, and to avoid “overfitting”. Following the recommendation of (79), the information gain criterion (discussed in section 3.3.1) is an example of feature selection algorithm to select a subset of features. Finally, from IR it is known that scaling the dimensions of the feature vector with their inverse document frequency \( idf \) (43) improves performance (see section 2.4).

For machine learning researchers, the big interest about this field is due to the fact that IR applications prove an excellent and challenging benchmark for their own techniques and methodologies, since IR applications usually handle with extremely high-dimensional feature spaces and provide data by the truckload. In the last two decades, this has resulted in more and more machine learning researchers adopting TC as one of their benchmark applications of choice, which means that cutting-edge machine learning techniques are being imported into TC with minimal delay from their original invention (80). In particular, the automated categorization of texts into pre-
3.1 Introduction

Defined categories has witnessed a booming interest due to increased availability of documents in digital form and the ensuing need to organize them. In the research community, the dominant approach to this problem is based on machine learning techniques: a general inductive process automatically builds a classifier by learning, from a set of preclassified documents, the characteristics of the categories. The advantages of this approach over the knowledge engineering approach (consisting in the manual definition of a classifier by domain experts) are a very good effectiveness, considerable savings in terms of expert labor power, and straightforward portability to different domains.

For application developers, this interest is mainly due to the enormously increased need to handle larger and larger quantities of documents, a need emphasized by increased connectivity and availability of document bases of all types at all levels in the information chain. This interest is also due to the fact that TC techniques have reached accuracy levels that rival the performance of trained professionals, and these accuracy levels can be achieved with high levels of efficiency on standard hw/sw resources. This means that more and more organizations are automating all their activities. Moreover thanks to great classified corpus such as the well known Reuters (see (81)) or the TREC (see (82)) it is possible to test algorithms with consistent and reliable datasets most of the time inclusive of interesting features such as a taxonomy extracted by the hierarchy of classes that in the particular case of my research topic was fundamental.

TC is a case of supervised learning where the set of categories and examples of documents belonging to those categories are given. This research will not concern problem of unsupervised learning, called text clustering, where categories are not known in advance.

TC as a research area appeared in the 1960s. Only in the 1990s it became a major field in information science due to the increased interest in its diverse applications such as document indexing with controlled vocabulary, filtering of irrelevant information, web page categorization, email management, detection of text genre, and many others. Clearly, TC techniques are a necessity nowadays when most information is produced and stored digitally (see chapter 2). Business and personal correspondence, scientific and entertaining articles, conference proceedings, patient data, are just a few examples of electronic text collections. With the advent of World Wide Web (WWW) another massive repository of text information was created. These huge text data
3. TEXT CATEGORIZATION

Demand automatic means of efficient and effective storage and retrieval that can be provided by means of TC [83].

3.2 A formal definition of Text Categorization

Before introducing the main contributions in the field of TC, let me give some formal definitions from [65].

**Definition (Text Categorization).** Text categorization is the task of assigning a boolean value to each pair \( \langle d_j; c_i \rangle \in D \times C \), where \( D \) is a domain of documents and \( C = \{ c_1; \ldots; c_C \} \) is a set of predefined categories.

A value of \( T \) assigned to \( \langle d_j; c_i \rangle \) indicates a decision to file \( d_j \) under \( c_i \), while a value of \( F \) indicates a decision not to file \( d_j \) under \( c_i \). More formally, the task is to approximate the unknown function \( \hat{\Phi} : D \times C \rightarrow \{T, F\} \) (that describes how documents ought to be classified) by means of a function \( \Phi : D \times C \rightarrow \{T, F\} \) called the classifier (also known as rule, or hypothesis, or model) such that \( \hat{\Phi} \) and \( \Phi \) “coincide as much as possible”.

Fundamental to the understanding of this task are two observations:

- the categories are just symbolic labels. No additional knowledge (either of a procedural or of a declarative nature) of their “meaning” is assumed available to help in the process of building the classifier. In particular, this means that the “text” constituting the category label (e.g., Sports in a news categorization task) is not to be used.

- the attribution of documents to categories should, in general, be realized on the basis of the semantics of the documents, and not on the basis of metadata (e.g., publication date, document type, publication source, etc.). That is, the categorization of a document should be based solely on endogenous knowledge (i.e., knowledge that can be extracted from the document itself) rather than on exogenous knowledge (i.e., data that might be provided for this purpose by an external source).

Given that the semantics of a document is an inherently subjective notion, it follows that the fundamental notion of TC, that of relevance of a document to a category, cannot be decided deterministically. This is exemplified by the well-known phenomenon
of inter-indexer inconsistency \cite{84}. TC is a subjective task \cite{80}: when two different humans must take a decision on whether to classify document $d_j$ under category $c_i$, they may disagree, and this in fact happens with relatively high frequency. Also a news article could be filed under one or another category, or under both, or even under neither, depending on the subjective judgment of the classifier. The above-mentioned notion of relevance of a document to a category basically coincides with the notion of relevance of a document to an information need, as from IR \cite{85}.

### 3.2.1 Single-Label versus Multi-Label Text Categorization

Generally it is impossible to categorize each document under a single label, because of the natural overlapping of the category spaces. As an example, the economics field often dovetails the politic one. This fact brings to enforce different constraints on the categorization task, depending on the application. For instance, we might want that under these conditions also, for a given integer $k$, each element of $C$ must be assigned to exactly $k$ (or $\leq k$, or $\geq k$) elements of $D$. For instance, this happens when we want categories to be evenly populated, or when we want them to be populated each to a certain degree. More importantly, we might want that for a given integer $k$, exactly $k$ (or $\leq k$, or $\geq k$) elements of $C$ must be assigned to each element of $D$. The case $k = 1$, as e.g., in \cite{86, 87, 88, 89, 90}, is often called the single-label case (or the non-overlapping categories case), whereas the general case in which any number of categories from 0 to $m$ may be assigned to the same document is dubbed the multi-label case \cite{91, 92, 93, 94}.

In this thesis work, I am particularly interested in the case in which each document can be labeled with a degree of relevance to one or more categories taken from the available pool. From a theoretical point of view, the single-label case is more general than the multi-label case, in the sense that an algorithm for single-label classification can also be used for multi-label classification by simply transforming a problem of multi-label classification with categories \{$c_1, \cdots, c_m$\} into $m$ independent problems of single-label classification with categories \{$c_i, \bar{c}_i$\}, for $i = 1, \cdots, m$. This requires, however, that categories are stochastically independent of each other, i.e., $f(d_j, c')$ does not depend on $f(d_j, c'')$ and vice versa, which is usually assumed to be the case (exceptions to this rule will be dealt with in). The converse is not true in general: if we have an algorithm for performing multi-label classification, it is not always the case that we can use it for single-label classification too. In fact, (i) it might not be obvious
3. TEXT CATEGORIZATION

how to choose a single “best” category among the \( k \) categories that the classifier has attached to the document, or (ii) for some documents \( k \) might be equal to 0.

Thus, I will assume from now on to be concerned with the single-label case, given its greater generality. This means that I will view the classification problem for the \( D \times C \) decision matrix as consisting of \( m \) independent problems of classifying the documents in \( D \) under a given category \( c_i \), for \( i = 1, \cdots, m \). A classifier for \( c_i \) is then a function \( \tilde{f}_i : D \in \{0, 1\} \) that approximates an unknown function \( f_i : D \in \{0, 1\} \).

3.2.2 Hard Categorization versus Ranking Categorization

While a complete automation of the TC task requires a True or False decision for each pair \( \langle d_j, c_i \rangle \), a partial automation of this process might have different requirements. For instance, given \( d_j \in D \) a system might simply rank the categories in \( C = \{c_1, \cdots, c_{|C|}\} \) according to their estimated appropriateness to \( d_j \), without taking any hard decision on any of them. Such a ranked list would be of great help to a human expert in charge of taking the final decision, since she/he could thus restrict the choice to the category (or categories) at the top of the list, rather than having to examine the entire set. Alternatively, given \( c_i \in C \) a system might simply rank the documents in \( D \) according to their estimated appropriateness to \( c_i \); symmetrically, for classification under \( c_i \) a human expert would just examine the top-ranked documents instead of the entire document set. These two modalities are sometimes called category-ranking TC and document-ranking TC \((95)\), respectively, and are the counterparts of DPC (document pivoted categorization) a given document is to be assigned category label(s) and CPC (category pivoted categorization) in which all documents that belong to a given category must be identified.

Semiautomated “interactive” classification systems \((96)\) are useful especially in critical applications in which the effectiveness of a fully automated system may be expected to be significantly lower than that of a human expert. This may be the case in which the quality of the training data is low, or when the training documents cannot be trusted to be a representative sample of the unseen documents that are to come, so that the results of a completely automatic classifier could not be trusted completely. In my work, unless explicitly mentioned, I deal with “hard” classification; however, many of the algorithms I will discuss in this chapter naturally lend themselves to ranking TC too.
3.3 How to represent documents, classes, and corresponding classifiers

Text documents, as they are represented in natural language, are not amenable to being interpreted by a classifier or by a classifier-building algorithm. Because of this, an indexing procedure that maps a text \( d \) into a schematic representation of its content needs to be invoked to represent such kind of knowledge.

3.3.1 Document Indexing

The activity of document indexing consists of mapping a document \( d_j \) into a compact representation of its content that can be directly interpreted (i) by a classifier building algorithm, and (ii) by a classifier, once it has been built \(^{(80)}\). The choice of a representation for text depends on what one regards as the meaningful textual units (the problem of lexical semantics) and the meaningful natural language rules for the combination of these units (the problem of compositional semantics). In true IR style, each document is usually represented by a vector of \( n \) weighted index terms (hereafter, simply terms) that occur in the document. Thus, formally the document is represented as a vector of term weights \( \vec{d}_j = \langle w_{1j}, ..., w_{\|\tau\|j} \rangle \), where \( \tau \) is the dictionary, i.e., the set of terms (also known as features), and \( 0 \leq w_{kj} \leq 1 \) quantifies the importance of \( t_k \) in characterizing the semantics of \( d_j \). Typical values of \( k \) are between 1 and 5.

Differences among the various approaches are accounted for by

1. different ways to understand what a term is;
2. different ways to weight terms.

A typical choice for issue (1) is to identify terms with all the words occurring in the document. This is often referred to as the bag of words approach to document representation. In a number of experiments \(^{(21, 97, 98)}\) it has been found that representations more sophisticated than this yields worse categorization effectiveness, thereby confirming similar results from IR \(^{(43)}\). In particular, a number of authors have tried to use noun phrases, rather than individual words, as indexing terms, but the experimental results found to date have not been encouraging.
3. TEXT CATEGORIZATION

The bag of word technique, that is the most frequent choice (65) consists on identifying terms either with the words occurring in the document (with the exception of stop words, i.e., topic-neutral words such as articles and prepositions, which are eliminated in a pre-processing phase), in particular their stems (i.e., their morphological roots), obtained by applying a stemming algorithm (99). A popular choice is to add to the set of word or stems a set of phrases, i.e., longer (and semantically more significant) language units extracted from the text by shallow parsing and/or statistical techniques (100). An example of a stop words set is given in table 5.1 and an example of words with the corresponding stems is given in table 5.2.

The method of compute weights may be binary-valued (i.e., \(w_{kj} \in \{0, 1\}\)) or (as most often it happens) real-valued (i.e., \(0 \leq w_{kj} \leq 1\)), depending on whether the classifier-building algorithm and the classifiers, once they have been built, require binary input or not. When weights are binary, these simply indicate presence/absence of the term in the document. Otherwise they are computed by either statistical or probabilistic techniques (see, e.g., (101)), the former being the most common option.

One popular class of statistical term weighting functions is \(tf \cdot idf\) and one of its most famous variations the Okapi BM25 (48) have been presented in section 2.4.

3.3.2 Dimensionality Reduction

In TC, unlike in IR, a Dimensionality Reduction (DR) phase is often applied so as to reduce the size of the document representations from \(T\) to a much smaller, predefined number. This has both the effect of reducing overfitting (i.e., the tendency of the classifier to better classify the data it has been trained on that new unseen data), and to make the problem more manageable for the learning method, since many such methods are known not to scale well to high problem sizes. DR often takes the form of features selection: each term is scored by means of a scoring function that captures its degree of (positive, and sometimes also negative) correlation with \(c_i\), and only the highest scoring terms are used for document representation. Alternatively, DR may take the form of feature extraction: a set of “artificial” terms is generated from the original term set in such a way that newly generated terms are both fewer and stochastically more independent from each other than the original ones used to be.

DR is also beneficial to reduce overfitting, that is, the phenomenon by which a classifier is tuned also to the contingent characteristics of the training data rather than
3.3 How to represent documents, classes, and corresponding classifiers

just the constitutive characteristics of the categories (65). Experiments have shown that, in order to avoid overfitting a number of training examples roughly proportional to the number of terms used is needed. Fuhr and Buckley (102) have suggested 50 – 100 as the number of training examples needed in TC tasks. This means that, if DR is performed, overfitting may be avoided even if a smaller amount of training examples is used. Obviously this process could remove potentially useful information on the meaning of the documents, then the reduction process must always be performed with care.

Different approaches can be distinguished by the nature of the resulting terms:

- DR by term selection: \( T' \) is a subset of \( T \)
- DR by term extraction: the terms in \( T' \) are not of the same type of the terms in \( T \) (e.g., if the terms in \( T \) are words, the terms in \( T' \) may not be words at all), but are obtained by combinations or transformations of the original ones.

In DR by term selection, the term selection (also called term space reduction - TSR), attempts to select the \( r \) terms that, when used for document indexing, yield the highest effectiveness. Some approach (i.e., (103)) generates a new term set by either adding or removing a term and testing a classifier trained on such term set on a validation set in order to choose the most effective terms. Since this approach is computationally too heavy, easier alternatives have been proposed such as the filtering approach that exploits some measures of the “importance” of the term for the TC task. A simple and effective example of this technique is the use of the document frequency \( \#T_r(t_k) \) of a term \( t_k \), that is, only the terms that occur in the highest number of documents are retained. Yang and Pedersen (79) have shown that with this approach it is possible to reduce the dimensionality up to a factor of 10 with no loss in effectiveness.

Other more sophisticated information-theoretic functions have been used in the literature, among them the DIA association factor (102), chi-square (40, 79, 100, 104, 105, 106), NGL coefficient (107, 108), mutual information (79, 96, 97, 108, 109, 110, 111, 112, 113), odds ratio (100, 108, 114), relevancy score (115), GSS coefficient (105), and Information Gain (10, 79, 98, 100, 110, 112, 114, 116) that will be better presented in section 5.1.

The mathematical definitions of these measures are well summarized in (65) and reported in table 3.1.
3. TEXT CATEGORIZATION

<table>
<thead>
<tr>
<th>Function</th>
<th>Denoted by</th>
<th>Mathematical form</th>
</tr>
</thead>
<tbody>
<tr>
<td>DIA association factor</td>
<td>$z(t_k, c_i)$</td>
<td>$P(c_i</td>
</tr>
<tr>
<td>Information gain</td>
<td>$IG(t_k, c_i)$</td>
<td>$\sum_{c \in [c_i,c]} \sum_{t \in [t_k,t]} P(t,c) \cdot \log \frac{P(t,c)}{P(t_k)P(c)}$</td>
</tr>
<tr>
<td>Mutual Information</td>
<td>$MI(t_k, c_i)$</td>
<td>$\log \frac{P(t_k,c)}{P(t_k)P(c)}$</td>
</tr>
<tr>
<td>Chi-square</td>
<td>$\chi^2(t_k, c_i)$</td>
<td>$</td>
</tr>
<tr>
<td>NGL Coefficient</td>
<td>$NGL(t_k, c_i)$</td>
<td>$\sqrt{</td>
</tr>
<tr>
<td>Relevancy Score</td>
<td>$RS(t_k, c_i)$</td>
<td>$\log \frac{P(t_k</td>
</tr>
<tr>
<td>Odds ratio</td>
<td>$OR(t_k, c_i)$</td>
<td>$P(t_k</td>
</tr>
<tr>
<td>GSS coefficient</td>
<td>$GSS(t_k, c_i)$</td>
<td>$P(t_k</td>
</tr>
</tbody>
</table>

Table 3.1: Main Functions used for Term Space Reduction purposes

All these functions try to capture the intuition that the best terms for $c_i$ are the ones distributed most differently in the sets of positive and negative examples of $c_i$, even the interpretations to this principle vary across different functions.

In (117) has been shown that with various classifiers and various initial corpora, sophisticated techniques such as Information Gain and Chi-square can reduce the dimensionality of the term space by a factor of 100 with no loss (or even with a small increase) of effectiveness.

The category of DR by term extraction attempts to generate, from the original set $T$ a set $T'$ of “synthetic” terms that maximize effectiveness. The rationale for using synthetic (rather than naturally occurring) terms is that, due to the pervasive problems of polysemy, homonymy, and synonymy, the original terms may not be optimal dimensions for document content representation. Methods for term extraction try to solve these problems by creating artificial terms that do not suffer from them. This can be done by either extracting the new terms from the old one or converting the original document into new representations based on the newly synthesized dimensions. Two well known term extraction methods experimented in TC, are the term clustering and latent semantic indexing.

Term clustering tries to group words with a high degree of pairwise semantic relatedness, so that the groups (or their centroids, or a representative of them) may be used instead of the terms as dimensions of the vector space. Term clustering is different from term selection, since the former tends to address terms synonymous (or
3.3 How to represent documents, classes, and corresponding classifiers

near-synonymous) with other terms, while the latter targets non-informative terms. (98) was the first to investigate the use of term clustering in TC.

The method he employed, called reciprocal nearest neighbor clustering, consists in creating clusters of two terms that are one the most similar to the other according to some measure of similarity. His results were inferior to those obtained by single-word indexing, possibly due to a disappointing performance by the clustering method: as (98) said (pag.48), “The relationships captured in the clusters are mostly accidental, rather than the systematic relationships that were hoped for”. Li and Jain (111) viewed semantic relatedness between words in terms of their co-occurrence and co-absence within training documents. By using this technique in the context of a hierarchical clustering algorithm, they witnessed only a marginal effectiveness improvement.

Both Lewis and Li and Jain provided examples of unsupervised clustering, since clustering is not affected by the category labels attached to the documents. Baker and McCallum (86) provided instead an example of supervised clustering, as the distributional clustering method they employed clusters together those terms that tend to indicate the presence of the same category, or group of categories. Their experiments, carried out in the context of a Naive Bayes classifier, showed only a 2% effectiveness loss with an aggressivity of 1,000, and even showed some effectiveness improvement with less aggressive levels of reduction. Later experiments by Slonim and Tishby (118) have confirmed the potential of supervised clustering methods for term extraction.

Latent semantic indexing (LSI, (119)) is a DR technique developed in IR in order to address the problems deriving from the use of synonymous, near synonymous, and polysemous words as dimensions of document representations. This technique compresses document vectors into vectors of a lower-dimensional space whose dimensions are obtained as combinations of the original dimensions by looking at their patterns of co-occurrence. In practice, LSI infers the dependence among the original terms from a corpus and “wires” this dependence into the newly obtained, independent dimensions. The function mapping original vectors into new vectors is obtained by applying a singular value decomposition to the matrix formed by the original document vectors. In TC this technique is applied by deriving the mapping function from the training set and then applying it to train and test documents alike. One characteristic of LSI is that the newly obtained dimensions are not, unlike in term selection and term clustering, intuitively interpretable. However, they work well in bringing out the “latent”
3. TEXT CATEGORIZATION

semantic structure of the vocabulary used in the corpus. Wiener et al. [115] used LSI in two alternative ways: (i) for local DR, thus creating several category-specific LSI representations, and (ii) for global DR, thus creating a single LSI representation for the entire category set. Their experiments showed that the former approach performs better than the latter, and both approaches perform better than simple TSR based on Relevancy Score. Schütze et al. [106] experimentally compared LSI-based term extraction with $\chi^2$-based TSR using three different classifier learning techniques (namely, linear discriminant analysis, logistic regression, and neural networks). Their experiments showed LSI to be far more effective than $\chi^2$ for the first two techniques, while both methods performed equally well for the neural network classifier. For other TC works that have used LSI or similar term extraction techniques, see Hull [1994], Li and Jain [1998] Schütze [120], Weigend et al. [121], and Yang [122].

3.3.3 Classifier learning

A text classifier for $c_i \in C$ is automatically generated by a general inductive process (the learner) which, by observing the characteristics of a set of documents pre-classified under $c_i$ or $\bar{c}_i$, gleans the characteristics that a new unseen document should have in order to belong to $c_i$. In order to build classifiers for $C$, one thus needs a set $\Omega$ of documents such that the value of $\Phi(d_j, c_i)$ is known for every $\langle d_j, c_i \rangle \in \Omega \times C$. In experimental TC it is customary to partition $\Omega$ into three disjoint sets $T_r$ (the training set), $V_a$ (the validation set), and $T_e$ (the test set). The training set is the set of documents observing which the learner builds the classifier. The validation set is the set of documents on which the engineer fine-tunes the classifier, e.g., choosing for a parameter $p$ on which the classifier depends, the value that has yielded the best effectiveness when evaluated on $V_a$. The test set is the set on which the effectiveness of the classifier is finally evaluated. In both the validation and test phase, “evaluating the effectiveness” means running the classifier on a set of preclassified documents ($V_a$ or $T_e$) and checking the degree of correspondence between the output of the classifier and the preassigned classes. Different learners have been applied in the TC literature. Some of these methods generate binary-valued classifiers of the required form $\hat{\Phi} : D \times C \rightarrow \{T, F\}$, but some others generate real-valued functions of the form $CSV : D \times C \rightarrow [0, 1]$ (CSV standing for categorization status value). For these latter, a set of thresholds $\tau_i$ needs to be determined (typically, by experimentation on a validation set) allowing to
3.3 How to represent documents, classes, and corresponding classifiers

turn real-valued CSVs into the final binary decisions \( \text{[123]} \). It is worthwhile to notice that in several applications, the fact that a method implements a real-valued function can be profitably used, in which case determining thresholds is not needed. For instance, in applications in which the quality of the classification is of critical importance (e.g., in filing patents into patent directories), post-editing of the classifier output by a human professional is often necessary. In this case, having the documents ranked in terms of their estimated relevance to the category may be useful, since the human editor can scan the ranked list starting from the documents deemed most appropriate for the category, and stop when desired.

Training efficiency (i.e., average time required to build a classifier \( \Phi_i \) from a given corpus \( \Omega \)), as well as classification efficiency (i.e., average time required to classify a document by means of \( \Phi_i \)), and effectiveness (i.e., average correctness of \( \Phi_i \)'s classification behavior) are all legitimate measures of success for a learner. In TC research, effectiveness is usually considered the most important criterion, since it is the most reliable one when it comes to experimentally comparing different learners or different TC methodologies, given that efficiency depends on too volatile parameters (e.g., different sw/hw platforms). In TC applications, however, all three parameters are important, and one must carefully look for a trade off among them, depending on the application constraints. For instance, in applications involving interaction with the user, a classifier with low classification efficiency is unsuitable. On the contrary, in multi-label TC applications involving thousands of categories, a classifier with low training efficiency also might be inappropriate (since many thousands of classifiers need to be learnt). Anyway, effectiveness tends to be the primary criterion in operational contexts too, since in most applications an ineffective, although efficient, classifier will be hardly useful, or will involve too much post-editing work on the part of human professionals, which might defy the purpose of using an automated system. In single-label TC, effectiveness is usually measured by accuracy, i.e., the percentage of correct classification decisions (error is the converse of accuracy, i.e., \( E = 1 - A \)). However, in binary (henceforth: in multi-label) TC, accuracy is not an adequate measure (see \( \text{[2.3.2]} \)). The reason for this is that in binary TC applications the two categories \( c_i \) and \( \bar{c}_i \) are usually unbalanced, i.e., one contains far more members than the other (see \( \text{[5.1.3]} \)). In this case, building a classifier that has high accuracy is trivial, since the trivial rejector, i.e., the classifier that trivially assigns all documents to the most heavily populated category (i.e., \( c_i \)),
has indeed very high accuracy; and there are no applications in which one is interested in such a classifier. As a result, in binary TC it is often the case that effectiveness \textit{wrt} category $c_i$ is measured by a combination of precision \textit{wrt} $c_i(\pi_i)$, the percentage of documents deemed to belong to $c_i$ that in fact belong to it, and recall \textit{wrt} $c_i(\rho_i)$, the percentage of documents belonging to $c_i$ that are in fact deemed to belong to it, as already discussed in section 2.3.1.

In multi-label TC being this an IR field, when effectiveness is computed for several categories the precision and recall results for individual categories must be averaged in some way; here, one may opt for micro-averaging (“categories count proportionally to the number of their positive training examples”) or for macro-averaging (“all categories count the same”), depending on the desired application (see equations 2.4, 2.5 in the section 2.3.2).

The former rewards classifiers that behave well on heavily populated (“frequent”) categories, while classifiers that perform well also on infrequent categories are emphasized by the latter. It is often the case that in TC research macro-averaging is the method of choice, since producing classifiers that perform well also on infrequent categories is the most challenging problem of TC. Since most classifiers can be arbitrarily tuned to emphasize recall at the expense of precision (and vice versa), only combinations of the two are significant.

The most popular way to combine the two is the function $F_\beta$ depicted in the section 2.3.2.

Finally, it should be noted that some applications of TC require cost-based issues to be brought to bear on how effectiveness is computed, thus inducing a utility-theoretic notion of effectiveness. For instance, in spam filtering (i.e., a binary TC task in which e-mail messages must be classified in the category \texttt{Spam} or its complement \texttt{NonSpam}), precision is more important than recall, since filing a legitimate message under \texttt{Spam} is a more serious error (i.e., it bears more cost) than filing a junk message under \texttt{NonSpam}. One possible way of taking this into account is using the $F_\beta$ measure with $\beta \neq 1$; using values of $0 \leq \beta < 1$ corresponds to paying more attention to precision than to recall, while by using values of $1 \leq \beta < \infty$ one emphasizes recall at the expense of precision.
3.3 How to represent documents, classes, and corresponding classifiers

3.3.4 Algorithms for Classifier Learning

The methods that have shown the best performance in comparative TC experiments so far are Support Vector Machine and Boosting [80]. They are also the newest methods in the classifier learning arena, and the ones with the strongest justification from computational learning theory. Another interesting algorithm belonging to the family of the instance-based learning methods is the k-nearest neighbors (k-NN) algorithm. Its most relevant features are the absence of training and its robustness to noise, but it also shown some strong consistency results [124]: as the amount of data approaches infinity, the algorithm is guaranteed to yield an error rate no worse than twice the Bayes error rate (the minimum achievable error rate given the distribution of the data).

Figure 3.2: SVM Algorithm, decision surfaces - The surfaces that separates the positive from the negative training examples in the SVM algorithms.

The Support Vector Machine (SVM) method has been introduced in TC by Joachims [78, 125] and subsequently used in several other TC works [97, 126, 127]. In geometrical terms, it may be seen as the attempt to find, among all the surfaces $\sigma_1, \sigma_2, \cdots$, in
3. TEXT CATEGORIZATION

$|T|$-dimensional space that separate the positive from the negative training examples (decision surfaces) (depicted in figure 3.2), the $\sigma_i$ that separates the positives from the negatives by the widest possible margin, i.e., such that the minimal distance between the hyper-plane and a training example is maximum; results in computational learning theory indicate that this tends to minimize the generalization error, i.e., the error of the resulting classifier on yet unseen examples. SVMs were usually conceived for binary classification problems \footnote{SVMLight is available at http://svmlight.joachims.org}, and only subsequently they have been adapted to multi-class classification \footnote{BSVM is available from http://www.csie.ntu.edu.tw/~cjlin/bsvm/}. As argued by Joachims \footnote{SVMLight is available at http://svmlight.joachims.org}, one advantage that SVMs offer for TC is that DR is usually not needed, as SVMs tend to be fairly robust to overfitting and can scale up to considerable dimensionalities. Recent extensive experiments by Brank and colleagues \footnote{BSVM is available from http://www.csie.ntu.edu.tw/~cjlin/bsvm/} also indicate that feature selection tends to be detrimental to the performance of SVMs.

Recently, efficient algorithms for SVM learning have been studied; as a consequence, the use of SVMs for high-dimensional problems such as TC is no more prohibitive from the point of view of computational cost. There are currently several freely available packages for SVM learning. The best known in the binary TC camp is the SVMLIGHT package \footnote{SVMLight is available at http://svmlight.joachims.org}, while one that has been extended to also deal with the general single-label classification problem is BSVM\footnote{BSVM is available from http://www.csie.ntu.edu.tw/~cjlin/bsvm/}.

Boosting algorithms build a collection of models using a “weak learner” and thereby reduces misclassification error, bias, and variance \footnote{Boosting algorithms build a collection of models using a “weak learner” and thereby reduces misclassification error, bias, and variance}. The algorithm is quite simple, beginning by building an initial model from the training dataset. Those entities in the training data which the model was unable to capture (i.e., the model mis-classifies those entities) have their weights boosted. A new model is then built with these boosted entities, which we might think of as the problematic entities in the training dataset. This model building followed by boosting is repeated until the specific generated model performs no better than random. The result is then a panel of models used to make a decision on new data by combining the “expertise” of each model in such a way that the more accurate experts carry more weight.

As a meta learner Boosting employs some other simple learning algorithm to build the models. The key is the use of a weak learning algorithm—essentially any weak learner can be used. A weak learning algorithm is one that is only somewhat better
than random guessing in terms of error rates (i.e., the error rate is just below 50%).
An example might be decision trees of depth 1 (i.e., decision stumps).

The Boosting meta-algorithm is an efficient, simple, and easy to program learning strategy. The popular variant called AdaBoost (an abbreviation for Adaptive Boosting) has been described as the “best off-the-shelf classifier in the world” ([132]). Boosting algorithms build multiple models from a dataset, using some other learning algorithm that need not be a particularly good learner. Boosting associates weights with entities in the dataset, and increases (boosts) the weights for those entities that are hard to accurately model. A sequence of models is constructed and after each model is constructed the weights are modified to give more weight to those entities that are harder to classify. In fact, the weights of such entities generally oscillate up and down from one model to the next. The final model is then an additive model constructed from the sequence of models, each model output weighted by some score. There is little tuning required and little is assumed about the learner used, except that it should be a weak learner. We note that boosting can fail to perform if there is insufficient data or if the weak models are overly complex. Boosting is also susceptible to noise.

Classifier committees (aka ensembles) are based on the idea that $k$ different classifiers $\Phi_1, \cdots, \Phi_k$ may be better than one if their individual judgments are appropriately combined. In the boosting method ([104] [133] [134] [135]) such classifiers are obtained by the same learning method (here called the weak learner), and are trained not in a conceptually parallel and independent way, but sequentially. In this way, in training classifier $\Phi_t$ one may take into account how classifiers $\Phi_1, \cdots, \Phi_{t-1}$ perform on the training examples, and concentrate on getting right those examples on which $\Phi_1, \cdots, \Phi_{t-1}$ have performed worst. Specifically, for learning classifier $\Phi_t$ each $\langle d_j, c_i \rangle$ pair is given an “importance weight” $h^t_{ij}$ (where $h^t_{ij}$ is set to be equal for all $\langle d_j, c_i \rangle$ pairs), which represents how hard to get a correct decision for this pair was for classifiers $\Phi_1, \cdots, \Phi_{t-1}$. These weights are exploited in learning $\Phi_t$, which will be specially tuned to correctly solve the pairs with higher weight. Classifier $\Phi_t$ is then applied to the training documents, and as a result weights $h^t_{ij}$ are updated to $h^{t+1}_{ij}$; in this update operation, pairs correctly classified by $\Phi_t$ will have their weight decreased, while pairs misclassified by $\Phi_t$ will have their weight increased. After all the $k$ classifiers have been built, a weighted linear combination rule is applied to yield the final committee. Boosting has proven
3. TEXT CATEGORIZATION

a powerful intuition, and the BOOSTEXTER system\textsuperscript{1} has reached one of the highest levels of effectiveness so far reported in the literature.

![Distribution of Prototypes for the k-NN Algorithm](image)

**Figure 3.3: The method of the nearest neighbor - A Distribution of Prototypes for the k-NN Algorithm**

The **Nearest Neighbor** method \textsuperscript{[95] [124] [136]} represents one of the simplest and most intuitive techniques in the field of statistical discrimination. It is a nonparametric method, where a new observation is placed into the class of the observation from the learning set that is closest to the new observation (see figure 3.3), with respect to the covariates used. The determination of this similarity is based on distance measures. Formally this simple fact can be described as follows: Let $L = \{(y_i, x_i), i = 1, \ldots, n_L\}$ be a training or learning set of observed data, where $y_i \in \{1, \ldots, c\}$ denotes class membership and the vector $x_i = (x_{i1}, \ldots, x_{ip})$ represents the predictor values. The determination of the nearest neighbors is based on an arbitrary distance function $d(\ldots)$. Then for a new observation $(y, x)$ the nearest neighbor $(y(1), x(1))$ within the learning set is determined by $d(x, x(1)) = \min(d(x, x_i))$ and $y = y(1)$, the class of the nearest neighbor, is selected as prediction for $y$. The notation $x(j)$ and $y(j)$ here describes the $j_{th}$ nearest neighbor of $x$ and its class membership, respectively. The most common distance employed are euclidean distance, or the absolute and the more general Minkowski distance (equation 3.1).

$$d(x_i, x_j) = \sqrt[q]{\sum_{s=1}^{p} |x_{is} - x_{js}|^q} \quad (3.1)$$

\textsuperscript{1}BooTexter is available from http://www.cs.princeton.edu/ schapire/boostexter.html
A first extension of this idea, that is widely and commonly used in practice, is the so-called k-nearest neighbor method ($kNN$). Here not only the closest observation within the learning set is referred for classification, but also the k most similar cases. The parameter $k$ has to be selected by the user. Then the decision is in favor of the class label, most of these neighbors belong to.

The weighted extension of $kNN$ ($wkNN$) is based on the idea that such observations within the learning set, which are particularly close to the new observation $(y,x)$, should get a higher weight in the decision than such neighbors that are far away from $(y,x)$. This is not the case with $kNN$: Indeed only the $k$ nearest neighbors influence the prediction; however, this influence is the same for each of these neighbors, although the individual similarity to $(y,x)$ might be widely different. To reach this aim, the distances, on which the search for the nearest neighbors is based in the first step, have to be transformed into similarity measures, which can be used as weights.

### 3.4 Applications of the Text Categorization

Automatic TC goes back at least to the early 1960s and to Maron’s seminal work. Since then, it has been used in a number of different applications. As depicted by Sebastiani the common traits among them are:

- The need to handle and organize documents in which the textual component is either unique, or dominant, or simplest to interpret, component.

- The need to handle and organize large quantities of such documents, i.e., large enough that their manual organization into classes is either too expensive or not feasible within the time constraints imposed by the application.

- The fact that the set of categories is known in advance, and its variation over time is small.

Applications may instead vary along several dimensions:

- The nature of the documents; i.e., documents may be structured texts (such as e.g., scientific articles), newswire stories, classified ads, image captions, e-mail messages, transcripts of spoken texts, hypertexts, or other. If the documents are hypertextual, rather than textual, very different techniques may be used,
since links provide a rich source of information on which classifier learning can leverage. Techniques exploiting this intuition in a TC context have been presented in (140; 141; 142; 143) and experimentally compared in (144).

- The structure of the classification scheme, i.e., whether this is flat or hierarchical. Hierarchical classification schemes may in turn be tree-shaped, or allow multiple inheritance. Again, the hierarchical structure of the classification scheme may allow radically more efficient, and also more effective, classification algorithms, which can take advantage of early subtree pruning (89; 127; 145), improved selection of negative examples (107), or improved estimation of word occurrence statistics in leaf nodes (146; 147; 148; 149).

- The nature of the task, i.e., whether the task is single-label or multi-label.

In the following, I briefly review the most important ones.

### 3.4.1 Automatic Indexing for Boolean Information Retrieval Systems

The first use to which automatic text classifiers were put at, and the application that spawned most of the early research in the field (139; 150; 151; 152; 153; 154), is that of automatic document indexing for use in IR systems relying on a controlled dictionary. As seen in section 2.2 the most prominent example of such IR systems is, of course, that of Boolean systems. In these systems, each document is assigned one or more keywords or keyphrases describing its content, where these keywords and keyphrases belong to a finite set of words called controlled dictionary and often consisting of a hierarchical thesaurus (e.g., the NASA thesaurus for the aerospace discipline, or the MESH thesaurus covering the medical field).

Usually, this assignment is performed by trained human indexers, and is thus an extremely costly activity. If the entries in the thesaurus are viewed as categories, document indexing becomes an instance of the document categorization task, and may thus be addressed by the automatic techniques described in this thesis. Note that in this case a typical constraint may be that \(k_1 \times k_2\) keywords are assigned to each document, for given \(k_1, k_2\). Document-pivoted categorization might typically be the best option, so that new documents may be classified as they become available.
3.4 Applications of the Text Categorization

Various automatic document classifiers explicitly addressed at document indexing applications have been described in the literature; see e.g., \cite{102,155,156,157,158,159}. The issue of automatic indexing with controlled dictionaries is closely related to the topic of automated metadata generation. In digital libraries we are usually interested to tag documents by metadata that describe them under a variety of aspects (e.g., creation date, document type or format, availability). Usually, some of these metadata are thematic, i.e., their role is to describe the semantics of the document by means of bibliographic codes, keywords or keyphrases. The generation of these metadata may thus be viewed as a problem of document indexing with controlled dictionary, and thus tackled by means of automatic TC techniques. An example system for automated metadata generation by TC techniques is the Klarity system\footnote{The Klarity system is available at http://www.topic.com.au/products/klarity.html}.

3.4.2 Document organization

In general, all issues pertaining to document organization and filing, be it for purposes of personal organization or document repository structuring, may be addressed by automatic categorization techniques. For instance, at the offices of a newspaper, incoming “classified” ads must be, prior to publication, categorized under the categories used in the categorization scheme adopted by the newspaper; typical categories might be e.g., Personals, Cars for Sale, Real Estate.

While most newspapers would handle this application manually, those dealing with a high daily number of classified ads might prefer an automatic categorization system to choose the most suitable category for a given ad. In this case, a typical constraint might be that exactly one category is assigned to each document. Similar applications might be the automatic filing of newspaper articles under the appropriate sections (e.g., Politics, Home News, Lifestyles), or the automatic grouping of conference papers into sessions. Document organization, both in the cases of paper documents and electronic documents, often has the purpose of making document search easier. An interesting example of this approach is the system for classifying and searching patents of the U.S. Patent and Trademark Office, described by Larkey \cite{90}. In this system documents describing patents are classified according to a hierarchical set of categories. Patent office personnel may thus search for existing patents related to a claimed new invention with greater ease.
3. TEXT CATEGORIZATION

3.4.3 Document Filtering

Document filtering refers to the activity of classifying a dynamic collection of documents, typically in the form of a stream of incoming documents dispatched in an asynchronous way by an information producer to an information consumer (160). A typical case of this is a news feed, whereby the information producer is a news agency (e.g., Reuters or Associated Press) and the information consumer is a newspaper. In this case, the filtering system should block the delivery to the consumer of the documents the consumer is not likely to be interested in (e.g., all news not concerning sports, in the case of a sports newspaper). Filtering can be seen as a case of single-label categorization, i.e., the categorization of incoming documents in two disjoint categories, the relevant and the irrelevant. Additionally, a filtering system may also perform a further categorization into topical categories of the documents deemed relevant to the consumer; in the example above, all articles about sports are deemed relevant, and should be further sub categorized according e.g., to which sport they deal with, so as to allow individual journalists specialized in individual sports to access only documents of high prospective interest for them.

Similarly, an e-mail filter might be trained to further classify previously filtered e-mail into topical categories of interest to the user (161). A document filtering system may be installed at the producer end, in which case its role is to route the information to the interested consumers only, or at the consumer end, in which case its role is to block the delivery of information deemed uninteresting to the user. In the former case the system has to build and update a “profile” for each consumer it serves (162), whereas in the latter case (which is the more common, and to which I will refer in the rest of this section) a single profile is needed. A profile may be initially specified by the user, thereby resembling a standing IR query, and is usually updated by the system by using feedback information provided by the user on the relevance or non-relevance of the delivered messages.

In the TREC community (163) this is called adaptive filtering, while the case in which no user-specified profile is available is called either routing or batch filtering, depending on whether documents have to be ranked in decreasing order of estimated relevance or just accepted/rejected.
3.4 Applications of the Text Categorization

In information science also document filtering has a tradition dating back to the 1960s, when, addressed by systems of varying degrees of automation and dealing with the multi-consumer case discussed above, it was variously called selective dissemination of information or current awareness (see e.g., [163]). The explosion in the availability of digital information, particularly on the Internet, has boosted the importance of such systems. These are nowadays being used in many different contexts, including the creation of personalized Web newspapers, “junk e-mail” blocking, and the selection of Usenet news. The construction of information filtering systems by means of machine learning techniques is widely discussed in the literature: see e.g., [106; 133; 165; 166; 167; 168; 169; 170; 171].

3.4.4 Word sense disambiguation

Word sense disambiguation (WSD) refers to the activity of finding, given the occurrence in a text of an ambiguous (i.e., polysemous or homonymous) word, the sense this particular word occurrence has. For instance, the English word “bank” may have (at least) two different senses, as in the Bank of England (a financial institution) or the bank of river Thames (a hydraulic engineering artifact). It is thus a WSD task to decide to which of the above senses the occurrence of bank in “Last week I borrowed some money from the bank” refers to. WSD is very important for a number of applications, including indexing documents by word senses rather than by words for IR or other content-based document management applications. WSD may be seen as a categorization task once we view word occurrence contexts as documents and word senses as categories. Quite obviously, this is a case in which exactly one category needs to be assigned to each document, and one in which document-pivoted categorization is most likely to be the right choice. WSD is viewed as a TC task in a number of different works in the literature; see e.g [172; 173]. WSD is just an example of the more general issue of resolving natural language ambiguities, one of the most important problems in computational linguistics. Other instances of this problem, which may all be tackled by means of TC techniques along the lines discussed for WSD, are context-sensitive spelling correction, prepositional phrase attachment, part of speech tagging, and word choice selection in machine translation. See the excellent [174] for an introduction to this field.
3. TEXT CATEGORIZATION
Chapter 4

Hierarchical Text Categorization

Most of the research on text categorization has focused on classifying text documents into a set of categories with no structural relationship among them (flat classification) (175). However, in many information repositories documents are organized in a hierarchy of categories to support a thematic search by browsing topics of interests.

Hierarchical text categorization deals with problems where categories are organized in form of a hierarchy. Furthermore, many information sources are organized as hierarchies, e.g., web repositories, digital libraries, patent libraries, email folders, product catalogs. In particular, several web repositories, such as DMOZ\textsuperscript{1}, Wikipedia\textsuperscript{2}, and Medical Subject Headings (MeSH) in MEDLINE\textsuperscript{3} encompass an underlying taxonomy. Taxonomies are also very useful in the field of news categorization (12), such as the one provided by the International Press Telecommunications Council\textsuperscript{4} and the RCV-taxonomy, proposed by Lewis (176) to perform hierarchical text categorization on the Reuters standard document collection. Therefore, devising effective hierarchical text categorization techniques has became a challenging issue in information retrieval.

The consideration of the hierarchical relationship among categories opens several additional issues in the development of methods for automated document classification. Questions concern the representation of documents, the learning process, the classification process, and the evaluation criteria of experimental results.

\textsuperscript{1}http://www.dmoz.org
\textsuperscript{2}http://www.wikipedia.org
\textsuperscript{3}http://medline.cos.com
\textsuperscript{4}http://www.iptc.org
4. HIERARCHICAL TEXT CATEGORIZATION

Most of the algorithms taken from the Text Categorization are often not suitable for Hierarchical Text Categorization purposes. Once applied to a Hierarchical classification problem, they are not capable of taking advantage of the information inherent in the class hierarchy, and may thus be suboptimal, in terms of efficiency and/or effectiveness.

On the other hand, the algorithms belonging to the state of the art such as Tree-Boost.MH embody several intuitions that had arisen before within Hierarchical Text Categorization: both feature selection and the selection of negative training examples should be performed locally, i.e., by paying attention to the topology of the classification scheme (177).

4.1 Formal Definitions

Before introducing the main contributions in the field of Hierarchical Text Categorization (HTC), let me give some formal definitions as done in (83).

**Definition (Hierarchical Text Categorization).** Hierarchical Text Categorization is a text categorization task performed according to a given taxonomy $H = \{C, \leq\}$, where $C = \{c_k \mid k = 1, 2, ..., N\}$ is a set of $N$ predefined categories and “$\leq$” is a reflexive, anti-symmetric, and transitive binary relation.

For any poset $H = \{C; \leq\}$ that represents a hierarchy, we assume the existence of the root category $\text{Root}(H)$ which is an ancestor of all other classes in the hierarchy: $\{\text{Root}(H)\} = \bigcap_{p \in C} \text{Ancestors}(p)$. The root category itself has no parent classes.
4.2 Reducing the Dimensionality of the Search Space

**Definition (Hierarchical Consistency).** A label set \( C_i \subseteq C \) assigned to an instance \( d_i \in D \) is called consistent with a given hierarchy if \( C_i \) includes the complete ancestor sets for every label \( c_k \in C_i \), i.e., if \( c_k \in C_i \) and \( c_j \in \text{Ancestors}(c_k) \), then \( c_j \in C_i \).

**Definition (Hierarchical Consistency Requirement).** Any label assignments produced by a hierarchical classification system on a given hierarchical text categorization task has to be consistent with a corresponding class hierarchy.

Hierarchical text categorization approaches can be classified according to how they reduce the dimensionality of the search space, build the training set, and which learning algorithm they adopt. In this section, I discuss relevant related work according to such classification.

### 4.2 Reducing the Dimensionality of the Search Space

Each document can be described by several sets of features, each of which is useful for the classification of the document at one level of the hierarchy. In hierarchical text categorization, feature selection has been used in a global or local way (83). A global approach to feature selection is similar to traditional feature selection in flat text categorization. A local approach, on the other hand, treats every internal category as a separate classification subtask and selects features for each subtask independently. Relevant features for a subtask are the ones that discriminate the children categories of the corresponding internal node.

A global approach is normally employed in systems where learning is also done in a global manner, i.e., only one classifier is built to distinguish among all categories in a hierarchy (178, 179). However, in (180) a global feature selection that splits the initial task into subtasks for learning a classifier is adopted. To manage an overwhelming number of potential global features, a simple and fast method of selecting a fixed, small number of terms from the beginning of each document, has been defined. McCallum et al. (147) use the global approach to classification while selecting the features locally at first and then taking the union of the feature subsets. Features are selected resorting to a mutual information approach.

A local approach to feature selection, unlike the global one, takes advantage of dividing a large initial problem into subproblems. The subproblems can differ substantially
4. HIERARCHICAL TEXT CATEGORIZATION

and, therefore, should be categorized by different terms. As a result, [89] proposed to
use local feature selection, i.e., choosing relevant features at each node of a hierarchy.
In their work they use expected cross-entropy and show that their hierarchical classifier
reaches the best performance with only a few words and its performance is superior to
the performance of a flat classifier. Before the pioneeristic work by Koller and Sahami,
there were a few works mentioning hierarchical categorization. Following that work,
a number of studies applied local feature selection for hierarchical text categorization
using different feature relevancy criteria such as Fisher index [140] [181] or information
gain [127]. Ruiz and Srinivasan [182] compare correlation, mutual information, and
odds ratio, finding that these three methods perform the same if mutual information
is enhanced with discarding low frequency terms. Moreover, Mladenčić and Grobelnik
[114] compare six different feature selection methods finding that odds ratio performs
better than the others. Also, they confirm Koller and Sahami’s conclusions that the
best results are achieved for relatively small feature subsets.

Specific feature selection methods for hierarchical categorization have also been
proposed [183]. In such work, features are chosen locally based on their frequency: a
feature is considered relevant for a category if its occurrence in the category is much
higher than in the parent category. In other words, since a parent category is formed as
a union of its children, a feature is relevant if it occurs in the category more frequently
than in the sibling categories altogether.

A comparison between the local and global approaches to feature selection has
been presented in [121]. Two methods, \(\chi^2\)-square and latent semantic indexing (LSI),
are used in combination with hierarchical neural networks. Surprisingly, the results
are mixed. On Reuters-22173 dataset the local feature selection with LSI outperforms
the corresponding global approach, whereas the local approach with \(\chi^2\)-square performs
better only on the subset of low frequency categories.

4.3 Building the Training Set

As it happens in TC, an important issue of HTC is document representation. As I
have depicted in the chapter 3 the most frequently used approaches in TC take into
account a bag of word representation of the text. Then for each document a vector
representation of features is built considering the most discriminant words with regards to the classification topic.

In the context of HTC a different, somewhat intermediate, solution can be adopted. Documents of both an internal category $C$ and its subcategories are represented by means of the same feature set, in order to build a classifier that assigns documents in $C$ to one of its direct subcategories. However, different internal categories may have different feature sets. In other words, by taking into account the hierarchy, it is possible to define several representation (sets of features) for each document. Each representation is useful for the classification of a document at one level of the hierarchy. For instance, documents of the general topic “Sport” can be well represented by general terms like “Game”, while documents concerning specific topics as “Football” are better represented from word as “Goalkeeper” or “Referee”.

To reach this goal, according to (175), training sets can be divided into two main categories (see figure 4.2): hierarchical training sets and proper training sets. Hierarchical training sets include documents of the subtree rooted in a category (positive examples) and documents of the sibling subtrees (negative examples). Proper training sets include documents of a category (positive examples) and documents of the sibling categories (negative examples).

![Diagram](image)

**Figure 4.2: Categories of training sets.** - Hierarchical Training Sets and Proper Training Set (taken by (175))

In the seminal work by Koller and Sahami (89) the hierarchy of categories is used in every processing step, included the adopted training set. In the classification step, which proceeds top-down, it is used to decide to which subtree the new document should be sent. Let me note that a limitation of the approach is that it is able to associate...
4. HIERARCHICAL TEXT CATEGORIZATION

documents only to the leaves of the hierarchy. McCallum et al. (147) proposed a method based on the naive Bayes learner. A unique feature set is defined for all documents by taking the union of all category vocabularies. Because of the uniqueness of the feature set, Bayesian classifiers associated at internal nodes are homogeneous, and the hierarchical organization of homogeneous classifiers is equivalent to a single flat classifier. Therefore, the hierarchical structure would have no practical impact on the classification process. Mladenić (114) used the hierarchical structure to decompose a problem into a set of subproblems, corresponding to categories (nodes in the hierarchy). The adopted training set is built from a set of positive examples, which is constructed from examples in the corresponding category node and all examples of its subtrees and a set of negative examples corresponding to all remaining documents. In (175) a hierarchical training set, which include documents of the subtree rooted in a category (positive examples) and documents of the sibling subtrees (negative examples), has been adopted. This choice is motivated by the fact that when no training document is associated to internal categories, proper training sets cannot be used, since it would be impossible to build a classifier.

In the literature, only few works have been presented that adopt a proper training set. In the work by (127) several SVMs are trained, one for each intermediate node, the sets of positive and negative examples being constructed from documents of categories at the same level. In the system CLASSI (107), the hierarchical classification of documents is obtained by combining several linear classifiers according to a tree structure (hierarchical classifier). The training set of each linear classifier includes all positive documents of the associated category and some selected documents of other categories. In (182) a variant of the Hierarchical Mixture of Experts (HME) model is used. As for the training set, the set of positive examples for an expert includes documents of the uniquely associated category while the set of positive examples for a gate includes all training documents of the set of associated categories. Some form of filtering is used for negative examples, since imbalanced data sets may affect the learning capability of backpropagation neural networks.

As for a final remark, let me recall that in (184) authors have showed that hierarchical training sets perform better than proper training sets.
4.4 The Learning Algorithm

Hierarchical categorization approaches can be divided into two categories (179, 185, 186): global (or big-bang) and local (or top-down level-based). A learning approach is called global if it builds only one classifier to discriminate all categories in a hierarchy. A global approach differs from flat categorization since it takes into account the relationships between the categories in a hierarchy. A learning approach is called local if it builds separate classifiers for internal nodes of a hierarchy. A local approach usually proceeds in a top-down fashion: first, it picks the most relevant categories of the top level and then, recursively, it makes the choice among the low-level categories. Both approaches have their weaknesses (186). The global approach is computationally heavy, it can not exploit different sets of features at different hierarchical levels, and it is not flexible (i.e., a classifier must be re-trained each time the hierarchical structure changes). The local approach, while computationally more efficient, has to make several correct decisions in a row to correctly classify one example, and errors made at top levels are usually not recoverable. Moreover, since at low levels the categories become smaller, the number of training examples can be insufficient to learn a reliable classifier.

Despite its computational overhead, the global approach has been employed in several text categorization systems. An association rule based method has been proposed in (185). In the subsequent study they extend their approach to accommodate multi-label categorization (187). Itskevitch proposes a slightly different approach for creating association rule hierarchical classifier (179). More recently, large margin classifiers have been extended to be applicable in the hierarchical settings (188). In this work, each node in a class hierarchy is associated with a prototype vector, and an instance is classified to the class with the most similar prototype. Another approach based on the maximum margin idea has also presented in the literature (189, 190). In (83) a general hierarchical procedure suitable for any conventional multi-label learning technique is presented. The approach has been experimented in bioinformatics applications.

In the literature many global approaches to hierarchical text categorization have been proposed. Nevertheless, local approaches are more popular due to their computational benefits. In general, local approaches can be divided into two groups: pachinko machine (sequential Boolean decisions) and probabilistic (multiplicative decisions). In pachinko machine a decision which path in a tree to take is made sequentially at each
4. HIERARCHICAL TEXT CATEGORIZATION

level of a hierarchy, while in a probabilistic method all paths are considered simultaneously, their probabilities are calculated as the product of individual probabilities of categories on a path and the most probable path is picked as a solution. Pachinko machine method has been widely used with different learning algorithms: linear [107, 180, 183], probabilistic [89, 180], decision rules [191], neural networks [182], and SVM [180, 186]. In [83] an extended version of pachinko machine method is proposed.

Probabilistic hierarchical local approaches are more computationally expensive than pachinko machine since classifiers have to be learnt at every node. Therefore, just a few studies on this technique have been proposed: it has been used in combination with probabilistic classifiers [192] and neural networks [115, 121].

Dumais and Chen compared the two local approaches, pachinko machine and probabilistic, and found no difference in performance [127].

There has been some effort in the research community to cope with one of the major problems of the local hierarchical approach: high-level error recovering. Cheng and colleagues propose two methods [181]. The first one is a pachinko machine with the possibility to return to the high-confidence ancestor node a few levels back if the classification probability drops below a given threshold. The second approach uses three different classification algorithms, two of which are standard pachinko machines with different feature sets and the third one is a classifier that dynamically skips some levels in the hierarchy. The decision is then made by the majority vote. In [193] several methods are proposed, where flat classification is performed at each level of a class hierarchy and the most probable category is chosen if its parent is also the most probable category at the level above; otherwise the best parent category is selected.

In [177] the absence of “boosting” methods is analyzed, and the TreeBoost.MH method is proposed to fill the gap concerning these interesting and high accurate algorithms. The TreeBoost.MH consists of a hierarchical variant of AdaBoost.MH, the most important family of the boosting algorithms.

In [175] the definition of the same feature set to represent document of a category \( C \) and all its subcategories permits the application of a multi-class learning algorithm to induce a classifier that categorizes a document (temporarily) assigned to \( C \) as belonging to a subcategory \( C' \) of \( C \). In their framework Ceci and Malerba consider three different learning approaches: (i) Naïve Bayes [194] modified in order to correctly handle documents of different length; (ii) A centroid-based method [195] where each centroid
Performance Evaluation

The performance of a TC System is usually measured in terms of its effectiveness, i.e., its ability to produce correct classification, and there exist several evaluation metrics for that (see section 3.3.3). Most studies on text categorization assume equal costs of different kinds of error, on the other hand some applications require non-unified misclassification costs. HTC calls for cost-sensitive learning since a hierarchical structure itself present built-in costs. For example, misclassifying a document in a sibling or parent of a correct category is intuitively preferable to misclassifying it to a very distant node. However, most work on HTC do not take into account the costs and evaluate system based on standard measurer (83).

Only a few researchers noted that HTC systems cannot be evaluated in a standard, “flat” way and, therefore, require special metrics. The first who pointed out the need of non-unified error costs were Wang and colleagues (185). In their work they assume that a set of misclassification costs for all category pairs $B(c_i, c_j)$ is given, and they optimize their association rule based classifier to minimize overall cost. As a possible choice for misclassification costs they propose to use the distance between nodes representing categories in a hierarchical tree. The distance between a correct and assigned category, $distance(c_i, c_j)$ is defined as the length of the shortest (undirected) path from node $c_i$ to node $c_j$ in a hierarchical graph. The same distance-based evaluation measure is used in (188). In their subsequent work Wang et al. replace distance-based costs with similarity-based costs (187). The similarity of two categories is measured as the similarity of the sets of documents belonging to the categories taking into account the difference in the coverage of the sets.
4. HIERARCHICAL TEXT CATEGORIZATION

Itskevich extends the work of Wang introducing probabilistic hierarchical measure based on the distance between categories in a hierarchical tree (179). Since a document can be classified into several categories (multi-label categorization), the probabilistic score of a classifier on a document $d_i$ is summed over all categories assigned to $d_i$.

Blockeel et al. also use a measure based on the distance between categories, but consider the weights for the edges of a hierarchy tree (197). The weights decrease exponentially with depth.

Cai and Hoffmann propose a more general measure based on a given set of user defined misclassification costs: $\text{cost}_1(v) \geq 0$ is the cost of assigning an item $d_i \in \text{Offspring}(v)$ to $w \notin \text{Offspring}(v)$, $\text{cost}_2(v) \geq 0$ is the cost of assigning an item $d_i \notin \text{Offspring}(v)$ to $w \in \text{Offspring}(v)$ (190). Then, the overall loss is computed as the sum of these costs for nodes in the symmetric difference of the ancestor sets of the real and predicted category.

Another measure, weighted penalty, is also suggested by (197). Here the nodes of the tree have weights (deeper nodes have smaller weights), and the distance between two nodes is calculated as the weight of their deepest common ancestor. Tsochantaridis et al. use a variant of this measure with weights defined as the height of a node in a hierarchical tree (189). Semantic similarity (198) can be seen as an extension of such a measure for the general case of a directed acyclic graph (DAG) hierarchy.

The semantic similarity measure is based on the minimal weight over all common ancestors of the two nodes. This measure has been specifically designed for the Gene Ontology. The weights are calculated as the probabilities of a category (or any of its descendant nodes) occurrence in given data. Sun and Lim propose their own category similarity and distance-based measures for hierarchical text categorization (186). Their category similarity (CS) measure is based on the content of documents comprising the categories and is computed as the cosine similarity between the feature vectors of two categories. The distance between two categories is the shortest distance between corresponding nodes in a hierarchy, similar to the one by Wang et al. (187). Then, based on these measures, category similarity and distance between categories, they modify the standard measures, precision/recall and accuracy/error. They consider

---

1 In mathematics and computer science, a directed acyclic graph (commonly abbreviated to DAG), is a directed graph with no directed cycles.
misclassification as being partly correct depending on how close the real and assigned categories are and add contributions from FP and FN to correct decisions (TP):

\[ A_i = \frac{TP_i + TN_i + FPCon_i + FNCon_i}{TP_i + TN_i + FP_i + FN_i} \] (4.1)

\[ E_i = \frac{FP_i + FN_i + FPCon_i - FNCon_i}{TP_i + TN_i + FP_i + FN_i} \] (4.2)

\[ \pi_i = \max(0, \frac{TP_i + FPCon_i + FNCon_i}{TP_i + FP_i + FNCon_i}) \] (4.3)

\[ \rho_i = \max(0, \frac{TP_i + FPCon_i + FNCon_i}{TP_i + FN_i + FPCon_i}) \] (4.4)

For category similarity based measures, the contribution of document \(d_j\) from FP (FN) to class \(c_i\) is defined as

\[ Con(d_j, C_i) = \sum_k (1 - \frac{\text{distance}(c_i, c_k)}{D_{acc}}), \] (4.5)

where summation is done over all document categories (for FPCon) or over all categories assigned to a document (for FNCon). Both types of contributions can be positive and negative and are restricted to the range \([-1,1]\).

An unusual approach to evaluation has been proposed by Ipeirotis and colleagues (191). Instead of measuring distance or content similarity of two category sets, correct \(C_i\) and predicted \(C'_i\), they measure the overlap in subtrees induced by the category sets.

In (83) Svetlana Kiritchenko tries to overcome this problem by formulating a natural, intuitive characteristics expected from a hierarchical evaluation measure. Hierarchical and non-hierarchical measures are compared, and a new hierarchical measure is proposed and compared. The new measure is the pair precision/recall with an addition: each example belongs/classified not only to a class, but also to all ancestors of the class in a hierarchical graph, except the root. Then the microaverage is calculated above the so called hierarchical precision \((h\pi)\) and hierarchical recall \((h\rho)\) and the subsequent hierarchical F-value \((hF_{\beta})\).

In the work of Ceci and Malerba (175) to show whether the hierarchical classifier built with their framework improves the performance when compared to a flat classifier several measures have been evaluated. The first one is a narrow defined accuracy, used
4. HIERARCHICAL TEXT CATEGORIZATION

to evaluate the performances of \(1 \leq r \leq r\) classifiers. Furthermore they define four additional measures:

1. The \textit{misclassification error}, which computes the percentage of documents misclassified into a category not related to the correct category in the hierarchy;

2. The \textit{generalization error}, which computes the percentage of documents misclassified into a supercategory of the correct category;

3. The \textit{specialization error}, which computes the percentage of document misclassified into a subcategory of the correct one;

4. The \textit{unknown ratio}, that measures the percentage of rejected documents.

This particular kind of measure will be exploited to evaluate the approach that this thesis proposes (see section \[7.2\]).

Evidently, a variety of measures try to move away from the conventional “flat” techniques and capture some of the specifics of hierarchical classification. Unfortunately, there are no well established criteria on what aspects of hierarchical information are the most significant and therefore, should be addressed in an evaluation measure. This is reflected in the diversity of the proposed techniques.

4.6 Applications

Most of the research were tested on classical text corpora Reuters-21578 (Reuters-22173)\footnote{http://www.davidlewis.com/resources/testcollections/reuters21578} 20-newsgroups\footnote{http://www.dmoz.com} and different subset of real-life web directories Yahoo! and DMOZ\footnote{http://www.uspto.com} Other real-life text collections with built-in hierarchies include US Patent database\footnote{http://portal.acm.org/dl.cfm} ACM Digital Library\footnote{http://www.looksmart.com} LookSmart Web Directory\footnote{http://www.looksmart.com} and biomedical articles OHSUMED\footnote{http://www.looksmart.com}.

In addition to this extensive set of text data, hierarchical text categorization has been successfully applied to other domains. Email classification is a subdomain of text categorization where topic hierarchies are often present\footnote{179, 200, 201}. Many users
organize (or would like to organize) their incoming email into a hierarchy of folders. Also, an automatic forwarding system can have the forwarding addresses organized in a hierarchy, for example by department. On the web there exist various topic hierarchies in the form of web directories that can be used not only to classify web pages but also to classify web databases. There are many web-accessible databases, such as Medline or ZDNet Product Review\(^1\) whose content is only accessible through search interfaces. For these resources, a special kind of categorization, database categorization is needed\(^{(191)}\).

Another interesting application of hierarchical text categorization is a user interface for search engines\(^{(202; 203)}\). Instead of returning a long flat list of documents that match a users query, these systems return the search results organized into an existing topic hierarchy. Such result presentation greatly helps users to quickly find the information they need\(^{(202)}\). This is achieved by simply classifying the results returned by a search engine into a predefined topic hierarchy.

However, a more efficient approach is to perform search in two steps: first, the categories that match a users query are identified and then, documents only from these categories that match the query are presented to the user\(^{(203)}\). The same approach can be applied to personalization systems where the search results are filtered and re-ranked according to a users preferences organized in a topic hierarchy, and then presented in a hierarchical structure\(^{(204; 205)}\).

Hierarchical text categorization can also be useful for question answering\(^{(206)}\). Before an automatic system attempts to answer a free form factual question from a given large collection of texts, it may be useful to first categorize the question into one of the semantic types from a given type hierarchy. For example, given the question “Who was the first female astronaut?” we know that the required answer is a person (or group of people) because of “who” and more specifically, that it is a woman. These semantic categories provide some constraints on the types of answers that should be found at later stages of the question answering process. Some sort of processing like this is being performed from the Google Squared lab\(^2\).

---

\(^1\)http://reviews-zdnet.com
\(^2\)http://www.google.com/squared
4. HIERARCHICAL TEXT CATEGORIZATION
Chapter 5

Hierarchical Text Categorization through a Progressive Filtering Approach

This thesis work is the inheritance of a problem I have tackled in 2006 concerning the exploitation of the knowledge about a taxonomy (i.e., the relationship among the classes representing a classification context), in order to improve the system performances. The algorithms I studied then, have found a natural application to the bioinformatics field (11; 14).

In particular, an infrastructure for the retrieval and classification of scientific publications has been developed. As a matter of fact, scientists need to retrieve and access a great amount of information useful to their study and research activity. The always increasing scientific yield on the biologic field (which was my case of study field at that time) tied together with the study of the human genome make it necessary to have automatic tools available. I worked to build a system providing these feature, employing the software agents engineering paradigm. Results achieved thanks to this approach (8) made it to originate the need to go into more depth with the study of the hierarchical techniques concerning text classification. In particular, being real text categorization application characterized by a huge imbalance between relevant (i.e., positive examples) and non-relevant (i.e., negative examples) documents, a novel progressive filtering approach deemed at remedy this issue has been gazed at (7).
5. HIERARCHICAL TEXT CATEGORIZATION THROUGH A PROGRESSIVE FILTERING APPROACH

Experimental results have shown that the proposed approach has better performances in comparison with the non-hierarchical (flat) approach. At the moment the results have been confirmed by an exhaustive experimental phase that allowed to compare the results with the state-of-the-art systems. Furthermore a theoretical study focused to the formal demonstration of the advantages of the proposed approach has been taken into account.

The study on the hierarchical categorization has been experimented exploiting the generic multiagent architecture X.MAS which will be better analyzed in chapter 6.

Figure 5.1: Strip - The need for improving search algorithms

A further field of study for the text categorization has comprised the automatic creation of datasets by a semantic approach. This study was developed into the DART\(^1\) project in collaboration with CRS\(^2\) researcher. Experiments performed on the Wordnet Domains taxonomy and Wikipedia articles shown that the approach is valuable. Moreover my works on the semantic approach to text categorization allowed to study user profiling with particular focus to the creation of contextual economical advertising.

\(^1\)DART (Distributed Agent-based Retrieval Tools) is a research project focused on studying, developing and testing of new information retrieval patterns and integrated tools to improve the quality of search engines results, with the main objective of satisfying user needs, http://www.crs4.it/ict/nda/doku.php?id=projects:dart

\(^2\)CRS4 (Center for Advanced Studies, Research and Development in Sardinia) is an interdisciplinary research center developing advanced simulation techniques and applying them, by means of High Performance Computing, to the solution of large scale computational problems, and developing innovative applications in the field of the Information and Communications Technology, see http://www.crs4.it/
5.1 The Progressive Filtering Approach

As discussed in chapter 4, HTC approaches can be classified according to which extent they reduce the dimensionality of the search space, the way they build the training set, and the adopted learning algorithm. In this chapter I first illustrate my approach according to the particular features of the analyzed scenario, then the algorithm for automated threshold selection will be presented and its impact discussed. Finally its learning complexity will be studied. In the chapter all the theory concerning the algorithm will be exhibited from an experimental point of view.

5.1 The Progressive Filtering Approach

What I propose is a so called “Progressive Filtering Approach” (PFA). Such approach consists of a progressive evaluation of the information performed by a pipeline of classifiers deemed at selecting the relevant information with an always higher competence falling from the root node.

Thus, in such approach the first thing to do is to unfold the taxonomy into pipelines of classifiers, as depicted in figure 5.2.

![Figure 5.2: A taxonomy and its corresponding pipelines.](image)

Each pipeline is composed by a number of classifiers equal to the depth of the corresponding node, each node of the pipeline being a binary classifier able to recognize whether or not an input belongs to the corresponding class (i.e., to the corresponding node of the taxonomy).

5.1.1 Document representation

As for document representation, I adopt the typical method of representing texts by using the bag of words approach, in which each word from a vocabulary corresponds to
a feature and a document to a feature vector. First, all non-informative words such as prepositions, conjunctions, pronouns and very common verbs are disregarded by using a stop-word list (see the example in table 5.1).

<table>
<thead>
<tr>
<th>Word</th>
<th>Stem</th>
</tr>
</thead>
<tbody>
<tr>
<td>about</td>
<td></td>
</tr>
<tr>
<td>after</td>
<td></td>
</tr>
<tr>
<td>all</td>
<td></td>
</tr>
<tr>
<td>also</td>
<td></td>
</tr>
<tr>
<td>an</td>
<td></td>
</tr>
<tr>
<td>and</td>
<td></td>
</tr>
<tr>
<td>another</td>
<td></td>
</tr>
<tr>
<td>any</td>
<td></td>
</tr>
<tr>
<td>are</td>
<td></td>
</tr>
<tr>
<td>as</td>
<td></td>
</tr>
<tr>
<td>at</td>
<td></td>
</tr>
<tr>
<td>because</td>
<td></td>
</tr>
<tr>
<td>been</td>
<td></td>
</tr>
<tr>
<td>before</td>
<td></td>
</tr>
<tr>
<td>being</td>
<td></td>
</tr>
<tr>
<td>between</td>
<td></td>
</tr>
<tr>
<td>both</td>
<td></td>
</tr>
<tr>
<td>but</td>
<td></td>
</tr>
<tr>
<td>came</td>
<td></td>
</tr>
<tr>
<td>can</td>
<td></td>
</tr>
<tr>
<td>come</td>
<td></td>
</tr>
<tr>
<td>could</td>
<td></td>
</tr>
<tr>
<td>did</td>
<td></td>
</tr>
<tr>
<td>do</td>
<td></td>
</tr>
<tr>
<td>does</td>
<td></td>
</tr>
<tr>
<td>each</td>
<td></td>
</tr>
<tr>
<td>else</td>
<td></td>
</tr>
<tr>
<td>for</td>
<td></td>
</tr>
<tr>
<td>from</td>
<td></td>
</tr>
<tr>
<td>get</td>
<td></td>
</tr>
<tr>
<td>got</td>
<td></td>
</tr>
<tr>
<td>has</td>
<td></td>
</tr>
<tr>
<td>had</td>
<td></td>
</tr>
<tr>
<td>he</td>
<td></td>
</tr>
<tr>
<td>have</td>
<td></td>
</tr>
<tr>
<td>her</td>
<td></td>
</tr>
<tr>
<td>here</td>
<td></td>
</tr>
<tr>
<td>him</td>
<td></td>
</tr>
<tr>
<td>himself</td>
<td></td>
</tr>
<tr>
<td>his</td>
<td></td>
</tr>
<tr>
<td>how</td>
<td></td>
</tr>
<tr>
<td>if</td>
<td></td>
</tr>
<tr>
<td>in</td>
<td></td>
</tr>
<tr>
<td>into</td>
<td></td>
</tr>
<tr>
<td>is</td>
<td></td>
</tr>
<tr>
<td>its</td>
<td></td>
</tr>
<tr>
<td>just</td>
<td></td>
</tr>
<tr>
<td>like</td>
<td></td>
</tr>
<tr>
<td>make</td>
<td></td>
</tr>
<tr>
<td>many</td>
<td></td>
</tr>
<tr>
<td>me</td>
<td></td>
</tr>
<tr>
<td>might</td>
<td></td>
</tr>
<tr>
<td>more</td>
<td></td>
</tr>
<tr>
<td>most</td>
<td></td>
</tr>
<tr>
<td>much</td>
<td></td>
</tr>
<tr>
<td>must</td>
<td></td>
</tr>
<tr>
<td>my</td>
<td></td>
</tr>
<tr>
<td>never</td>
<td></td>
</tr>
<tr>
<td>now</td>
<td></td>
</tr>
<tr>
<td>of</td>
<td></td>
</tr>
<tr>
<td>on</td>
<td></td>
</tr>
<tr>
<td>only</td>
<td></td>
</tr>
<tr>
<td>or</td>
<td></td>
</tr>
<tr>
<td>other</td>
<td></td>
</tr>
<tr>
<td>our</td>
<td></td>
</tr>
<tr>
<td>out</td>
<td></td>
</tr>
<tr>
<td>over</td>
<td></td>
</tr>
<tr>
<td>re</td>
<td></td>
</tr>
<tr>
<td>said</td>
<td></td>
</tr>
<tr>
<td>same</td>
<td></td>
</tr>
<tr>
<td>see</td>
<td></td>
</tr>
<tr>
<td>should</td>
<td></td>
</tr>
<tr>
<td>since</td>
<td></td>
</tr>
<tr>
<td>so</td>
<td></td>
</tr>
<tr>
<td>some</td>
<td></td>
</tr>
<tr>
<td>still</td>
<td></td>
</tr>
<tr>
<td>such</td>
<td></td>
</tr>
<tr>
<td>take</td>
<td></td>
</tr>
<tr>
<td>than</td>
<td></td>
</tr>
<tr>
<td>that</td>
<td></td>
</tr>
<tr>
<td>the</td>
<td></td>
</tr>
<tr>
<td>their</td>
<td></td>
</tr>
<tr>
<td>them</td>
<td></td>
</tr>
<tr>
<td>then</td>
<td></td>
</tr>
<tr>
<td>there</td>
<td></td>
</tr>
<tr>
<td>these</td>
<td></td>
</tr>
<tr>
<td>this</td>
<td></td>
</tr>
<tr>
<td>those</td>
<td></td>
</tr>
<tr>
<td>through</td>
<td></td>
</tr>
<tr>
<td>to</td>
<td></td>
</tr>
<tr>
<td>too</td>
<td></td>
</tr>
<tr>
<td>under</td>
<td></td>
</tr>
<tr>
<td>up</td>
<td></td>
</tr>
<tr>
<td>use</td>
<td></td>
</tr>
<tr>
<td>very</td>
<td></td>
</tr>
<tr>
<td>want</td>
<td></td>
</tr>
<tr>
<td>was</td>
<td></td>
</tr>
<tr>
<td>way</td>
<td></td>
</tr>
<tr>
<td>we</td>
<td></td>
</tr>
<tr>
<td>well</td>
<td></td>
</tr>
<tr>
<td>were</td>
<td></td>
</tr>
<tr>
<td>what</td>
<td></td>
</tr>
<tr>
<td>when</td>
<td></td>
</tr>
<tr>
<td>where</td>
<td></td>
</tr>
<tr>
<td>which</td>
<td></td>
</tr>
<tr>
<td>while</td>
<td></td>
</tr>
<tr>
<td>who</td>
<td></td>
</tr>
<tr>
<td>will</td>
<td></td>
</tr>
<tr>
<td>with</td>
<td></td>
</tr>
<tr>
<td>would</td>
<td></td>
</tr>
<tr>
<td>you</td>
<td></td>
</tr>
<tr>
<td>your</td>
<td></td>
</tr>
</tbody>
</table>

Table 5.1: An example of Stop Word List

Subsequently, the most common morphological and inflexional suffixes are removed by adopting a standard stemming algorithm (99) (see the example in table 5.2). This could bring some error, as shown in the last rows of the table 5.2, but is a really reasonable compromise to get the computational improvement taken by this method.

<table>
<thead>
<tr>
<th>Word</th>
<th>Stem</th>
</tr>
</thead>
<tbody>
<tr>
<td>circus</td>
<td>circu</td>
</tr>
<tr>
<td>antihero</td>
<td>antihero</td>
</tr>
<tr>
<td>reflection</td>
<td>reflect</td>
</tr>
<tr>
<td>running</td>
<td>run</td>
</tr>
<tr>
<td>earnings</td>
<td>earn</td>
</tr>
<tr>
<td>bowling</td>
<td>bowl</td>
</tr>
<tr>
<td>abbreviated</td>
<td>abbrevi</td>
</tr>
<tr>
<td>thumb</td>
<td>thumb</td>
</tr>
<tr>
<td>axes</td>
<td>axe</td>
</tr>
<tr>
<td>axis</td>
<td>axe</td>
</tr>
</tbody>
</table>

Table 5.2: Some example of Stemming
5.1 The Progressive Filtering Approach

After having determined the overall sets of features, their values are computed for each document resorting to the well-known TF-IDF method presented in section 2.4. To reduce the high dimensionality of the feature space, I locally select the features that represent a node by adopting the information gain method discussed in section 3.3.1.

5.1.2 The learning algorithm

As for the training activity, a hierarchical training set is built to train all classifiers. For any given node, the training set is composed by documents of the subtree rooted in a category as positive examples, and documents of the sibling subtrees as negative examples (see figure 4.2a).

The novel part of the approach relies on the adopted learning algorithm. Given a taxonomy, where each node represents a classifier entrusted with recognizing all corresponding positive inputs (i.e., relevant documents), any given input traverses the taxonomy as a “token”, starting from the root. If the current classifier recognizes the token as positive, it passes it on to all its children (if any), and so on. The typical result consists of activating one or more branches within the taxonomy, in which the corresponding classifiers have been activated by the given token. Let me note that, partitioning the taxonomy in pipelines gives rise to a set of new classifiers, each one corresponding to a pipeline. As for the taxonomy depicted in figure 5.4, each of the six pipelines corresponds to a classifier. This means that to equal classifiers does not correspond an equal behavior. Each classifier, in fact, is trained in accordance with the pipeline it belongs to and its confusion matrix is concerned with the last node of the pipeline. Then, for each classifier in the pipeline a threshold-selection algorithm (described in section 5.3) is applied and thresholds are calculated taking into account all the classifiers of the pipeline. For instance, the partitioning of the taxonomy depicted in figure 5.2 corresponds to six pipelines (i.e., six classifiers). According to a threshold-selection algorithm, the pipeline $\Pi_1 = \{A, B, C\}$ could recognize as positive an input $X$, whereas the $\Pi_2 = \{A, B\}$ does not. Let me also note that, in principle, the proposed approach does not guarantee the hierarchical consistency requirement. Actually, in this work I consider only the results of the leaves of the pipelines, so that the hierarchical consistency requirement is preserved. As a future work, I am also planning to study how to preserve this requirement considering all the nodes of the taxonomy. See (207) for a detailed study on PFA.
5. HIERARCHICAL TEXT CATEGORIZATION THROUGH A PROGRESSIVE FILTERING APPROACH

5.1.3 The input imbalance issue

In this work, I am also interested in studying PFA taking into account the input imbalance that typically occurs in real-world scenarios (see also section 2.3.2). Japkowicz (208) studied the class imbalance problem, i.e., the problem related to domains in which (in binary classification) one class is represented by a large number of examples whereas the other is represented by only a few. Actually, this and other works (209; 210; 211) are concerned with the imbalance that occurs in the training set. To this end, in my work, I perform the training activity in two phases. First, each classifier is trained by using a balanced dataset; then, the threshold algorithm is applied and thresholds are calculated taking into account the input imbalance. The underlying motivation is that, in real world applications, the ratio between relevant and non-relevant documents is typically very low, so that classifiers trained with a balanced training set are not adequate to deal with those environments. To my best knowledge, proposed hierarchical text categorization techniques do not take into account this issue, in my opinion –in order to tackle real-world problems– they should do it. The algorithm I propose in the next section handles with this problem, and it will be shown how its performance improve in an imbalance context.

5.2 The Impact of PFA

In this section I assess the impact of the proposed approach from a theoretical perspective. In particular, my target is (i) putting into evidence how the proposed approach could overwhelm the imbalance between relevant and non-relevant documents, and (ii) demonstrating that the threshold algorithm is computationally more efficient than a greedy approach.

With the aim of assessing how much the use of pipelines of classifiers allows to cope with the imbalance between relevant and not-relevant documents, let me consider the pipeline Π depicted in figure 5.3.

5.2.1 Overall Transformation Performed by a Pipeline of Classifiers

The first step toward this goal consists of separating the intrinsic (expected) behavior of a classifier from the set of inputs, say \( X \). Let me denote with \( \Xi_C(X) \) the confusion matrix of a classifier \( C \) applied to \( X \), calculated considering the behavior of \( C \) with
5.2 The Impact of PFA

Figure 5.3: An example of pipeline \{C_1 \cdots C_L\}.

respect to the last classifier of the pipeline (i.e., a leaf). \(\Xi_C(X)\) is populated by \(p\) positive examples and \(n\) negative instances, let me note that with positive (negative) examples I mean examples that belong (not belong) to the last classifier of the pipeline. Under the hypothesis of statistical significance, I can write:

\[
\Xi_C(X) = \begin{bmatrix} n & 0 \\ 0 & p \end{bmatrix} \cdot \Gamma(C) \tag{5.1}
\]

In other words, I assume that the transformation performed by a classifier \(C\) can be isolated from the inputs it processes, at least from a statistical perspective. In so doing, the confusion matrix for a given set of inputs is obtained as the product between (i) a term that accounts for the number of positive \((p)\) and negative \((n)\) instances and (ii) a term \(\Gamma(C)\) that represents the expected transformation performed by \(C\) on the given input set.

Let me now make explicit the transformation reported by Equation (5.1):
5. HIERARCHICAL TEXT CATEGORIZATION THROUGH A PROGRESSIVE FILTERING APPROACH

\[ \Xi_{C}(X) = \begin{bmatrix} TN & FP \\ FN & TP \end{bmatrix} = \begin{bmatrix} n & 0 \\ 0 & p \end{bmatrix} \begin{bmatrix} \gamma_{00} & \gamma_{01} \\ \gamma_{10} & \gamma_{11} \end{bmatrix} = \begin{bmatrix} n \cdot \gamma_{00} & n \cdot \gamma_{01} \\ p \cdot \gamma_{10} & p \cdot \gamma_{11} \end{bmatrix} \tag{5.2} \]

where \( \gamma_{ij}, i, j = 0, 1 \) denotes the percent of inputs (belonging to class “i” and classified as “i”) that are expected to be correctly classified \((i = j)\) or misclassified \((i \neq j)\). \( \Gamma \) will be called normalized confusion matrix hereinafter, to emphasize the fact that its rows sum to 1 (easy to verify from Equation (5.2)).

To study the properties of a pipeline of classifiers, it is important to obtain a closed formula able to represent the behavior of the pipeline as it were in fact a single classifier.

In order to reach this target I do a statistical independence hypothesis on the behavior of the classifiers. This means that the confusion matrix of a classifier belonging to the pipeline is independent (i.e., it is not affected) by the behavior of the other classifiers in the pipeline. This fact is not true in general, but allow to extremely simplify the theoretical study of the phenomenon that in its more general paint is not modified by the classifier independency.

Going back to \( \Pi \), let me denote with \( \Pi_{k}, k = 1, 2, \ldots, L \) the pipeline truncated at the \( k \)-th element.

The representation for a generic \( L > 0 \) can be obtained by defining a base case \((k = 1)\) together with a recursive step \((k > 1)\). In absence of ambiguities, the normalized confusion matrix of a pipeline \( \Pi_{k} \) will be denoted as \( \Psi(\Pi_{k}) \) and their components \( \psi_{ij}^{(k)}, i, j = 0, 1, 1 \).

**Base case** \((k = 1)\):

\[ \Psi(\Pi_{1}) = \begin{bmatrix} \psi_{00}^{(1)} & \psi_{01}^{(1)} \\ \psi_{10}^{(1)} & \psi_{11}^{(1)} \end{bmatrix} = \begin{bmatrix} \gamma_{00}^{(1)} & \gamma_{01}^{(1)} \\ \gamma_{10}^{(1)} & \gamma_{11}^{(1)} \end{bmatrix} = \Gamma(C_{1}) \tag{5.3} \]

**Recursive step** \((k > 1)\):

\[ \Psi(\Pi_{k}) = \begin{bmatrix} \psi_{00}^{(k)} & \psi_{01}^{(k)} \\ \psi_{10}^{(k)} & \psi_{11}^{(k)} \end{bmatrix} = \begin{bmatrix} \psi_{00}^{(k-1)} & 0 \\ \psi_{10}^{(k-1)} & 0 \end{bmatrix} + \begin{bmatrix} 0 & \psi_{01}^{(k-1)} \\ 0 & \psi_{11}^{(k-1)} \end{bmatrix} \cdot \Gamma(C_{k}) \tag{5.4} \]

\(^{1}\)With the assumption that “0” (“1”) means means negative (positive).
5.3 The Threshold-Selection Algorithm

By unfolding the recurrence relation given by Equations 5.3 and 5.4 one can obtain an explicit representation of the transformation performed by a pipeline Π. In so doing, I obtain (k = 1, 2, ..., L):

\[
\Psi(\Pi_k) = \left[ \sum_{j=1}^{k} (\gamma^{(j)}_{00} \cdot \prod_{i=1}^{j-1} \gamma^{(i)}_{10} \cdot \prod_{i=1}^{k} \gamma^{(i)}_{10}) \prod_{i=1}^{k} \gamma^{(i)}_{11} \right] \left[ \sum_{j=1}^{k} (\gamma^{(j)}_{10} \cdot \prod_{i=1}^{j-1} \gamma^{(i)}_{01} \cdot \prod_{i=1}^{k} \gamma^{(i)}_{01}) \prod_{i=1}^{k} \gamma^{(i)}_{01} \right] = \left[ 1 - \prod_{i=1}^{k} \gamma^{(i)}_{01} \prod_{i=1}^{k} \gamma^{(i)}_{01} \right] \left[ 1 - \prod_{i=1}^{k} \gamma^{(i)}_{11} \prod_{i=1}^{k} \gamma^{(i)}_{11} \right]
\]

(5.5)

In any case, apparently, Ψ(Π) plays for Π the role that Γ(C) plays for C. Since Ψ(Π_k), k = 1, 2, ..., L is still normalized (one can easily verify it by induction from Equations 5.3 and 5.4), we are now permitted to see a pipeline as a single classifier, whose behavior is represented by Ψ. In symbols:

\[
\Xi_{\Pi}(X) = \begin{bmatrix} n & 0 \\ 0 & p \end{bmatrix} \cdot \Psi(\Pi)
\]

(5.6)

We know from Equation 5.6 that the elements of Ψ(Π_k) can be used to represent the overall transformation performed by Π_k. Hence, to calculate precision (P) and recall (R) for any given pipeline Π_k is now straightforward:

\[
\pi(\Pi_k) = \left( 1 + \frac{n}{p} \cdot \frac{\psi^{(k)}_{01}}{\psi^{(k)}_{11}} \right)^{-1} = \left( 1 + \frac{n}{p} \cdot \prod_{i=0}^{k} \frac{\gamma^{(i)}_{01}}{\gamma^{(i)}_{11}} \right)^{-1}
\]

(5.7)

\[
\rho(\Pi_k) = \psi^{(k)}_{11} = \prod_{i=1}^{k} \gamma^{(i)}_{11}
\]

(5.8)

5.3 The Threshold-Selection Algorithm

In a hierarchical classification context, a classifier is learned for each internal category C of the hierarchy. As Ceci and Malerba point out in their work (175), the classifier is used to decide, during the classification of a new document, which category C’ is the most appropriate to receive the document. In general, however, a document should not be necessarily passed down to a subcategory of C. To support the classification of such documents, it is necessary to compute the thresholds that represent the “minimal score” returned by the classifier, such that a document can be considered to belong to a direct subcategory.
5. HIERARCHICAL TEXT CATEGORIZATION THROUGH A PROGRESSIVE FILTERING APPROACH

Their algorithm for automated threshold determination is based on a bottom-up strategy and tries to minimize a measured based on a tree distance. The algorithm is recursive and takes as input the category $C$ and a set of thresholds already computed for some siblings of $C$ and their descendants (only the root at the first invocation). It returns the union of the input set of thresholds computed for all descendants of $C$. In particular, if $C'$ is a direct subcategory of $C$, the threshold $Th_C\left(C'\right)$ associated to $C'$ is determined by examining the sorted list $V$ of classification scores and by selecting the middle point between the two values in $V$, such that the expected error (estimated on the basis of the distance between two nodes in a tree structure) is minimized.

In my proposal, as already discussed, each pipeline can be viewed as a classifier, so that finding the best configuration of a pipeline corresponds to calculate the best combination of thresholds of the involved classifiers. In fact, the best (combination of) threshold(s) strictly depends on the input imbalance.

Let me recall here that in classical TC (see chapter 3), once defined a function $CSV_i : D \rightarrow [0, 1]$ that, given a document $d$ belonging to a set of documents $D$, returns a categorization status value for it (i.e., the evidence that $d$ should be categorized under $C_i$), a threshold $\theta_i$ is defined such that $CSV_i(d) \geq \theta_i$ is interpreted as a decision to categorize $d$ under $C_i$, while $CSV_i(d) < \theta_i$ is interpreted as a decision not to categorize $d$ under $C_i$. A particular case occurs when the classifier provides a binary judgment, i.e., $CSV_i : D \rightarrow \{0, 1\}$. In this case, the threshold is trivially any value in the (0,1) open interval. In PFA, given a pipeline $\Pi$ composed by $n$ classifiers, the corresponding $CSV_i$ can be defined as $CSV_i : D \rightarrow [0, 1]$ meaning that, given a document $d$ and a class $C_i$ returns the evidence that $d$ should be categorized under $C_i$ for each classifier belonging to $\Pi$. Therefore, a vector of thresholds $\overrightarrow{\theta_i}$ is defined such that, for each component $\theta_k$ with $k = 1..n$, $CSV_i(d) \geq \theta_k$ is interpreted as a decision to categorize $d$ under $C_i$, i.e., each classifier that belongs to the pipeline recognizes $d$ as belonging to the corresponding category. On the contrary, $CSV_i(d) < \theta_k$ is interpreted as a decision to not categorize $d$ under $C_i$, i.e., at least one classifier that belongs to the pipeline does not recognize $d$ as belonging to it.

For each pipeline $\Pi$ the best combination of thresholds, initially set to 0, is calculated, considering the actual ratio between positive and negative examples that depends on the given scenario.
5.3 The Threshold-Selection Algorithm

Bearing in mind that the lower the threshold the less restrictive the classifier, the best combination of thresholds is calculated according to a bottom-up algorithm that relies on two functions:

- **repair**, which operates on the given classifier by increasing (↑) or decreasing (↓) its threshold until the selected utility function reaches a maximum.

- **calibrate**, which operates going downwards from the given classifier to its offspring by repeatedly calling repair. It is intrinsically recursive and at each step it calls repair to actually calibrate the current classifier.

To define the threshold-selection algorithm, say TSA, I borrow from modal logic the operators □ and ♦, used in this context to denote repair and calibrate, respectively. For instance, the statement ↑ □ C denotes a repair step that operates on the C classifier by increasing (↑) its threshold until reaches a maximum.

TSA is defined as follows, all thresholds being initially set to zero:

\[
\text{TSA} := \text{fork } k = L \text{ downto } 1 \text{ do } \uparrow \Diamond C_k
\]

which indicates that ♦ is applied at each node of a pipeline, starting from the leaf (\(k = L\)), \(L\) being the depth of the current pipeline.

Given a pipeline, the calibrate function is defined as follows:

\[
\begin{align*}
\uparrow \Diamond C_k := & \uparrow \Box C_k, \ k = L \\
\uparrow \Diamond C_k := & \uparrow \Box C_k + \downarrow \Diamond C_{k+1}, \ k < L
\end{align*}
\]

and

\[
\begin{align*}
\downarrow \Diamond C_k := & \downarrow \Box C_k, \ k = L \\
\downarrow \Diamond C_k := & \downarrow \Box C_k + \uparrow \Diamond C_{k+1}, \ k < L
\end{align*}
\]

where the + operator actually denotes a sequence operator, meaning that in the formula a + b action a is performed before action b.

To better illustrate the approach, let me consider the unfolding, reported in figure 5.4, which corresponds to the pipeline of three classifiers \(\Pi = \{C_1, C_2, C_3\}\):

\[
\begin{align*}
\text{TSA} := \text{for } k = 3 \text{ downto } 1 \text{ do } \uparrow \Diamond C_k &= \\
\uparrow \Diamond C_3 := & \uparrow \Box C_3, \text{ step 1} \\
\uparrow \Diamond C_2 := & \uparrow \Box C_2 + \downarrow \Diamond C_3 = \uparrow \Box C_2 + \downarrow \Box C_3, \text{ step 2} \\
\uparrow \Diamond C_1 := & \uparrow \Box C_1 + \downarrow \Diamond C_2 = \uparrow \Box C_1 + \downarrow \Box C_2 + \uparrow \Diamond C_3 = \\
& \uparrow \Box C_1 + \downarrow \Box C_2 + \uparrow \Box C_3, \text{ step 3}
\end{align*}
\]
5. HIERARCHICAL TEXT CATEGORIZATION THROUGH A PROGRESSIVE FILTERING APPROACH

![Diagram showing the unfolding of the threshold-selection procedure for a pipeline composed by three classifiers.]

**Figure 5.4:** Unfolding the threshold-selection procedure for a pipeline composed by three classifiers.

Once calculated the best combination of thresholds for a given imbalance, the pipelines are ready to be used in the corresponding scenario.

As noted in the previous section, the behavior of $\Pi_1 = \{C_1, C_2, C_3\}$ is different from the one of $\Pi_2 = \{C_1, C_2\}$ and of $\Pi_3 = \{C_1\}$.

### 5.3.1 How Precision, Recall and F1 Change along a Pipeline

To better assess the effectiveness of PFA, let me analyze the performance of a pipeline $\Pi$ along the pipeline itself. My analysis proceeds by induction, assuming of having assessed the behavior of a given pipeline of $k$ classifiers, and then verifying what happens after adding one further classifier to the pipeline.

For the sake of simplicity, in the following I will denote as $\psi_{ij}, i, j = 0, 1$ the elements of $\Psi(\Pi_k)$, with $\psi'_{ij}, i, j = 0, 1$ the elements of $\Psi(\Pi_{k+1})$, and with $\gamma_{ij}, i, j = 0, 1$ the elements of $\Gamma(C_{k+1})$.

First of all, let me make explicit the dependency between a generic term of $\Psi(\Pi_{k+1})$ and the terms of $\Psi(\Pi_k)$. From Equation (5.6):

$$
\psi'_00 = \psi_{00} + \psi_{01} \cdot \gamma_{00} \\
\psi'_10 = \psi_{10} + \psi_{11} \cdot \gamma_{10} \\
\psi'_01 = \psi_{01} \cdot \gamma_{01} \\
\psi'_11 = \psi_{11} \cdot \gamma_{11}
$$

It is now straightforward to write precision and recall for a pipeline $\Pi_{k+1}$:
5.3 The Threshold-Selection Algorithm

\[ \pi(\Pi_{k+1}) = \left( 1 + n \cdot \frac{\psi'_{01}}{\psi_{11}} \right)^{-1} = \left( 1 + \frac{n}{p} \cdot \frac{\psi_{01}}{\psi_{11}} \cdot \frac{\gamma_{01}}{\gamma_{11}} \right)^{-1} \]  
(5.13)

\[ \rho(\Pi_{k+1}) = \psi'_{11} = \psi_{11} \cdot \gamma_{11} \]  
(5.14)

As for precision:

\[ \pi(\Pi_{k+1}) - \pi(\Pi_k) \geq 0 \iff \gamma_{01} \leq \gamma_{11} \]  
(5.15)

which indicates that \( \pi \) is monotonically increasing along a pipeline provided that \( \gamma_{01} \leq \gamma_{11} \) for each classifier of the pipeline.

As for recall:

\[ \rho(\Pi_{k+1}) - \rho(\Pi_k) \geq 0 \iff \psi_{11} \cdot \gamma_{10} \leq 0 \]  
(5.16)

which is clearly satisfied only when \( \gamma_{10} = 0 \) for each classifier of the pipeline. In other words, \( \rho \) is monotonically decreasing along a pipeline, the higher \( \gamma_{10} \) (i.e., the percent of false negatives) the greater the decrement of \( \rho \).

It is worth pointing out that neither precision nor recall typically make sense in isolation of the other. Hence, to better assess the characteristics of the proposed approach, let me also study the behavior of the \( F_1 \) measure (39), which combines precision and recall giving equal importance to them.

\[ F_1 = 2 \cdot \left( \frac{1}{\pi} + \frac{1}{\rho} \right)^{-1} = 2 \cdot \left[ \left( 1 + \frac{n}{p} \cdot \frac{\psi_{01}}{\psi_{11}} \right) + \left( \frac{1}{\psi_{11}} \right) \right]^{-1} \]  
(5.17)

\[ F_1(\Pi_{k+1}) - F_1(\Pi_k) \geq 0 \iff \frac{n}{p} \cdot \psi_{01} \cdot \frac{1 - (\gamma_{10} + \gamma_{01})}{\gamma_{10}} \geq 1 \]  
(5.18)

which means that, in presence of imbalance, PFA typically guarantees an improvement along the pipeline as long as the term \( \psi_{01} = \Pi_{i=1}^{k}(\gamma_{01}^{(i)}) \) (which decreases monotonically) is counterbalanced by the ratio \( n/p \). In other words, due to the exponential behavior of \( \psi_{01} \), \( F_1 \) may increase along a pipeline depending on the rate of misclassifications, until a value, say \( k^* = \arg \min_k K(n/p, \Psi^{(k)}) \) (which clearly depends on the imbalance) is reached, the greater \( n/p \) the higher \( k^* \). For instance, with a ratio \( p/n = 1000 \) and \( \gamma_{01} = \gamma_{10} = 0.1 \), then \( k^* = 4 \).
5. HIERARCHICAL TEXT CATEGORIZATION THROUGH A PROGRESSIVE FILTERING APPROACH

Table 5.3: PFA vs. FLAT approach.

<table>
<thead>
<tr>
<th></th>
<th>PFA ((\Pi_k = C_k \ldots \circ C_2 \circ C_1))</th>
<th>FLAT ((C_k))</th>
</tr>
</thead>
<tbody>
<tr>
<td>(P)</td>
<td>(\left(1 + \frac{n}{p} \cdot \prod_{i=0}^{k} \frac{\gamma^{(i)}}{\gamma^{(i1)}}\right)^{-1})</td>
<td>(\left(1 + \frac{n}{p} \cdot \frac{\gamma^{(k)}}{\gamma^{(k1)}}\right)^{-1})</td>
</tr>
<tr>
<td>(R)</td>
<td>(\prod_{i=1}^{k} \gamma^{(i)}) (\gamma^{(k1)})</td>
<td>(\gamma^{(k)})</td>
</tr>
<tr>
<td>(F_1)</td>
<td>(2 \cdot \left(1 + \frac{n}{p} \cdot \prod_{i=0}^{k} \frac{\gamma^{(i)}}{\gamma^{(i1)}} + \prod_{i=1}^{k} \frac{1}{\gamma^{(i1)}}\right)^{-1})</td>
<td>(2 \cdot \left(1 + \frac{n}{p} \cdot \frac{\gamma^{(k)}}{\gamma^{(k1)}} + \frac{1}{\gamma^{(k1)}}\right)^{-1})</td>
</tr>
</tbody>
</table>

5.3.2 Comparing PFA with the FLAT Approach

Yet more important is to assess what happens while comparing PFA with the flat approach. Starting from Equation (5.7) and (5.8), the comparison is straightforward. Results are reported in table 5.3.

In principle, PFA is neither better nor worse than the flat approach. In fact, precision is monotonically increasing whereas recall is monotonically decreasing along a pipeline. As for \(F_1\), it follows a “precision-like” behavior first, followed by a “recall-like” behavior (one can easily understand why looking at \(F_1\) as reported in table 5.3).

Nevertheless, PFA permits to control the amount of false positives and false negative with more degrees of freedom. In fact, assuming the behavior of a classifier can be controlled by acting on an acceptance threshold \(\theta\), it turns out that several classifiers in pipeline can better adapt to a given problem, due to the possibility of selecting a different threshold along a pipeline. In other words, a pipeline \(\Pi = \{C_1, \ldots, C_L\}\) can better adapt to a specific classification problem for its tuning capability performed on the corresponding vector of thresholds, say \(\vec{\theta} = (\theta_L, \ldots, \theta_1)\). Unfortunately, the task of threshold calibration is characterized by a very high time complexity, being actually a problem of meta-learning (i.e., a learning problem whose instances are in fact learning...
5.4 Learning Complexity of TSA

Finding the best combination of thresholds in a pipeline can be viewed as the problem to find a maximum in a function $F$ that depends on the corresponding thresholds $\bar{\theta}$:

$$\exists \bar{\theta} \in [0, 1] : F(\bar{\theta}) = \text{MAX}(F)$$  \hspace{1cm} (5.19)

It is easy to note that comprehensively trying each possible combination of thresholds (brute-force approach) will identify the very best combination. Unfortunately, this is not applicable in practice. An approximation consists of considering a finite set of points into the range $[0, 1]$ in which the thresholds could vary. In this way, considering $p$ points and a pipeline of $L$ nodes, the corresponding computational complexity is $O_{\text{greedy}}(pL)$. On the other hand, as for the TSA, its computational complexity can be calculated considering that, being $\sigma$ the average step in the range in which the thresholds can vary (i.e., $[0, 1]$), at the first step the cost of the search in the first node is $\sigma \cdot p$, at the second one is $2\sigma p$, and so on. So that, the overall complexity is $O_{\text{TSA}}(\sigma \cdot p \cdot \frac{L^2}{2})$.

As an example, let me consider $L = 4$, $\sigma = 0.5$, $p = 100$, in this case I calculate $O_{\text{greedy}} = 100,000,000$ and $O_{\text{TSA}} = 400$. Let me also note that, experimentally, the greedy approach must be applied adopting a $p$ lower than the one that can be adopted when applying TSA. In any case, also considering $p_{\text{greedy}} = 20$ and $p_{\text{TSA}} = 2000$, the corresponding computational complexities are: $O_{\text{greedy}} = 160,000$ and $O_{\text{TSA}} = \sigma \cdot 16,000$. It is easy to note that with $L = 4$, the algorithm is at least 10 times more fast with respect to the greedy one.
5. HIERARCHICAL TEXT CATEGORIZATION THROUGH A PROGRESSIVE FILTERING APPROACH
Chapter 6

A Generic MultiAgent Architecture for Information Retrieval

In the last twenty years, AI researchers have concentrated their efforts in the field of intelligent autonomous agents, i.e., systems capable of autonomous sensing, reasoning and acting in complex environments. Suitable single-agent architectures have been devised to overcome the complexity problems that arise while trying to give agents a flexible behavior (212), (213), (214), (215).

In section 2.5 I have explained that agents have been widely proposed as a solution to these problems in a huge variety of contexts.

Although generic guidelines to build general-purpose architectures are required, in my view it is important to concentrate the efforts in a specific application field in order to bridge the gap between theoretical and pragmatacal issues. To this end, in this chapter I present X.MAS, a generic multiagent architecture explicitly devoted to implement information retrieval tasks (5). The proposed architecture has been adopted in several applications. To put into evidence how to bridge the gap from theory to practice, I illustrate and discuss some relevant applications of X.MAS.

Focusing on the role of software agents, the following categories can be identified in a context of information retrieval: (i) information agents, tailored to extract and handle information while accessing information sources (216), (ii) filter agents, able to transform information according to user preferences (217), (iii) task agents, able to help...
6. A GENERIC MULTIAGENT ARCHITECTURE FOR INFORMATION RETRIEVAL

![Figure 6.1: Strip - Software Agents](image)

users to perform tasks typically in cooperation with other agents (218), (iv) interface agents, in charge of interacting with the user such that she/he interacts with other agents throughout them (219), and (v) middle agents, devised to establish communication among requesters and providers (220).

6.1 The Abstract Architecture

From a theoretical perspective, an IR task involves three main activities: (i) extracting the required information, (ii) encoding and processing it according to the specific application, and (iii) providing suitable feedback mechanisms to improve the overall performances. Figure 6.2 shows a generic architecture able to perform these activities.

The information extraction module is aimed at extracting data from information sources through specialized wrappers. In general, given an information source $S$, a specific wrapper $W_S$ must be implemented, able to map each data $D_S$, designed according to the constraints imposed by $S$, to a suitable description $O$, which contains relevant information in a structured form –such as title, author(s), description, and images.

The encoding-and-processing module is aimed at encoding information that flows from external sources (i.e., the selected information sources) and at progressively filtering it to the end user by retaining only relevant data. The actual encoding strictly depends on the specific application (pre-processing activities, such as feature selection,
6.2 The Concrete Architecture

An information retrieval system must take into account several issues, the most relevant being: (i) how to deal with different information sources and to integrate new information sources without re-writing significant parts of it, (ii) how to suitably encode data in order to put into evidence the informative content useful to discriminate among categories, (iii) how to control the unbalance between relevant and non-relevant articles (the latter being usually much more numerous than the former), (iv) how to allow the user to specify her/his preferences, and (v) how to exploit the user feedback to improve the overall performance of the system.

The user feedback module is devoted to deal with any feedback optionally provided by the end-user. In general, trivial –though effective– solutions can been implemented, e.g., solutions based on artificial neural networks (ANNs) or $k$-NN classifiers.
6. A GENERIC MULTIAGENT ARCHITECTURE FOR INFORMATION RETRIEVAL

The above problems are typically strongly interdependent in state-of-the-art systems. To better concentrate on these aspects separately, I defined a layered multiagent architecture, able to promote the decoupling among all aspects deemed relevant.

6.2.1 X.MAS Macro-Architecture

The X.MAS architecture (depicted in figure 6.3) encompasses four main levels: information, filter, task, and interface. The communication between adjacent levels is achieved through suitable middle agents, which form a corresponding mid-span level.

![Figure 6.3: The concrete architecture.](image)

At the information level, agents are entrusted with extracting data from the information sources. Each information agent is associated to one information source, playing the role of wrapper.

At the filter level, agents are aimed at selecting information deemed relevant to the users, and to cooperate to prevent information from being overloaded and/or redundant. In general, two filtering strategies can be adopted: generic and personalized. The former
applies the same rules to all users; whereas the latter is applied when a customized behavior is required for a specific user.

At the task level, agents arrange data according to users personal needs and preferences. In a sense, they can be considered as the core of the architecture. In fact, they are devoted to achieve user goals by cooperating together and adapting themselves to the changes of the underlying environment.

At the interface level, a suitable interface agent is associated with each different user interface. In fact, a user can generally interact with an application through several interfaces and devices (e.g., PC, PDA, mobile phones, etc.).

At mid-span levels, agents are aimed at establishing communication among requesters and providers. Agents at these architectural levels can be implemented as matchmakers or brokers, depending on the specific application.

### 6.2.2 X.MAS Micro-Architecture

X.MAS agents can implement several capabilities, depending on the actual application and on their specific role. From my perspective, assuming that information sources are a primary operational context for software agents, the following categories can be identified focusing on their specific role: (i) information agents, able to access to information sources and to collect and manipulate such information, (ii) filter agents, able to transform information according to user preferences, (iii) task agents, able to help users to perform tasks by solving problems and exchanging information with other agents, (iv) interface agents, in charge of interacting with the user such that she/he interacts with other agents throughout them, and (v) middle agents, devised to establish communication among requesters and providers. Although this taxonomy is focused on a quite general perspective, alternative taxonomies could be defined focusing on different features.

In particular, one may focus on capabilities rather than roles, a software agent being able to embed any subset of the following capabilities: (i) autonomy, to operate without the intervention of users; (ii) reactivity, to react to a stimulus of the underlying environment according to a stimulus/response behavior; (iii) proactiveness, to exhibit goal-directed behavior in order to satisfy a design objective; (iv) social ability, to interact with other agents according to the syntax and semantics of some selected communication language; (v) flexibility, to exhibit reactivity, proactiveness, and social
ability simultaneously (221); (vi) personalization, to personalize the behavior to fulfill user’s interests and preferences; (vii) adaptation, to adapt to the underlying environment by learning how to react and/or interact with it; (viii) cooperation, to interact with other agents in order to achieve a common goal; (ix) deliberative capability, to reason about the world model and to engage planning and negotiation, possibly in coordination with other agents; (x) mobility, to migrate from node to node in a local- or wide-area network.

X.MAS agents are JADE agents capable of (i) interacting exchanging FIPA-ACL messages, (ii) sharing a common ontology in accordance with the actual application, and (iii) exhibiting a specific behavior according to their role. As for agent internals, figure 6.4 shows the micro-architecture for agents belonging to each architectural level. Let us note that each agent encompasses a scheduler aimed at controlling the information flow between adjacent levels. Information and interface agents embody information sources and specific devices, respectively. Filter and task agents encompass an actuator that depends on the actual application. Middle agents contain a dispatcher aimed at
6.2 The Concrete Architecture

handling interactions among requesters and providers.

6.2.3 Comparing X.MAS with other MAS solution to IR

As I already said by developing X.MAS I have concentrated the efforts in the specific application field of IR in order to bridge the gap between theoretical and pragmatical issues. So, taking into account all the described requirements in an IR context, the population of software agents has been endowed with features depending on the specific functional layer they belongs to.

With regards to the systems I took as a reference model (described in section 2.5), I integrated into the architecture the most important features highlighted in the other works, with particular care to have both deployment simplicity for the final developer and a powerful expressivity.

For example, I took from NewT (70) the capability to learn user preferences and act on her/his behalf has been taken, in particular exploiting the characteristic of the kNN algorithms to run in noisy contexts (mainly due to few examples). The capabilities of Letizia (71) to browse the Web have been implemented, giving however more independence to the agents, as in SoftBots (73). Having taken into account a decentralized approach, many intuition were taken by Retsina (74), a multiagent system infrastructure applied in many domains. Retsina is an open MAS infrastructure that supports communities of heterogeneous agents where three types of agents have been defined: interface agents, able to display the information to the users; task agents, able to assist the user in the management of her/his information; and information agents, able to gather relevant information from the selected sources. In my opinion the role of encoding the information in a suitable format for the layers that come below the information one, is too crucial and complex (see section 3.3) not to be casted to a more individual and specialized level.

Furthermore the X.MAS architecture can count on a powerful set of general purpose libraries, the Java UltraUnits, capable of performing Information Retrieval tasks on all the main information sources, wrapping concepts in abstract data structures, helping with objects manipulation, and handling with operative system and graphics interfaces.
6. A GENERIC MULTIAGENT ARCHITECTURE FOR INFORMATION RETRIEVAL

6.3 Building Information Retrieval Systems by Using X.MAS

In order to highlight how to bridge the gap from theory to practice by adopting X.MAS, some relevant systems are presented: (i) NEWS.MAS, aimed at classifying news articles belonging to online newspapers [9, 12, 13]. (ii) WIKI.MAS, concerned with the problem of classifying Wikipedia contents according to a predefined set of classes [6]. (iii) MAM.MAS, focused on giving a support to professors and students while interacting with a media asset management system [5]. (iv) SEA.MAS, a MultiAgent System devoted to address the problem of monitoring boats in a marine reserve [3, 5, 222]. (v) PACMAS/BIO, built on the predecessor architecture of X.MAS: PAC is standing for personalized, adaptive, and cooperative, being these the main features that describe the specific role of each agent [223], aimed at retrieving and classifying bioinformatics publications [11, 14]. (vi) SSP (Secondary Structure Predictor), a MultiAgent System aimed at predicting secondary structures of proteins [11], (vii) PAA (Plan Acquisition Architecture), an Architecture exploiting most of the libraries of X.MAS in order to recover and translate web articles into plans [2].

6.3.1 NEWS.MAS: News Retrieval through X.MAS

6.3.1.1 The Scenario.

As well depicted in previous chapter, the continuous growth of information sources on the web, together with the corresponding volume of daily-updated contents, makes the problem of finding news and articles a challenging task. In the paper [9] I present an extension of X.MAS aimed at creating press reviews from online newspapers by progressively filtering information that flows from sources to the end user, so that only relevant articles are retained. Once extracted, newspaper articles are classified according to a hierarchical text categorization approach defined in [8]. Moreover, an optional feedback provided by the user is exploited to improve the overall performances. The system (depicted in figure 6.5) is built upon X.MAS that, as I have shown, supports the implementation of personalized, adaptive and cooperative multiagent systems devised to retrieve, filter and reorganize information in a web-based environment.

6.3.1.2 The Implementation.

To implement this specific application, X.MAS has been customized as follows:
Figure 6.5: The architecture of NEWS.MAS - A sketch of the NEWS.MAS architecture

- **Information level**, aimed at wrapping information sources. The ability of the system to deal with new information sources affects only this level (i.e., a corresponding adapter or wrapper agent must be devised and implemented for each new kind of information source to be processed);

- **Filter level**, devoted to suitably encode the text content according to an information-gain heuristics. Agents belonging to this architectural level encode (and embed) the text content of an article into a vector of words, which in turn is used to discriminate among existing categories;

- **Task level**, devoted to identify relevant articles depending on the user interests. Agents belonging to this architectural level are aimed at performing two-tiered action: first the input is classified in accordance with the existing taxonomy, then the intended category (defined by composing existing categories with *and*, *or*, and *not* operators) is used to decide whether it interests the user or not. The former action embeds suitable policies aimed at controlling the negative impact of the unbalance between relevant and non-relevant articles, whereas the latter allows
6. A GENERIC MULTIAGENT ARCHITECTURE FOR INFORMATION RETRIEVAL

the user to explicitly specify her / his preferences about the set of relevant vs. non-relevant articles;

- **Interface level**, agents belonging to this level are aimed at performing the last check in order to decide whether the given input is of interest for the user and – optionally– at providing a feedback by the user, which can be exploited to improve the overall ability of discriminating relevant from non relevant inputs.

6.3.2 WIKI.MAS: X.MAS for Classifying Wikipedia Contents

6.3.2.1 The Scenario.

As already pointed out in section 2.5, supporting users in handling the enormous and widespread amount of web information is becoming a primary issue. Currently, the most overshadowing and noteworthy web information sources are developed according to the collaborative-web paradigm (67), also known as Web 2.0. It represents a paradigm shift in the way users approach the web. Users (also called prosumers) are no longer passive consumers of published content, but become involved, implicitly and explicitly, as they cooperate by providing their own content in an “architecture of participation” (68).

![Figure 6.6: WIKI.MAS at a glance - A screenshot of the WIKI.MAS application](image-url)
Among others, Wikipedia, an online encyclopedia based on the notion that an entry can be added/edited by any web user, has became an important benchmark for all Internet users interested in searching for definitions and/or references. Unfortunately, Wikipedia search engine allows users to choose their interests among macro-areas (e.g., Arts, History, and Science), which is often inadequate to express what the user is really interested in. Moreover, such search engine does not provide a feedback mechanism able to allow the user to specify non-relevant items –with the goal of progressively adapting the system to her/his actual interests.

Using X.MAS, we developed a system explicitly aimed at retrieving and classifying Wikipedia contents according to a predefined set of classes, i.e., those belonging to WordNet Domains.

6.3.2.2 The Implementation.

To implement this specific application (see figure 6.6), X.MAS has been customized as follows:

- **Information level.** Agents at this level are aimed at dealing with the huge information source provided by Wikipedia. To this end a suitable wrapper has been implemented, able to handle the structure of a document by saving the informations about the corresponding metadata (e.g., title, abstract, keywords, section headers, see figure 6.7) and by surfing across the whole reference links through the cooperation with other information agents.

- **Filter level.** Filter agents are aimed at selecting information deemed relevant to the users, and at cooperating to prevent information from being overloaded and redundant. A suitable encoding of the text content has been enforced at this level to facilitate the work of agents belonging to the task level. In particular, all non-informative words such as prepositions, conjunctions, pronouns and very common verbs are removed using a stop-word list. After that, a standard stemming algorithm removes the most common morphological and inflexional suffixes. Then, for each category, feature selection, based on the information-gain heuristics, has been adopted to reduce the dimensionality of the feature space.

\[^1\text{http://www.wikipedia.org/}\]
6. A GENERIC MULTIAGENT ARCHITECTURE FOR INFORMATION RETRIEVAL

Figure 6.7: Wikipedia Page - The different Sections in a Wikipedia Page

- **Task level.** Task agents are devoted to identify relevant Wikipedia documents, depending on user interests. Agents belonging to this architectural level are aimed at performing two kinds of actions: classify any given input in accordance with the selected set of classes, and decide whether it may be of interest to the user or not. Each task agent has been trained by resorting to state-of-the-art algorithms that implement the $k$-NN technique, in its “weighted” variant \(^{(137)}\). Furthermore, to express what the user is really interested in, we implemented suitable composition strategies by using extended boolean models \(^{(220)}\). In fact, typically, the user is not directly concerned with “generic” topics that coincide with the selected classes (such as *Arts*, *History*, or *Science*). Rather, a set of arguments of interest can be obtained by composing these generic topics with suitable logical operators (i.e., and, or, and not). In the proposed system, we adopted a quite general soft boolean perspective, in which the combination is evaluated using $P$-norms \(^{(220)}\).

- **Interface level.** Interface agents are aimed at performing the feedback from the user –which can be exploited to improve the overall ability of discriminating
relevant from non-relevant inputs. So far, a simple solution based on the $k$-NN technology has been implemented and experimented to deal with the problem of supporting the user feedback. When a non-relevant article is evidenced by the user, it is immediately embedded in the training set of the $k$-NN classifier that implements the feedback. A check performed on this training set after inserting the negative example allows to trigger a procedure entrusted with keeping the number of negative and positive examples balanced. In particular, when the ratio between negative and positive examples exceeds a given threshold (by default set to 1.1), some examples are randomly extracted from the set of “true” positive examples and embedded in the above training set.

6.3.3 MAM.MAS: X.MAS for a Media Asset Management System

6.3.3.1 The Scenario.

E-learning differentiates from the traditional learning in its ability to train anyone, anytime, and anywhere, thanks to the openness of the Internet. Without the temporal and spatial limitation, one can have an independent and individual learning space. Currently, several Digital Asset Management (DAM) systems, also called Media Asset Management (MAM), have been proposed and devised. As for e-learning, such systems are aimed at storing, managing, and organizing course materials, bibliography, and teacher notes.

Among other provided services, MAM systems must supply suitable support during the insertion phase. In particular, classification techniques might be devised to improve such systems to suitably organize contents and to help users in managing such data.

Using X.MAS, we developed a system aimed at supporting users in inserting multimedia contents in a MAM system. Being interested in handling university courses, typically organized in a hierarchy of classes, suitable hierarchical classification techniques has been studied. In particular, the current version of the system implements a hierarchical text categorization approach and is able to deal with text documents. Let us note that any multimedia document could be processed if a suitable textual description is given. In our experiments the system classifies data according to the taxonomy related to university courses, i.e., the one concerned with the bachelor’s degree in electronic engineering. Figure 6.8 illustrates a portion of the adopted taxonomy.
6.3.3.2 The Implementation.

As for the implementation, X.MAS has been customized as follows:

- **Information level.** Input documents are the files provided by teachers during the insertion phase. Such documents can be simple-text-, formatted-, pdf-files, or multimedia contents together with their textual description. Information agents are able to handle all these kinds of documents by extracting the corresponding information.

- **Filter level.** As in WIKIMAS, filter agents are aimed at selecting information deemed relevant to the users, and to cooperate to prevent information from being overloaded and redundant. Suitable encoding techniques have been enforced: all non-informative words are removed using a stop-word list; a standard stemming algorithm removes the most common morphological and inflexional suffixes; and, for each category, feature selection, based on the information-gain heuristics, has been adopted to reduce the dimensionality of the feature space.

- **Task level.** Task agents perform the hierarchical text categorization, resorting to a progressive filtering technique as described in [228]. In particular, each task agent has been trained by resorting to state-of-the-art algorithms that implement the
6.3 Building Information Retrieval Systems by Using X.MAS

wk-NN technique, whereas the progressive filtering is provided by the cooperation of the corresponding agents.

• Interface level. Interface agents are devoted to interact with the user within the MAM in order to support her/him while inserting documents. Further agents are aimed at performing user feedback. So far, a simple solution based on an ANN has been implemented. This solution consists of training an ANN with a set of examples classified as “of interest to the user”. When the amount of feedback provided by the user has trespassed a given threshold, the ANN is trained again—after updating the previous training set with the information provided by the user.

6.3.4 SEA.MAS: X.MAS for Monitoring Boats in Marine Reserves

6.3.4.1 The Scenario.

In the summertime, in Sardinia and in its small archipelago, tourists sometimes sail in protected or forbidden areas close to the coast. Monitoring such areas with the goal of discriminating between authorized and unauthorized boats, is quite complicated. In fact, along Sardinian coasts, there are two-hundred tourist harbors with about thirteen thousand places available for boats and several services for boat owners. Monitoring large areas without suitable resources (such as radars) can be highly uneconomic, since staff operators would be (and typically are) compelled to directly patrol them over time. A typical solution consists of using a radar system controlled by a central unit located ashore in a strategical position. Radar signals allow to detect the positions of the boats that sail in the controlled area.

With the goal of monitoring and signaling intrusion in marine reserves, we experimented a multiagent solution in which authorized boats are equipped with suitable devices able to transmit (through the GSM technology) their position (through the GPS technology). In this way, the corresponding scenario encompasses two kinds of boats: authorized, recognizable by the GPS+GSM devices, and unauthorized. Both kinds of boats are expected to be identified by a digital radar able to detect their position in the protected area. Comparing the positions sent by boats with those detected by the radar allows to identify unauthorized boats.
6. A GENERIC MULTIAGENT ARCHITECTURE FOR INFORMATION RETRIEVAL

6.3.4.2 The Implementation.

The multiagent system aimed at monitoring boats in marine reserves described in this chapter has been called SEA.MAS, to highlight the fact that it stems from X.MAS. The adoption of X.MAS comes from the fact that the problem of monitoring and signaling intrusions in marine reserves can be seen as a particular information retrieval task, in which radar and GPS+GSM devices are information sources, whereas authorized and unauthorized boats are categories to be discriminated.

The first step for customizing X.MAS to a specific application consists of extending each abstract class with the goal of providing the required features and capabilities. Let us summarize them level by level:

- **Information level.** In SEA.MAS, information sources are the digital radar and GPS+GSM devices. For each information source, a suitable information agent has been devised to embody the information provided therein. In particular, we implemented a wrapper for the digital radar and a wrapper for the GPS+GSM devices (see figure 6.9).
invoking the middle agent corresponding to the middle-span level “Information-Filter”.

- **Filter level.** In SEA.MAS, filter agents are aimed at encoding the information extracted by the information agents. The encoding activity consists of creating events containing the position of the detected boats and their identification code, when available. Moreover, filter agents are devoted to avoid two kinds of redundancy: information detected more than once from the same device (caching) or throughout different devices (information overloading). Then, the middle agent corresponding to the middle-span level “Filter-Task” forwards the event to the corresponding task agent, assuming the role of yellow pages if the identification code is available or the role of broker otherwise. If a detected event is not related to an authorized boat, the middle agent creates a task agent able to handle the event.

- **Task level.** In SEA.MAS, a task agent is created for each boat, the underlying motivation being the need to centralize the knowledge regarding the position of a boat and its state. As for the position, events are classified as belonging either to anonymous sources or to known sources. For known sources the state reports their identification code and –when available– further information (i.e., a description of the boat and/or owner’s data). The main tasks of the agents belonging to this level are: (i) to follow a boat position during its navigation, also dealing with any temporary lack of signal; (ii) to promptly alerting the interface agents in the event that unauthorized boats are identified; and (iii) to handle messages coming from the interface level (e.g., false alarm notification).

- **Interface level.** In SEA.MAS, suitable interface agents allow the system administrator and staff operators to interact with the system. In both cases, the corresponding interface agent is aimed at getting a feedback from the user, for instance to inform relevant agents about changes occurred in the environment or about faults that might occur in devices located on the authorized boats. User feedback can also be used to improve the overall ability of discriminating among authorized and unauthorized boats. Currently, this kind of user feedback is performed through a simple solution based on the k-NN technology (described in section
When either a false positive or a false negative is evidenced by the user, it is immediately embedded in the training set of the k-NN classifier that implements the feedback.

6.3.5 PACMAS/BIO: X.MAS for Classifying Bioinformatic Publications

6.3.5.1 The Scenario.

A growing amount of information is currently being generated and stored in the World Wide Web, so that—at least in principle—researchers can easily find relevant publications and scientific literature. However, especially for beginners, it is still very hard to determine which papers are of interest without an explicit classification of the topics they are involved in. To solve this problem, effective information filtering techniques are primary features to be provided. In fact, beyond conventional search engines, users need specific tools and methods for using all available scientific resources. To this end, a technique for classifying bioinformatics publications according to a taxonomic domain knowledge is proposed and the corresponding system built upon PACMAS, the predecessor of X.MAS [14]. To evaluate the performance of the system, tests have been performed with publications extracted from the BMC Bioinformatics site and from the PubMed Central digital archive.

6.3.5.2 The Implementation.

As for the implementation, X.MAS is being customized as follows:

- **Information level.** Three kinds of agents have been implemented for PACMAS/BIO: an agent that wraps an RSS web site (BMC Bioinformatics), an agent that wraps a Web Service (PubMed Central), and an agent that wraps the adopted taxonomy (the TAMBIS ontology, see figure 6.10).

- **Filter level.** In PACMAS/BIO a suitable encoding of the text content has been enforced at the filter level to facilitate the work of agents belonging to the task level. In particular, all non-informative words such as prepositions, conjunctions, pronouns and very common verbs are removed using a stop-word list. After that, the stemming algorithm of Porter (see section 3.3.1) removes the most common...
6.3 Building Information Retrieval Systems by Using X.MAS

morphological and inflexional suffixes. Then, for each category of the taxonomy, feature selection, based on the information-gain (see section 3.3.2) heuristics, has been adopted to reduce the dimensionality of the feature space.

- **Task level.** In PACMAS/BIO, a classifier for each item in the taxonomy has been implemented by resorting to the k-NN technique, in its “weighted” variant (presented in section 3.3.4). The motivation for the adoption of this particular technique stems from the fact that it does not require specific training and it is very robust with respect to noisy data. Furthermore, the adoption of the wk-NN variant is related with the choice of P-norms as in WIKI.MAS, for implementing the “and” operation, as P-norms combinations rules require values in [0,1]. Task agents are trained in order to recognize a specific class, each class being an item of the adopted taxonomy.

Task agents are also devoted to measure the classification accuracy according to the confusion matrix (229).

- **Interface level.** At the interface level, agents and users interact through a suitable
6. A GENERIC MULTIAGENT ARCHITECTURE FOR INFORMATION RETRIEVAL

A generic multiagent architecture for information retrieval involves a suitable agent associated with each different user interface. In fact, a user can generally interact with an application through several interfaces and devices (e.g., PCs, PDAs, mobile phones). Interface agents are also devoted to handle user profile and to propagate it by the intervention of middle agents.

6.3.6 SSP: X.MAS for Protein Secondary Structure Prediction

6.3.6.1 The Scenario.

In recent years, there has been increasing interest in decentralized approaches for solving complex real-world problems, many of them falling into the area of cooperative multiagent learning (230). In principle, two main approaches to learning in multiagent systems exist: centralized and decentralized. The former consists of implementing the learning process by a single agent, without any interaction with other agents. The latter is related to situations in which several agents are engaged in the same learning process and cooperate to perform the learning task. Most of the work in this field use reward-based methods, such as reinforcement learning (231, 232) and stochastic search methods, in particular evolutionary computation approaches (233, 234).

This work has been devised to allow researchers make experiments on complex tasks while resorting to evolutionary computation strategies. The corresponding multiagent system encompasses one or more populations of experts running at the same time, all being capable after training of collaborating to perform the task of predict Protein Secondary Structure (10).

In biochemistry and structural biology, secondary structure is the general three-dimensional form of local segments of biopolymers such as proteins and nucleic acids (DNA/RNA). It does not, however, describe specific atomic positions in three-dimensional space, which are considered to be tertiary structure. Predicting protein tertiary structure is essential to understand the functionality of a proteins. This is a very challenging problem from only its amino acid sequence, but using the simpler secondary structure definitions is more tractable and has been the focus for research for a long time.
6.3.6.2 The Implementation.

To perform the goal, different populations of agents are involved in specific coordination sub-tasks (in particular, handling training and tests, encoding inputs, and combining outputs). It is worth pointing out that the only interaction between experts that belong to different populations occurs during the output combination process, whereas for any other aspect different populations evolve separately in accordance with the selected training strategy, internal architecture, and input encoding.

![Figure 6.11: Typical implementation of the proposed generic multiagent architecture](image)

Figure 6.11 highlights a typical snapshot of the generic multiagent architecture, showing two populations of experts, together with the main corresponding coordination agents that occur during the training process, namely: (i) training handlers, entrusted with training the populations, (ii) encoders, entrusted with coding the occurring inputs according to the specific requirements imposed by the selected application domain, and (iii) output managers, entrusted with performing output combination. During tests, training handlers are substituted by test handlers, which are entrusted with benchmarking the populations against known data sets while collecting and storing results for subsequent analysis.
As described in (235), a typical population is composed of hybrid experts, each encompassing a “genetic” and a “neural” part. To be more precise, each expert can be modeled as a triple $\Gamma = (g, h, w)$, where $g$ (the guard) maps a subset of the input to \{false, true\}, $h$ (the embedded expert) is a classifier or predictor trained to recognize a particular configuration of the input space, and $w$ is a weighting function, used while performing output combination. Hence, $\Gamma$ coincides with $h(x)$ for any input $x$ that matches $g$, otherwise it is not defined meaning that the corresponding expert has not been activated by the current input (see figure 6.12). Furthermore, an expert $\Gamma$ contributes to the final prediction according to the value $w(x)$ of its weighting function, which represents the expert strength in the output combination process.

Experimental results, performed by training 400 experts on a set of 5K proteins characterized by a degree of pairwise similarity under the threshold of 25%, reach a Q3 (i.e., the overall accuracy) of about 75%, which is comparable with the one exhibited by other state-of-the-art predictors.
6.3 Building Information Retrieval Systems by Using X.MAS

6.3.7 PAA: X.MAS for Recovering Plans from the Web

6.3.7.1 The Scenario.

Automated Planning (AP) is an AI field whose goal is to automatically generate sequence of actions that solve problems. One of the main difficulties in its extensive use in real-world application lies in the fact that it requires the careful and error-prone process of defining a declarative domain model. This is usually performed by planning experts who should know about both the domain in hand, and the planning techniques (including sometimes the inners of these techniques or the tools that implement them).

In order planning to be widely used, this process should be performed by non-planning experts. On the other hand, in many domains there are plenty of electronic documents (including the Web) that describe processes or plans in a semi-structured way. These descriptions mix natural language and certain templates for that specific domain. One such examples is the www.WikiHow.com website (see figure 6.13) that includes plans in many domains, all plans described through a set of common templates.
In [2], we present a suite of tools that automatically extract knowledge from those unstructured descriptions of plans to be used for diverse planning applications.

6.3.7.2 The Implementation.

Most of the libraries and the distributed computing capabilities rely on the X.MAS framework (see figure 6.14), in particular:

- The Information level (namely the Crawler subsystem) is devoted to crawl and store pages and category sections of a web site. It can find an article by queries in natural language, and archive into the system all the articles belonging to a specific category.

- The Filter Level (namely the Page Processor subsystem) is devoted to deal with a web page extracting the contents deemed important for AP purposes; its tools are devised to facilitate the recognition of sections, together with the contents type. The aim of its tools is (i) to process a given HowTo article in order to extract the useful information in terms of semistructured relations. It also keeps some information about the page structure and the raw contents for further potential processing purposes; (ii) to integrate semantic tools, necessary to build an augmented plan in the form of predicates containing unstructured components. Exploiting the entire knowledge base, this information will be used to build plans.
• The Task Level (namely the Plan Acquisition subsystem) includes tools that allow to create plans from web pages and to build new plans. The tools belonging to the Plan Acquisition subsystem are: (i) the PlanBuilder which aim is to build plans in predicate logic, and (ii) the AnalysisTool that embodies a suite of statistical tools.

• The Interface Level embodied all the interfaces deemed at querying the system by its users, and administrators. Currently an application panel and a web service integrating all the tools of the architecture were devised.

The experiments shown that the system performed rather well on plan extraction (above 68% of accuracy), considering the complexity of the semantic analysis tasks and the need to handle many outliers.
6. A GENERIC MULTIAGENT ARCHITECTURE FOR INFORMATION RETRIEVAL
Chapter 7

Experimental Results

Experiments have been performed to validate the proposed approach with respect to the impact of PFA in the input imbalance as well as the performance improvement of adopting TSA with respect to adopting a greedy approach.

To validate the impact of the proposed approach in the input imbalance three series of experiments have been performed: first, performances calculated resorting to PFA have been compared with the ones calculated resorting to the corresponding flat approach. Subsequently, the approach has been tested to assess the theoretical results concerning with the improvement of performances while augmenting the pipeline depth. Finally, performances have been calculated in terms of generalization-, specialization-, misclassification-, and unknown-error, according to (175).

![Sex of Technology](image)

**Figure 7.1: Strip - The Sex of Technology**
7. EXPERIMENTAL RESULTS

To validate the algorithm efficiency, the performances calculated adopting the proposed threshold-selection algorithm have been compared with the ones corresponding to apply an uniformed greedy algorithm able to select the best combination of thresholds into a finite set of combinations.

Experiments have been performed by using: (i) a parallel of 3 Blade Dell, each of them with a double Quad-core CPU Intel Xeon E5450, 3Ghz with a 1333MHz BUS and 16 Gb RAM; (ii) a SUN Workstation with two Opteron 280, 2Ghz+ and 8Gb Ram; and (iii) a cluster of 3 Acer Mini PC Intel Core Duo E2140, 1.6Ghz with 2Gb RAM.

7.1 The Adopted Datasets

Existing document collections suffer from one or more of the following drawbacks: (i) few documents, (ii) lack of the document full text, (iii) inconsistent or incomplete category assignments, (iv) peculiar textual properties, and (v) limited availability. Furthermore, typically, researchers do not have documentation on how collections were produced, and on the nature of the underlying categories. These problems are particularly severe in hierarchical text categorization, where researchers often impose their own hierarchies.

Until the mid-1990s researchers mostly ignored the hierarchical structure of categories that occur in several domains. Researchers, typically, performed their experiments by using ad-hoc corpora (89; 140; 236) making comparisons extremely difficult and unreliable. Therefore it is a straightforward step to perform experiments on new collections (81). In particular, several proposals of hierarchical methods for text classification have been made using the Reuters standard document collection, along with the definition of suitable class hierarchies.

The two corpora chosen for this study are the benchmark datasets Reuters Corpus Volume I (RCV1-v2) (176) and DMOZ, the collection of HTML documents referenced in a web directory developed in the Open Directory Project (ODP). The corpora differ considerably in the training set size, in the hierarchical structure of categories, as well as in the procedure adopted for the classification of documents. For the sake of completeness, a brief description of such collections is reported in the following.
7.1 The Adopted Datasets

7.1.1 Reuters

Reuters, Ltd.\footnote{http://www.reuters.com} is a leading global provider of financial information, news and technology to financial institutions, the media, businesses and individuals. The Reuters Corpus Volume I (RCV1) \footnote{http://www.reuters.com} is an archive of over 800,000 manually categorized newswire stories made freely available by Reuters, Ltd. for research purposes.

Being interested in hierarchical text categorization, in this thesis work I adopt the second version of the Reuters Corpus dataset (RCV1-v2) \footnote{http://www.reuters.com}. In this corpus, stories have been coded into four hierarchical groups: Corporate/Industrial (CCAT), Economics (ECAT), Government/Social (GCAT) and Markets (MCAT). Those sets were designed originally around requirements of business information professionals, although this was broadened to include the needs of end users in large corporates, banks, consultancy, marketing and advertising firms, and financial services. The complete list consists of 126 codes. However, not all of these were used in the coding phase. In particular, the codes labeled as “current news” and those marked as “temporary” are unused, as well as further 10 codes. Therefore, the total number of codes actually assigned to the data is 103.

In this thesis, the level-1 categories have been created giving rise to 13 categories. The corresponding nodes have been populated labeling the union of the corresponding \( k \) children of level-2 classes: \( C_n = \bigcup(C_{nx}), x \in [1..k] \).

Summarizing, removing classes not exploitable for pipeline purposes –e.g., political parties (GPOL), sports stories (GSPO), weather conditions (GWEA)– and adding the level-1 categories, in my experiments I adopt a set of 93 categories. Every node is then labeled concatenating a letter that corresponds to the root (i.e., E, C, G, and M), with a sequence of digits that take into account the path into the taxonomy (figure 7.2 sketches an example of pipeline belonging to the Corporate/Industrial code).

7.1.2 DMOZ

The Open Directory Project (ODP) also known as DMOZ\footnote{http://www.dmoz.org} is a multilingual open content directory of World Wide Web links owned by Netscape (see figure 7.3), constructed and maintained by a community of volunteer editors.
7. EXPERIMENTAL RESULTS

Figure 7.2: An example of pipeline in the Reuters corpus.

The purpose of this project is to provide a valuable alternative to automated search engines that are increasingly unable to turn up useful results to search queries. ODP uses a hierarchical ontology scheme for organizing site listings. Listings on a similar topic are grouped into categories, which can then include smaller categories.

The original categories were for Adult, Arts, Business, Computers, Games, Health, Home, News, Recreation, Reference, Regional, Science, Shopping, Society, and Sports. While these fifteen top-level categories have remained intact, the ontology of second- and lower-level categories has undergone a gradual evolution.

In this work, I selected 17 categories and considered pipelines depths up to 6 (see figure 7.4 for an extract of the DMOZ taxonomy).

Since with the Reuters corpus it is not feasible to use 6-depth pipelines, adopting DMOZ allows me to investigate more in details the impact of PFA also considering its limitations with a high number of classifiers per pipeline. On the contrary, unfortunately, the number of documents belonging to the leaf categories becomes spare giving rise to some perturbation in the results.
7.2 Evaluation Measures

To evaluate the performances of PFA several measures have been considered.

First, I am interested in compare the performance of the proposed approach with respect to the corresponding flat one. To this end the approach is validated resorting to a conventional metric for calculating the performances of a text categorization system: macro-averaging. Macro-averaged scores are calculated by first calculating precision and recall for each category and then taking their average. With respect to the micro-averaging measure that gives equal weight to every document (document-pivoted measure), macro-averaging gives equal weight to every category (category-pivoted measure) \(^{40}\). In other words, as already pointed out in 2.3.2 macro-averaging first evaluates \(P\) and \(R\) “locally” for each category, and then “globally” by averaging over the results of the different categories.

Furthermore, to evaluate the performances of the approach, I resort to \(F_1\), obtained from the more general definition of \(F_\beta\) by imposing that \(P\) and \(R\) have the same degree of importance.

Then, the approach is validated adopting a suitable set of measures defined to test
7. EXPERIMENTAL RESULTS

a hierarchical approach. To do this, I resort to evaluation measures, defined in (175) and discussed in section 4.5.

7.3 Implementation Details

Experiments have been performed by customizing to this specific task X.MAS, the generic multiagent architecture depicted in chapter 6.

For the purpose of this thesis work the X.MAS architecture has been customized as follows:

At the information level, agents extract data from the information sources. To this end, two wrappers have been implemented: one to embed the Reuters dataset, and one the DMOZ.

At the filter level, agents select information deemed relevant to the final user, as well as for the training and test phase. These agents perform the encoding of the information, exploiting algorithms such as information gain (see 3.3.2).

At the task level, agents arrange data according to the training/test needs. Cooperating together and adapting themselves to the changes of the underlying environment, task agents perform the HTC tasks.

At the interface level, a suitable interface agent has been implemented to perform and control different training and test phases of the process.

Adopting X.MAS allows to easily distribute the computational load, as well as to limit the communication overhead and, thanks to the software-agent persistence, to avoid classical problems due to hardware maintenance.

7.4 Results

7.4.1 Evaluating the Impact of the Approach in the Input Imbalance

The main question I am interested in is the effectiveness of the proposed approach with respect to flat classification. In order to make a fair comparison, the same classification system has been adopted, i.e., a classifier based on the \( wk \)-NN technology. In particular the distance weighted function \( w_i = \frac{1}{j} \) (where \( i \) is the ordinal number of the prototype in the list of the nearest) suggested in (158) has been used, having it shown a good behavior in previous tests.
First, each classifier is trained with a balanced data set of up to 1000 documents by using 200 (tf*idf) features selected according to the information gain method. Then, the best thresholds are selected. To make a fair comparison, both the thresholds of the pipelines and the flat classifiers have been chosen considering the $F_1$ measure as utility function.\footnote{The actual utility function can be chosen according to the constraint imposed by the given scenario. For instance, $F_1$ is suitable if one wants to give equal importance to precision and recall.}

As for the pipelines, the automated mechanism described in the chapter has been adopted. During the calibration / repair phase a step of $10^{-4}$ has been used by TSA to increase (decrease) the threshold.

Experiments have been conducted by assessing the behavior of the proposed hierarchical approach in presence of different percentage of positive examples versus negative examples, i.e., from $2^{-1}$ to $2^{-7}$.

**PFA vs Flat Classification.** To assess the effectiveness of the approach with respect to the corresponding flat approach, I calculated macro-averaging precision and recall. To better put into evidence the importance of the pipeline depth, for PFA I considered only pipelines that end with a leaf node, whereas for the flat approach I considered only classifiers that correspond to a leaf.

Figures 7.5, 7.6 and 7.7, 7.8 show macro-averaging of precision and recall for Reuters and DMOZ dataset, respectively. Precision and recall have been calculated both for the flat classifiers and the pipelines varying the input imbalance. Results show that, as for precision, the distributed solution based on pipelines has reported better results than those obtained with flat the model. On the contrary, as expected, results on recall are worse than those obtained with the flat model.

To better highlight the overall results, figure 7.9 and Table 7.1 show the performance improvements on Reuters dataset of the proposed approach with respect to the flat one. The improvement has been calculated in terms of $F_1$, for different values of imbalance between positive and negative examples and for pipelines of different lengths.

In these experiments the imbalance (positive percentage over all documents) grows exponentially based on power of 2 (i.e., $2^{-1}$, $2^{-2}$, $2^{-7}$ of positive). The performance improvement is calculated in percentage with the formula $(F_1(\text{pipeline}) - F_1(\text{flat}))\%$.\footnote{The actual utility function can be chosen according to the constraint imposed by the given scenario. For instance, $F_1$ is suitable if one wants to give equal importance to precision and recall.}
### 7. EXPERIMENTAL RESULTS

<table>
<thead>
<tr>
<th>imbalance</th>
<th>depth</th>
<th>pipeline</th>
<th>flat</th>
<th>improvement (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$2^{-1}$</td>
<td>2</td>
<td>0.807</td>
<td>0.800</td>
<td>0.768</td>
</tr>
<tr>
<td>$2^{-1}$</td>
<td>3</td>
<td>0.787</td>
<td>0.781</td>
<td>0.593</td>
</tr>
<tr>
<td>$2^{-1}$</td>
<td>4</td>
<td>0.909</td>
<td>0.903</td>
<td>0.580</td>
</tr>
<tr>
<td>$2^{-3}$</td>
<td>2</td>
<td>0.462</td>
<td>0.452</td>
<td>0.979</td>
</tr>
<tr>
<td>$2^{-3}$</td>
<td>3</td>
<td>0.494</td>
<td>0.471</td>
<td>2.330</td>
</tr>
<tr>
<td>$2^{-3}$</td>
<td>4</td>
<td>0.658</td>
<td>0.633</td>
<td>2.542</td>
</tr>
<tr>
<td>$2^{-5}$</td>
<td>2</td>
<td>0.242</td>
<td>0.233</td>
<td>0.922</td>
</tr>
<tr>
<td>$2^{-5}$</td>
<td>3</td>
<td>0.262</td>
<td>0.230</td>
<td>3.261</td>
</tr>
<tr>
<td>$2^{-5}$</td>
<td>4</td>
<td>0.375</td>
<td>0.329</td>
<td>4.581</td>
</tr>
</tbody>
</table>

Figure 7.10 and Table 7.2 show the performance improvements on DMOZ dataset. As previously noted, these results are subject to perturbations due to the limited amount of examples, in fact, for several leaf categories the number of positive examples is very low (e.g., 12). Nevertheless, PFA performs always better that the corresponding flat.

Summarizing, experimental results –having the adopted taxonomies a maximum depth of six– show that the proposed approach performs always better than the flat one, the filtering effect of a pipeline being not negligible.

**Hierarchical Metrics.** The overall error in terms of generalization -, specialization -, misclassification -, and unknown- error has been calculated. Let me note that this results strictly depend on the adopted utility function, in fact changing the utility function the confusion matrix of the pipeline changes accordingly.

Figure 7.11 and 7.12 depict the results obtained varying the imbalance on Reuters and on DMOZ dataset, respectively.

Analyzing the results it is easy to note that, as expected, the generalization- and the misclassification-error grow with the imbalance, whereas the specialization- and the unknown-error decrease.

As for the generalization-error, being the percentage of documents misclassified into a supercategory of the correct one, it depends on the overall number of FNs. As we already noted, the greater the imbalance, the greater the amount of FNs, so that the
Table 7.2: Performance improvement on DMOZ dataset

<table>
<thead>
<tr>
<th>imbalance</th>
<th>depth</th>
<th>pipeline</th>
<th>flat</th>
<th>improvement (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$2^{-1}$</td>
<td>3</td>
<td>0.775</td>
<td>0.764</td>
<td>1.077</td>
</tr>
<tr>
<td>$2^{-1}$</td>
<td>4</td>
<td>0.859</td>
<td>0.850</td>
<td>0.836</td>
</tr>
<tr>
<td>$2^{-1}$</td>
<td>5</td>
<td>0.827</td>
<td>0.777</td>
<td>4.982</td>
</tr>
<tr>
<td>$2^{-1}$</td>
<td>6</td>
<td>0.852</td>
<td>0.819</td>
<td>3.247</td>
</tr>
<tr>
<td>$2^{-3}$</td>
<td>3</td>
<td>0.434</td>
<td>0.428</td>
<td>0.657</td>
</tr>
<tr>
<td>$2^{-3}$</td>
<td>4</td>
<td>0.593</td>
<td>0.579</td>
<td>1.436</td>
</tr>
<tr>
<td>$2^{-3}$</td>
<td>5</td>
<td>0.495</td>
<td>0.412</td>
<td>6.388</td>
</tr>
<tr>
<td>$2^{-3}$</td>
<td>6</td>
<td>0.584</td>
<td>0.515</td>
<td>6.907</td>
</tr>
<tr>
<td>$2^{-5}$</td>
<td>3</td>
<td>0.212</td>
<td>0.202</td>
<td>0.937</td>
</tr>
<tr>
<td>$2^{-5}$</td>
<td>4</td>
<td>0.315</td>
<td>0.279</td>
<td>3.639</td>
</tr>
<tr>
<td>$2^{-5}$</td>
<td>5</td>
<td>0.254</td>
<td>0.195</td>
<td>5.844</td>
</tr>
<tr>
<td>$2^{-5}$</td>
<td>6</td>
<td>0.363</td>
<td>0.276</td>
<td>7.732</td>
</tr>
</tbody>
</table>

generalization-error increases with the imbalance. In a certain sense, we can state that, in presence of input imbalance, the trend of the generalization-error is similar to the trend of the recall.

As for the specialization-error, being the percentage of documents misclassified into a subcategory of the correct one, it depends on the overall number of FPs. As we already noted, the greater the imbalance, the lower the amount of FPs, so that the specialization-error decreases with the imbalance. In a certain sense, we can state that, in presence of input imbalance, the trend of the specialization-error is similar to the trend of the precision.

As for the unknown- and misclassification-error, let me first note that an imbalance of positive and negative examples can be suitably dealt with by exploiting the filtering effect of classifiers in the pipeline. So that, the classifiers belonging to the toppest levels of the pipeline can decide to reduce the overall number of FNs accepting as positive, documents that actually are not. In this way, the unknown-error, which depends on the amount of FNs, decreases, whereas the misclassification-error, which depends on the amount of FPs, increases.
7. EXPERIMENTAL RESULTS

Table 7.3: TSA vs greedy

<table>
<thead>
<tr>
<th>step</th>
<th>time s</th>
<th>time h</th>
<th>time s</th>
<th>time h</th>
<th>ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>step1</td>
<td>10333</td>
<td>2.87</td>
<td>382</td>
<td>0.11</td>
<td>27.02</td>
</tr>
<tr>
<td>step2</td>
<td>97907</td>
<td>27.20</td>
<td>3497</td>
<td>0.97</td>
<td>28</td>
</tr>
<tr>
<td>step3</td>
<td>213724</td>
<td>59.37</td>
<td>7610</td>
<td>2.11</td>
<td>28.08</td>
</tr>
<tr>
<td>step4</td>
<td>328372</td>
<td>91.21</td>
<td>11634</td>
<td>3.23</td>
<td>28.22</td>
</tr>
</tbody>
</table>

Table 7.4: $F_1$ in presence of input imbalance

<table>
<thead>
<tr>
<th>Input imbalance</th>
<th>Greedy algorithm</th>
<th>Threshold-selection algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td>$2^{-1}$</td>
<td>0.830</td>
<td>0.835</td>
</tr>
<tr>
<td>$2^{-2}$</td>
<td>0.722</td>
<td>0.733</td>
</tr>
<tr>
<td>$2^{-3}$</td>
<td>0.619</td>
<td>0.632</td>
</tr>
<tr>
<td>$2^{-4}$</td>
<td>0.497</td>
<td>0.515</td>
</tr>
<tr>
<td>$2^{-5}$</td>
<td>0.404</td>
<td>0.428</td>
</tr>
<tr>
<td>$2^{-6}$</td>
<td>0.323</td>
<td>0.345</td>
</tr>
<tr>
<td>$2^{-7}$</td>
<td>0.245</td>
<td>0.273</td>
</tr>
</tbody>
</table>

7.4.2 Threshold-Selection Algorithm vs a Greedy Approach

As for the computational effectiveness of the proposed threshold-selection algorithm with respect to a greedy approach, experimental results confirm the theoretical ones presented in 5.4.

Experiments have been performed on 24 pipelines with depth $L = 4$. A step $\Delta_{TSA} = \frac{1}{p_{TSA}} = 0.0005$ has been adopted to increment the threshold in TSA; whereas a step $\Delta_{greedy} = \frac{1}{p_{greedy}} = 0.05$ has been adopted for the greedy approach for computational reasons. Table 7.3 illustrates the results comparing, for each unfolding step, the time spent (in seconds and hours) by the greedy algorithm and TSA. The last column puts into evidence the ratio between the two approaches.

Furthermore, I compared results in terms of $F_1$ considering different imbalances in the input. Table 7.4 summarizes the results. As you can note, applying the threshold-selection algorithm the performances are always better than applying the greedy one.
Figure 7.4: An extract of the DMOZ taxonomy.
7. EXPERIMENTAL RESULTS

Figure 7.5: Comparison of precision in the Reuters dataset

Figure 7.6: Comparison of recall in the Reuters dataset
7.4 Results

Figure 7.7: Comparison of precision in DMOZ dataset

Figure 7.8: Comparison of recall in DMOZ dataset
7. EXPERIMENTAL RESULTS

Figure 7.9: Performance improvement on Reuters dataset

Figure 7.10: Performance improvement on DMOZ dataset
7.4 Results

Figure 7.11: Hierarchical measures on Reuters dataset

Figure 7.12: Hierarchical measures on DMOZ dataset
7. EXPERIMENTAL RESULTS
Chapter 8

Conclusions and Future Directions

This thesis work has presented the study of novel techniques for Hierarchical Text Categorization and has experimentally demonstrated the validity of the approach. Experiments have been performed with the help of an architecture explicitly devised for dealing with Information Retrieval tasks.

In the first part of the thesis, the main advantage of the hierarchical perspective has been presented and the theoretical basis of this field have been analyzed. This complex scenario has been described in the first part of my PhD thesis, spread along the first three chapters, aimed at depicting the theory of Hierarchical Text Categorization. In particular, chapter 2 shows the impact of Information Retrieval in the current technology, together with the importance that it holds in everyday life. Chapter 3 reports a survey on Text Categorization, with the goal of highlighting the most relevant algorithms and techniques useful to understand the emerging field of Hierarchical Text Categorization (chapter 4) –which this thesis is focused on.

The second part of my PhD thesis introduces a novel approach to Hierarchical Text Categorization based on Progressive Filtering (PFA).

In Chapter 5, PFA is defined and then a comprehensive study about its impact on Text Categorization is made. PFA typically requires the adoption of a threshold selection algorithm (TSA), aimed at optimizing the performance of any system built in accordance with its guidelines. A TSA algorithm has been devised and implemented, taking into account its computational complexity. Furthermore, given the granularity
of the search (i.e., the search step), the performance of the proposed TSA algorithm has been compared with a brute-force approach.

In Chapter 6 a generic architecture, called X.MAS, based on software agents has been devised and implemented with the goal of making easier the implementation of actual systems aimed at solving Information Retrieval tasks. Several case studies are presented to highlight the potential of X.MAS.

X.MAS has also been customized to perform experimental comparisons between PFA and the flat approach, focusing on their capability of contrasting the imbalance between positive and negative examples. Experiments (chapter 7), performed on Reuters and DMOZ, clearly show that PFA is able to counteract high imbalances.

As for future directions: (i) from a research perspective, the whole taxonomy could be investigate instead of the corresponding set of pipelines; (ii) from a benchmarking perspective, further metrics to calculate the performances of PFA could be adopted and experimented; and (iii) from an experimental perspective, PFA could be tested on other datasets, such as TREC or MeSH.
References


REFERENCES


REFERENCES


REFERENCES
REFERENCES


REFERENCES


[163] David D. Lewis. The TREC-4 Filtering Track. In TREC, 1995. 56


129


J. Platt.

W. Wibowo and H. Williams.

P.G. Ipeirotis, L. Gravano, and M. Sahami.


REFERENCES


Xin Li and Dan Roth. Learning Question Classifiers In COLING, 2002.


[REFERENCES]


[232] CLAUS SKAANNING. [Rao-Robins Sampling for Inference in Large and Complex Bayesian Networks with Applications in Genetics] Aalborg University, Aalborg, 1997. [99]


Declaration

I herewith declare that I have produced this paper without the prohibited assistance of third parties and without making use of aids other than those specified; notions taken over directly or indirectly from other sources have been identified as such. This paper has not previously been presented in identical or similar form to any examination board.

The thesis work was conducted from November 2006 to November 2009; this document was completed in January 2010.

Cagliari, January 10th, 2010

Andrea Addis