AUTOMATIC WEB RESOURCE COMPILATION USING DATA MINING

by

Nuno Filipe Fonseca Vasconcelos Escudeiro

Tese de Mestrado em Análise de Dados e Sistemas de Apoio à Decisão

MSc Thesis on Data Analysis and Decision Support Systems

Supervised by

Dr Alípio Jorge

Faculdade de Economia

Universidade do Porto

2004

AUTOMATIC WEB RESOURCE COMPILATION USING DATA MINING

by

Nuno Filipe Fonseca Vasconcelos Escudeiro

Tese de Mestrado em Análise de Dados e Sistemas de Apoio à Decisão

MSc Thesis on Data Analysis and Decision Support Systems

Supervised by

Dr Alípio Jorge

Faculdade de Economia

Universidade do Porto

2004

004 (043) / ESCm / AUT

Universidade do Porto
Faculdade de Engenharia
Biblioteca
Nº 93563
CDU
Data 02 / 01 / 2008

Acknowledgements

My thanks to Alípio Jorge, my supervisor, for the advice, encouragement and friendship he was always available to give.

My thanks to Paula, my wife, and Francisca, my daughter, for their support and for the time I took them.

Resumo

Nesta dissertação propomos uma metodologia que automatize a recolha de recursos na Web e facilite a sua exploração. Um recurso é uma colecção de documentos referentes a um tópico específico definido pelo utilizador. A intervenção do utilizador é explicitamente requerida numa fase inicial, quando este especifica as suas necessidades de informação e fornece alguns documentos exemplificativos. Após esta fase inicial, de definição e especificação das necessidades de informação, a metodologia mantém-se alinhada com a contínua evolução das preferências do utilizador que são permanentemente monitorizadas e seguidas sem que seja necessário requerer explicitamente a sua intervenção. Para tal, a metodologia analisa as preferências do utilizador a partir das suas acções – guardar, imprimir, visualizar, alterar a categoria de documentos – que são automaticamente registadas durante cada sessão. Desta forma o utilizador fornece informação valiosa ao sistema sem qualquer esforço adicional. A metodologia prevê um nível de apresentação, desenhado com o objectivo de permitir a exploração e análise de colecções volumosas de documentos, através do qual o utilizador explora os seus recursos.

Os recursos são compilados através de um processo de *meta-search*, onde as pesquisas são programadas por um agente que analisa o compromisso entre a actualidade do recurso e a percentagem de documentos duplicados nas respostas do processo de recolha. As pesquisas são programadas de forma a manter a actualidade do recurso, reduzindo, simultaneamente, o número de pesquisas efectuadas.

A metodologia propõe também os mecanismos necessários para avaliar e controlar de forma automática a qualidade global do sistema. Esta qualidade é definida num espaço tridimensional cujas dimensões quantificam o desempenho no que se refere ao nível de Automação, Eficácia e Eficiência. Cada uma destas dimensões agrega um conjunto de medidas relevantes para a qualidade global do sistema: o nível de Automação é calculado a partir da carga de trabalho que é explicitamente requerida ao utilizador; a Eficácia é calculada a partir das medidas de *precison* e *accuracy*; a Eficiência é

calculada com base nas medidas de *recall*, *freshness* e *novelty*. O sistema mede e regista permanentemente o valor dos seus parâmetros de qualidade globais, que são usados para activar procedimentos correctivos ou preventivos de forma a corrigir ou antecipar uma degradação da qualidade global do sistema.

A classificação de páginas Web assume-se como uma tarefa crítica na nossa metodologia. Para avaliar da adequação de técnicas de aprendizagem semi-supervisionada foram desenhadas e realizadas algumas experiências. A realização destas experiências foi suportada por um protótipo que implementa parte da metodologia proposta e que foi implementado no decurso deste trabalho. Em particular este protótipo foi utilizado para compilar dois recursos distintos e para estudar a taxa de erro e a robustez da tarefa de classificação semi-automática.

Abstract

This dissertation proposes a methodology to automatically retrieve resources from the Web and to keep them up-to-date. A resource is a document collection on some specific topic defined by the user. The user intervention is explicitly required at an initial phase, when the user specifies his or her information need and provides some exemplary documents. Once this initial specification is concluded, the methodology automatically keeps lined up with the user's interests. The methodology keeps track of the user's preferences and its evolution over time without the need to explicitly require additional user effort. User's interests are monitored and analyzed from user actions – such as printing, saving, viewing or changing the current category of a document – that are automatically recorded during user sessions. This way the user feeds the system with valuable feedback without noticing it. Users explore resources through a presentation layer, designed with the main purpose of enabling the analysis of large document collections.

Resources are retrieved through a continuous meta-search process, where queries are scheduled by an agent that analyses the compromise between resource freshness and percentage of duplicates at the meta-search answers, in order to keep the resource up-to-date while reducing the number of submitted queries.

We also propose a framework to automatically evaluate and control the system's global quality. This framework is based on a three-dimensional space with dimensions representing *Automation*, *Efficacy* and *Efficiency*. Each dimension in this space aggregates a set of measures, relevant to the system's global quality: Automation is computed from user workload; Efficacy is computed from precision and accuracy; Efficiency is computed from recall, freshness and novelty. The quality of the system is permanently being monitored; the system periodically measures and stores the values of its quality parameters. The methodology uses this quality log to trigger corrective and preventive procedures, in order to automatically correct or prevent quality degradation.

The classification of Web pages is a critical task at our methodology. Some experiences were conducted to evaluate semi-supervised learning techniques. We have used the

prototype deployed during this work to compile two distinct resources and to study accuracy and robustness on them.

Contents

1 Introduction

The World Wide Web, or simply the Web (World Wide Web Consortium, http://www.w3c.org), may be seen as a huge collection of documents, or pages, freely produced and published by a very large number of people, without any solid editorial control. This is probably the most democratic – and anarchic, comprehensive and widespread mean for anyone to express his or hers feelings, comments, convictions and ideas, independently of ethnics, sex, religion or any other characteristic of human societies.

The Web is a comprehensive, dynamic, permanently up-to-date repository of information regarding most of the areas of human knowledge (Hu, 2002) and supporting an increasingly important part of commercial, artistic, scientific and personal transactions, which gives rise to a very strong interest from individuals, as well as from institutions, at a universal scale. By the end of year 2002 the Web would comprise about 4000 million pages, corresponding to 60 TB of information, as reported by Web Characterization Project, http://wcp.oclc.org. According to recent surveys from Search Engine Watch (http://searchenginewatch.com), Nua (http://www.nua.ie), and Nielsen, (http://www.nielse-netratings.com) – organisms that collect information on Internet traffic and utilization – in 2004 there were over 900 million web users. In a recent study, Nielsen's forecasts anticipate the dimension of worldwide Internet population to be of around 1070 millions by 2005, 1210 millions by 2006 and 1350 millions by 2007, increasing at rates higher than 10% a year. This study indexes a representative panel of Internet users from the United Sates of America and concludes that people are increasingly using the Internet as a primary source of information, news, and entertainment, spending, by October 2004, 40,2% of their time on the Internet analyzing content and 4,3% of that time searching, while, by October 2003, these figures were, respectively, 35,3% and 3%.

These figures show that the Web itself, and the private and institutional interests it promotes, are growing and changing rapidly (Lim et al, 2001), at a global scale, both as mean of divulgation and dissemination and also as a source of generic and specialized information. Web users have already realized the potential of this huge information

source and use it for many purposes, mainly in order to satisfy specific information needs, which is a major and promising utilization of Web given its comprehensiveness and the interest it has been raising in all areas of human endeavor.

However, retrieving information from the Web is not an easy task since its characteristics place many difficulties to users willing to explore it as an information source. Moreover, retrieving information is just the first step. Information retrieved from the Web is usually very extensive, composed of large document collections. Organizing this information in a convenient way improves the efficiency of the final stage of exploring the gathered information, the ultimate target of all this process. In order to take advantage of the intrinsic value present in this huge informational system there is a need for tools that help people to explore it and to retrieve, organize and analyze relevant information. In this dissertation we propose a methodology to assist users in the task of retrieving resources from the Web and exploring them. A resource is a document collection that is representative of some specific topic defined by the user.

1.1 Challenges

Satisfying a user's information need through the web environment raises several difficulties to the user that can be worked around. Some of these difficulties come from the Web itself:

- *Web dimension*: the large amount of information available on the Web makes it hard to search for the relevant information in reasonable time;
- *Web dynamics*: the Web is continuously changing its state, which requires a continuous effort from users who wish to keep in track.

The absence of any solid editorial control over web content contributes largely to Web success but also raises problems to the process of information retrieval:

- *document heterogeneity*: the absence of rules, globally establishing the structure and format of web documents, allows anyone to publish documents about anything, in any format, structure and language;
- *content quality*: web documents may contain errors, may be invalid or totally incorrect; besides some documents are near or exact copies of other.

The user and the interface between the user and the Web generate some other difficulties:

- *specifying information needs*: a user information need is usually expressed through a set of keywords that the user believes are representative, which becomes a difficult task due to the ambiguity and complexity of natural language. Any other way of specifying an information need must be simultaneously expressive, to allow for the specification of any topic, and simple to understand and use, in order to be easily adopted by anyone;
- *presenting information*: the answers produced by web search services are composed by sets of documents, which are usually large. To be able to efficiently analyze and explore these extensive document collections a human user requires some tools that may help in this task.

This set of difficulties is not exhaustive but presents some of the most significant problems to overcome by any system aimed at gathering information from the Web. For all these problems there are solutions; however, some of them are not fully satisfactory and new problems arise frequently.

1.2 Current solutions

There are some solutions that help users fulfill their information needs; yet none of them seems to address all of the important stages in the process. The satisfaction of an information need on the Web is usually seen as an ephemeral one-step process of information search; the user is usually not assisted in the subsequent, and very important, tasks of organizing, analyzing and exploring the real value and relevance of the answers produced. The specification of an information need is another important and usually neglected task; although most search services rely on the traditional form of specification based on a set of keywords, there are a few systems that try to learn user needs from other sources of evidence (Bueno et al, 2001; Chen et al, 2001; Mladenic, 1999; Lieberman, 1995).

Users interested in obtaining information from the Web have several systems available that can help them. In a general way these systems are either search engines or topic

directory systems. Public web access systems, whether search engines or topic directories, are generic tools aiming to satisfy every query; they are not prepared to specifically satisfy a particular information need but, instead, are intended to give a satisfactory answer to any user query.

Search engines are very efficacious and extremely useful tools – it is frequently easier to find a document on the Web than at our own hard disks – but present their answers to the user in a flat list, which is not easy to explore when the answer is a large set of document.

Topic directories, on the other hand, organize documents according to a pre-defined, generally accepted, taxonomy but one that is expected to satisfy any generic information need, which is not prepared to organize document collections in an ad-hoc fashion, particularly important to a specific user need. There are some specialized systems, which are intended to satisfy narrow information needs, such as CiteSeer (http://www.citeseer.com) for scientific literature (Lawrence, 1999a), but even in these cases all the users of the restricted group of interest are depending on the same rigid structure and base concepts, despite their specific information needs.

In general the user is limited to the specification of a set of keywords and has no other means to interact and personalize the system answer or adapt it to some particular information need. This generalized lack of adaptive or personalization skills of the current search services may generate significant difficulties to the user who is interested in collecting information on a specific topic and organize it according to some particular structure, from a huge information repository such as the Web.

1.3 Our hypothesis

When the information need is persistent the user wishes to keep an updated image, of the topic of interest on the Web, along time. He or she will probably also be interested in organizing this web image according to some ontological structure, which should not be generic but, instead, should be specific to the user and to the topic.

In our dissertation we claim that the afore mentioned problems, along with the need to keep an updated and organized image of the Web, regarding a particular information

need, may be overcome with the help of a semi-automatic resource compilation methodology. This methodology should assist the users in the process of compiling resources on the Web in a way that is effortless to the user, with the following characteristics:

- allow for the broad specification of any topic, despite its nature, which requires a flexible and simple topic specification language and method;
- give the user the power to personalize and adapt the system's answer to his or hers particular needs;
- keep track of, and adapt resource's organization to drift in user needs without explicitly requiring additional user efforts;
- maintain accurate, fresh and comprehensive views of topics;
- provide mechanisms to enable the effective exploration of resources, which are large collections of document.

The hypothesis of this dissertation is that it is possible to retrieve, explore and keep track of a document collection on some particular topic (for instance Artificial Intelligence), organized according to some taxonomy (for instance Documents, Events and Research groups) – both specified by the user through examples, based on some particular information need, which may be non static – improving the user's satisfaction while saving his or hers time.

More precisely we intend to define a methodology, which can assist users in the task of maintaining a resource, related to some topic of interest, organized according to some particular hierarchy of concepts defined by the user.

1.4 Contributions of the thesis

It has been our aim in this work to produce some contributions that might help users satisfying their specific information needs through the Web. We believe we have accomplished the following:

- **Methodology**: the main concern of this work has been the development of one methodology for semi-automatic resource compilation on the Web. In this dissertation, we will describe one such methodology.
- **Dataset acquisition and classification**: the acquisition and non-automatic classification of web pages is a time consuming task subject to human error (Macskassy et al, 1998). In the methodology that we propose, an information need is specified through examples. These examples take the form of a set of web pages that must be searched, retrieved and partially classified. We have deployed a tool, which is responsible for the semi-automatic execution of these tasks, and implements simple mechanisms to prevent human misclassification.
- **Semi-supervised experimental evaluation**: some experiences have been conducted in order to estimate the performance and adequacy of semi-supervised learning in our framework. The results we have obtained shed some light on the importance of semi-supervised learning for the task of automatic resource compilation on the Web.
- **Adaptive quality control**: we propose a global performance evaluation framework, measuring and adapting system performance or quality, which may be operated either by the user or automatically by the system itself. The present system's quality, from the user's point of view, might be continuously measured, allowing for automatic corrections if and when significant deviations are detected – through the execution of corrective procedures – or foreseen – executing preventive procedures. This adaptive quality control framework may also detect and follow drift in user information need, which will naturally occur if user interests are long termed.

1.5 Structure of the thesis

After this introduction, chapter 2 – Web mining – defines the Web, briefly reviews the web mining process and describes the available web search services.

In chapter 3 – Automatic resource compilation – we try to identify the potential users of automatic resource compilation systems, and describe the state of the art of web mining

major areas of interest to this type of systems. Still in this chapter we compare some of the systems that address this problem and try to relate them to our work.

Chapter 4 – webTOPIC: the proposed methodology – is dedicated to the description of our methodology. The proposed methodology has four crucial areas: topic specification, resource acquisition, resource organization and resource presentation, which are explained in detail. We also define here our quality control framework and describe the adaptive quality sub-system and how it can be used to evaluate and conduct the system, at real time, towards user drifting interests.

In chapter 5 – Experimental evaluation – we evaluate the suitability of semi-supervised learning for our purposes. In order to do it, several experiences have been designed and conducted; these experiences are described and the results are analyzed.

Chapter 6 – Conclusions – presents our conclusions and points out some directions for future research.

2 Web mining

2.1 The Web

The Web is the environment where our work will be developed. An appropriate definition and characterization is essential and will be given below.

The *Web* is a public service constituted by a set of applications aimed at extracting documents from computers accessible in Internet – the Internet is a network of computer networks. We may also describe the Web as an information repository distributed over millions of computers interconnected through Internet (Baldi et al, 2003).

The term *web document*, or *web page*, refers to any file that we may retrieve from the Web through its URL – *URL, Uniform Resource Locator*, or *URI, Uniform Resource Identifier*, is a character string describing a unique address of a document on the Web; web browsers present web documents to users in response to a URL.

A *web site* is defined as a distinct location in the Internet, identified by an IP address that returns a web page in response to a HTTP request. One site is the set of all interconnected web pages, stored at the same IP address. *IP, Internet Protocol*, address in the network address of a computer in Internet that allows connecting to a server through TCP. *HTTP* is an acronym for *HyperText Transport Protocol*, the transport layer protocol used in Internet to transfer hypertext documents as well as other types of files.

The World Wide Web Consortium (W3C) (http://www.w3c.org) defines Web in a broad way: "the World Wide Web is the universe of network-accessible information, an embodiment of human knowledge". The statement of a more concise definition (American Heritage Dictionary, 2000) may help our goals: "World Wide Web is the complete set of documents residing on all Internet servers that use the HTTP protocol, accessible to users via a simple point and click system".

This last definition describes the Web in a clear but incomplete way since it does not include all of it. It is common to accept the distinction between surface web and deep

web: the *surface web* is the web portion that can be reached by following the links present in documents; the *deep web* is the set of documents that cannot be reached through the web link structure, and therefore are not accessible through crawling – *crawling* is the process of exploring the Web by following the links in a document to obtain new documents. The deep web is mainly constituted by web pages that are dynamically generated in response to a user request.

When using the last definition some care is needed since the Web is not exclusively a collection of documents, but it is also made up of services. A *web service* is a software application identified by a URL, whose interfaces and bindings are capable of being defined, described, and discovered as XML artifacts (W3C glossary, http://www.w3.org/TR/2002/WD-wsa-gloss-20020605). Although web services can be conceptualized as web documents it seems important to state this aspect of the Web.

2.1.1 Characterization

Web dissemination, since its earliest days – at the end of the 1980 – until now, has been so vigorous that it is not rare to take it for the Internet (http://en.wikibooks.org/wiki/The_Internet). The web dimension and rate of growth are difficult to measure since there is no universal catalog referencing all web URLs. Besides, Web comprises a huge volume of documents and its content is very dynamic. These characteristics make it very difficult to have an accurate image of the Web. Despite these difficulties some efforts have been made in order to characterize it (O'Neill et al, 2003; Bharat et al, 1998; Lawrence et al, 1998). The Web Characterization Project (WCP) (http://wcp.oclc.org) may help us understand some interesting web properties.

The WCP results refer to the *public web* – a *public web site* is one that offers free and unrestricted access to a significant portion of its content. Web dimension can be estimated from the number of sites. The last survey from the WCP, referring to the end of year 2002, stated that the mean number of pages per site was estimated to be 441; the mean size of a web page stood between 10 and 20 KB; the number of public web sites was about 3,080 millions. Based on these figures we can estimate the public web to be constituted by about 1400 million pages with a total size of 20 TB. Taking into account that the public web represented 35% of the global web, we can estimate that the size of

the whole Web, by that time, would have been of around 4000 million pages, corresponding to 60 TB.

The Web's growing rate was initially very high but in the last few years this tendency has been smoothed and it seems reasonable to assume that the web dimension is reaching a steady state (O'Neill et al, 2003).

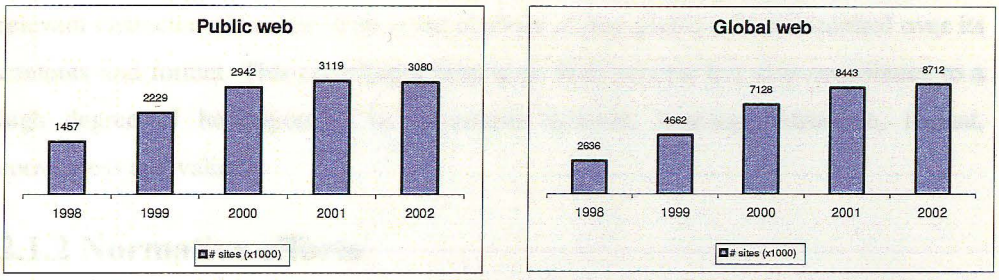


Figure 2.1 – Web growing rate

It is also interesting to take a look at WCP results concerning web site volatility. Table 2.1 shows the percentage of IP addresses that identify a web site in a certain year given that it already identified a web site in the previous year.

	1998	**1999**	**2000**	**2001**	**2002**
1998	100%	52%	28%	15%	8%
1999		100%	50%	27%	14%
2000			100%	47%	26%
2001				100%	46%
2002					100%

Table 2.1 – Web site volatility

Despite this high volatility, with sites disappearing at a rate of near 50% a year, the Web has always been increasing or, at least, maintaining its global dimension. The web contents refreshment rate is estimated to be of about 600 GB/month (Chakrabarti, 2003): sites disappear at a high rate but new ones are generated at faster rates, according to (Kahle, 1997) the average lifetime for a URL is 44 days – an URL is considered to be

"alive" while it returns a valid web document when invoked – besides, the number of pages per site has also been slightly increasing (Bharat et al, 1998).

Due to its comprehensiveness, with contents related to most subjects of human activity, and global public acceptance, either at a personal or institutional level, the Web is widely explored as an information source. Web's dimension and dynamic nature become unfavorable characteristics when it comes to retrieve information. Another relevant characteristic of the Web is the absence of any global editorial control over its contents and format. This contributes largely to Web success but also contributes to a high degree of heterogeneity in documents content, language, structure, format, correctness and validity.

2.1.2 Normative efforts

Although the problems rose by the Web's dimension and dynamics require special treatment it seems that the major difficulties, concerning the processing of web documents, are generated by the lack of editorial rules and the lack of a common ontology, which would allow for automatic unambiguous document specification and interpretation. In the absence of such normative rules, each document has to be treated as unique. In this scenario, document processing cannot be based on any underlying structure. Although HTML already involves some structure its use is not mandatory. Therefore, the higher level of abstraction that may assure compatibility with a generic web document is the common "bag of words" (Chakrabarti, 2003). This low abstraction level is not very helpful for automatic processing, requiring significant computational costs.

Some attempts are being made in order to increase the structure of web documents and the semantics of their constituent elements. The *Semantic Web* is an attempt from W3C to develop a framework that facilitates the automatic processing of web information. According to W3C "the Semantic Web is the abstract representation of data on the World Wide Web, based on the RDF (Resource Description Framework) standards and other standards to be defined. It is being developed by the W3C in collaboration with a large number of researchers and industrial partners". The main purpose of the Semantic Web is to bring to the Web "the idea of having data defined and linked in a way that it

can be used for more effective discovery, automation, integration, and reuse across various applications". The semantic web seems to be a promising project but it is yet at an early stage. Nevertheless, the interest in this framework and the benefits it might bring along already exist (Berendt, 2002; Lu et al, 2002).

The Web is a vast, comprehensive, and popular repository, containing information related to almost all human activities. Despite the difficulties this medium poses to non-automatic as well as to automatic processing, it has been increasingly explored and has been motivating efforts, from both academic and industry, that aim to facilitate this exploration.

2.2 Web mining process

Web mining is the overall process of discovering and extracting potentially useful and previously unknown information from web documents and services. This process can be generally viewed as being composed of the following phases (adapted from (Kosala et al, 2000; Etzioni, 1996)):

1. *Acquisition* ("Resource finding" in (Kosala et al, 2000), "Resource discovery" in (Etzioni, 1996)): the aim of this phase is to find and retrieve, from the Web, as many relevant documents as possible while retrieving as few non-relevant documents as possible; relevance is some measure of the importance of the document concerning the user information need. This is a phase of Information Retrieval (Baeza-Yates et al, 1999) where a few problems arise, such as: how can the user specify a particular information need; what relevance measures should be used and how can we obtain them in order to rank documents; how to keep an up-to-date image of the Web so that it is possible to answer the user query in due time.
2. *Pre-processing* ("Information pre-processing and selection" in (Kosala et al, 2000), "Information extraction" in (Etzioni, 1996)): comprises any transformation process applied to the retrieved documents. These transformation processes are usually applied to reduce the number of terms present in documents (by eliminating irrelevant terms – stop-word removal – and by reducing terms to their

semantic root – stemming), to generate document models that are adequate for the following phases and to extract meaningful information from the documents (requiring Information Extraction techniques).

3. *Learning* ("Generalization" in both (Etzioni, 1996) and (Kosala et al, 2000)): this phase, which intends to find patterns and to learn how to satisfy the web mining process objectives, is usually implemented with the support of some statistical, machine learning or data mining techniques, adapted to the specific characteristics of the Web. These characteristics generate new problems, which usually require some special attention: the lack of structure in web documents; the feature space dimension is usually much higher than the number of instances; instance (document) labeling is a very expensive task, to mention a few.

4. *Analysis*: the phase of validation and interpretation of the results provided by the mining process usually requires human intervention, particularly in the current web environment where the process objectives are difficult to express, ambiguous and may be continuously evolving. Although there is necessarily a validation step made in laboratory, we believe that some further sort of relevance feedback, explicitly or implicitly provided by the user interaction with the web mining process may help in keeping aligned with the objectives.

5. *Presentation*: we believe this is a fundamental phase in web mining, mainly for two reasons: first, we need to present the results of the web mining process to the user; second, we would like to have some means to capture user behavior, which is reflected through the actions that the user performs during his or her interaction with the system. Although the presentation of the results does not necessarily have to apply visual elements, we certainly need some way to inform the user about the effects produced by the web mining process. In addition, mining large document collections, which is a particular concern to us, might be significantly improved with the aid of specialized graphical interfaces. This second reason is based on the hypothesis that some kind of relevance feedback may be obtained from user interaction. Assuming this hypothesis, we could benefit if we had some way to present the automatically compiled collection to the user, in order to trigger some reactions, capture them and then process and analyze user behavior reflected

during this interaction with the system.

We strongly believe that the user's final goal, which is assumed in our work to be the satisfaction of a certain information need, will not be properly achieved without giving him or her some simple means to analyze document collections (especially large document collections) in order to perceive their global content, structure and coverage. To accomplish this, we propose the fifth phase – Presentation – which is not originally proposed neither in (Etzioni, 1996), which considers only the first three phases, nor in (Kosala et al, 2000), which considers the first four.

It is clear from the several phases involved in web mining that this research field encompasses contributions from several fields, such as: Information Retrieval (IR), Information Extraction (IE), Artificial Intelligence, Data Mining, Statistics, Database and Social Networks, especially Bibliometry (Chakrabarti, 2003), to name a few. Despite their origin, the web mining techniques that were originated in other fields had to be especially adapted to the specific characteristics of the Web.

In web IR, for instance, one has to pay attention to some circumstances that are not present in traditional IR: spammers (Chakrabarti, 2003), motivated by economic reasons and other, deliberately add popular query terms to documents unrelated to those terms (text can be hidden from user if font and background have the same color, the so called *font color spamming*); web documents may be replicated, generating near or exact copies, and complete web sites may be mirrored in different addresses; web document's content validity, correctness and quality are highly variable and difficult to measure; the subject of web documents is also difficult to determine and the labeling process is expensive and time consuming. None of these problems arise in traditional IR.

In typical machine learning and data mining problems, the examples are described in a tabular form where columns represent the properties, characteristics or features that describe the concept and each line corresponds to an individual instance or example. The features are unambiguous; their semantic is precisely known and every single example is described with exactly the same feature set. In a typical problem from these fields, the number of features is usually some orders of magnitude smaller than the number of examples. Document models used in the learning task of web mining and the document collections that are explored usually do not share these desirable

characteristics: the feature space dimension is usually much larger than the number of available examples (documents), each particular document is described by a subset of all features and the intersection of these subsets is of reduced dimension, resulting in a sparse document/term matrix, which creates additional difficulties to the learning task.

One relevant drawback of a web query, when compared to a database query, which is due to the lack of structure in web documents, is that it is not usually possible to answer a web query without some degree of uncertainty (Baeza-Yates et al, 1999). While in the database field a query is unambiguous and its mapping to the examples is precise, in the web environment such precision is not possible since the query specification – a set of keywords – is usually ambiguous and the relevance of a certain document to the user information need, specified through that query, is, therefore, subject to ambiguity and uncertainty.

These special features of web mining result mainly from the volume and dynamics of the Web, from the lack of structure in web documents and from the ambiguity of the user's specifications.

Web mining is guided by two long-term goals (Borges, 2000): helping to build good web sites and assisting the individual user while navigating the Web. The methodology proposed in this dissertation aims at satisfying both these goals: on the one hand it might help building good web sites if we look at it from an editorial point of view – our methodology might be used to personalize the gathering, organization and presentation of document collections, according to each particular site client/user information needs, on the other hand the methodology might be used by a particular user who is trying to keep up-to-date, or needs to obtain information on a given subject organized in some particular fashion.

2.3 Web mining major areas

It is generally accepted that web mining is currently being developed towards three main research directions, which are related to the type of data that is mined: web content mining, web structure mining and web usage mining (Kosala et al, 2000). Recently another type of data – document change, page age and information recency –

is generating some research interest: it is related to a temporal dimension and allows for analyzing the growth and dynamics – over time – of the Web (Baeza-Yates et al, 2002; Lim et al, 2001; Cho et al, 2000). This categorization of web mining main directions is merely conceptual, these areas are not mutually exclusive and some techniques dedicated to one of them may use data that is typically associated with others.

2.3.1 Web content mining

Web content mining concerns the discovery of useful information from web data. Web data is available in many different formats (Baeza-Yates, 2003) – textual, metadata, links, multimedia objects, hidden and dynamic pages and semantic data. Besides, some of the information sources are also diverging from the "traditional" HTML pages, as many applications are being migrated or made accessible on the web environment. Although some of this information is not publicly accessible, there are some distinct types of sources and formats of information available online. Despite this diversity in data and source types, the HTML format is predominant and the major research efforts are still directed towards text and hypertext documents. The IR, IE and text mining fields assume a rather important position in the context of web content mining.

2.3.2 Web structure mining

Web structure mining tries to infer knowledge from the link structure on the Web (Chakrabarti, 1999). Web documents typically point at related documents through a link, or hyperlink – consisting of a text string, known as the anchor text, and an associated URL, which is the address of the pointed document – forming a kind of social network that can be represented by a directed graph where nodes represent documents and arcs represent the links between them. The analysis and exploration of this graph is the main goal of web structure mining (Donato et al, 2000; Kumar et al, 2000; Kleinberg et al, 1999). In this field, two algorithms, which rank web pages according to their relevance, have been receiving special attention since their appearance in 1998: PageRank (Brin et al, 1998) and Hyperlink Induced Topic Search, or HITS (Kleinberg, 1998).

PageRank, used in the Google search engine (http://www.google.pt) is a score of

popularity; which measures the interest of a web page as a function of the number of links that point at it. The hypothesis is that the more citations the document has, the more interesting it is. A major advantage is that since this measure is independent from the query, it can be pre-computed. PageRank is computed by a random surfer, which randomly follows links on the Web. The importance of a page is proportional to the number of times that the page is visited during this random walk process (Chakrabarti, 2000).

The HITS algorithm, used in the Clever search engine (Chakrabarti et al, 1998), associates to each web page two measures: hub and authority. A *hub* is defined as a page that contains a set of links to documents referring to some common topic. An *authority* is a page that is pointed to by a set of hubs. The hub measure of a page increases with the authority measure of the pages that it points to. The authority measure increases with the hub measure of the pages that point to it. Both measures are iteratively assigned to the document collection until convergence is reached. HITS depends on a search engine to retrieve a set of documents that match the user query and that are expected to include the authoritative pages regarding the user information need. This set of documents, called the *root set*, is extended by including all the in-links – pages that point to the current page – and out-links – pages that are pointed to by the current page – of the pages in the root set. This extended set of pages is then used to iteratively compute each page's hub and authority measures and, finally, pages are ordered by their authority measure.

The current macro-structure of the Web may be represented by a model (Broder et al, 2000) including a central core of strongly connected components – pages that link to each other; a set of pages which point into pages lying in this central core, this set is referred to as the *in* set; a set of pages, referred to as *out*, which are mainly pointed by pages in this central core; a set of pages, called *tubes*, which link to pages from the *in* and *out* sets, without going through the main core; and several sets of isolated, disconnected components, called *islands*. Although there is no evidence, from previous research on social networks, supporting this structural pattern and no guarantee that it will prevail in the future (Chakrabarti, 2003), present experiences (Baeza-Yates et al, 2002) seem to confirm this macro structure model and, furthermore, explore it in order to improve IR algorithms.

2.3.3 Web usage mining

Web usage mining tries to explore user behavior on the Web by analyzing data originated by user interaction and automatically recorded during the session in web server logs, proxy server logs, browser logs, user profiles, registration forms, bookmarks, cookies and other. The applications of web usage mining usually intend to either learn a user profile or learn user navigation patterns. Web usage mining is essentially aimed at predicting the next user request based on the analysis of previous user requests. The prediction may be based on user profiles that the system learns.

The web usage mining process follows one of two common approaches. One such approach extracts data from server log files and uses this log data directly, after some appropriate pre-processing; the other approach starts by loading the usage data to some pre-defined relational, or multidimensional, database schema and then applies data mining and OLAP techniques to help users extract relevant patterns – *OLAP, OnLine Analytical Processing*, refers to the technology that enables fast access to large volumes of data which is organized in dimensions and aggregated (Agrawal et al, 1997) in order to focus on relevant facts while ignoring irrelevant details.

When users interact with a web site, web server logs record their requests. A *web server log* is a text file that contains one entry for each request made to the server. The information that is recorded includes, among others (Borges, 2000): the IP address of the computer making the request, date and time of the request, URL of the page where the request was originated, size of file transferred, status of the request indicating if it was successful.

Log data pre-processing is an essential step of the web usage mining process, and may be seen as consisting of two tasks (Cooley et al, 1997): data cleaning and transaction identification. Server log data may be very noisy due to some operational characteristics of the web environment: a user can be allocated more than one IP address during the same session, the same user may visit the same server at different moments with different goals, some users connect through proxy servers, one single user request may originate several requests at the server once HTTP protocol requires a different connection for each file and an HTML page may be composed of several files, the widespread use of cache servers on the Web may result in some requests that do not

reach the server once they are served by the cache server. All these facts require heavy pre-processing before one can properly explore log data. Besides, many of the problems of web access data quality and completion can only be minored, but not entirely solved, through the adequate use of pre-processing heuristics.

Markov models are very common in modeling user requests or user paths within a site. Association rules and other standard data mining and OLAP techniques are also explored (Cooley et al, 1997). (Borges, 2000) presents an overview of the most relevant work in web usage mining.

2.4 User tasks

When users want to fulfill a certain information need from the Web their interaction with it may be classified as browsing or retrieval (also known as searching) according to their purpose (Baeza-Yates et al, 1999). *Browsing* is the process of retrieving information when the main objectives are not clearly defined in the beginning and whose purpose might change during the interaction with the system. When users' objectives are clearly defined their interaction with the Web is classified as *retrieval* or *searching*. User's objectives are usually specified through a set of (usually one or two) keywords that, in his or hers understanding, define his or hers specific information need.

The retrieval or searching of information might further be divided into ad-hoc and filtering: an *ad-hoc* retrieval process is one where the user poses different queries to a static collection while in the *filtering* process the user poses the same static queries to a dynamic collection. In filtering the result set is usually presented to the user without being subject to any previous ranking process. The special case of filtering with relevance ranking is called *routing*. Routing is ranked filtering.

These user tasks may be implicitly associated to the subjacent information needs. Ad-hoc searching will probably be used when the user is exploring a kind of digital library, looking for information on some specific definite topic. The task of filtering may be automatically executed by a system that tries to keep track of a certain topic. This is the task that is guaranteed by resource compilation systems. The browsing task

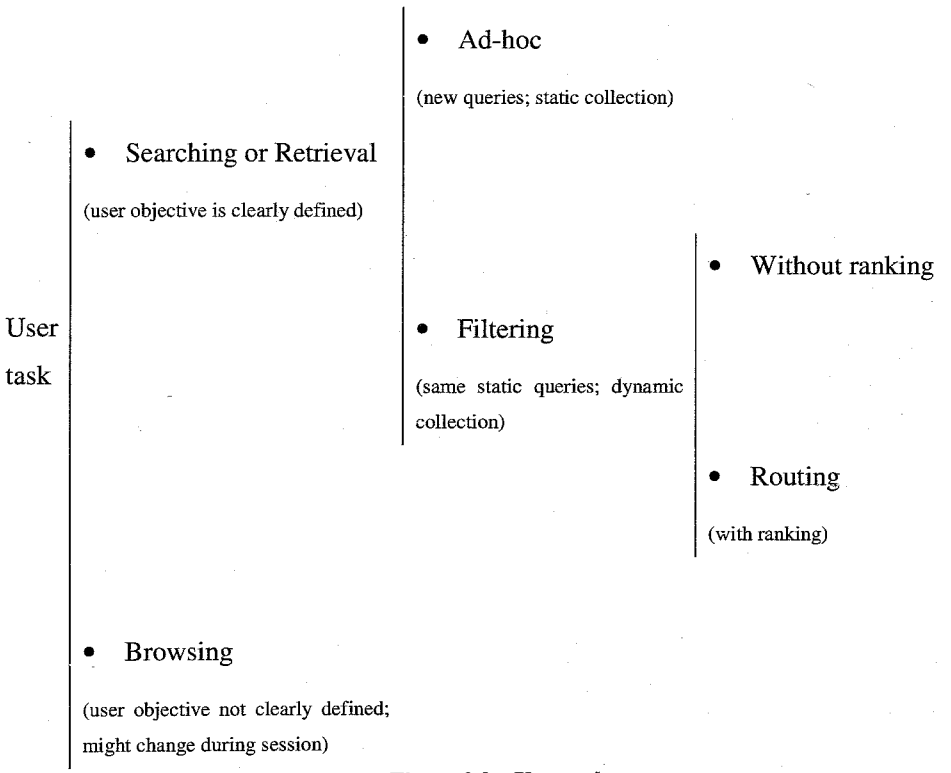


Figure 2.2 – User task

seems appropriate when the user wants to get a first glance at a certain topic, trying to learn a generic overview. In this scenario collateral subjects are important once they help understanding related issues and influences and help to establish domain frontiers and interfaces.

In a general learning process the user might start by browsing, to get a general idea on the topic, then searching for some specific sub-topic of major interest, and finally, if the interest in the topic is high, starts filtering on a permanent basis. The methodology we propose in this thesis aims at helping the user in this last step.

2.5 Web search services

The Web has become the largest easily available repository of information (Baeza-Yates, 2003). Despite the variability in its contents' quality and correctness, it has been increasingly used as a source of information for the most various purposes and activities

of people in general. By the end of 2002 the Web had more than 3 million distinct sites and over 900 million users online by the middle of 2004, as reported by Web Characterization Project and Nielsen.

Finding information on the Web would be simpler if there was a universal web index or catalog. Since it does not exist, knowing the content of the Web requires the retrieval and analysis of web documents.

Web search services are publicly available tools for retrieving documents that might satisfy specific information needs. Due to the high volume and diversity of the Web, it is permanently explored in order to virtually satisfy any information need anyone can think of. In this scenario, web search services assume a rather important role, and special research efforts have been given to the personalization and increasing performance of such systems. In this section we will concentrate on issues related to the web search services that help users to explore this huge, heterogeneous and dynamic information source.

Web characteristics (web dimension, dynamics and heterogeneity and contents quality) are adverse to the retrieval task, posing a few specific problems (Baeza-Yates, 2003; Wang et al, 2003) that must be solved. Among these we would like to stress a few, which are of special interest to us:

- Keeping the index fresh: (Cho et al, 2000) estimate that more than 40% of the pages from *.com* domain change every day. The high dynamics of web environment – where documents are updated, removed and added very frequently and without centralized control – originate several problems, such as broken links, that largely contribute to user dissatisfaction and to decrease the performance and the quality of the search engines' index databases. Another aspect concerning the maintenance of the index is that a user may be interested in keeping a retrospective view of the topic; in this case it would help if the system could provide successive frozen views of the topic, showing the topic representation on the Web over time.
- Improving ranking: in particular to make it personal, adequate to each particular user and moment of time. We believe that an adaptive resource compilation system must depend on a ranking function that is related and dependent of three

main dimensions: user, topic and time.

- Exploiting user feedback: some techniques explicitly require user feedback to analyze the correctness of the system answer and use that information to adjust the models in use, as is the case of Rochio's method (Chakrabarti, 2003). We believe that the actions performed by the user, while interacting with the document collection provided by the system, are informative of the user interests; besides, this information can be automatically acquired without requiring any additional explicit effort from the user.
- Improving presentation: the system's interface is the only way the user has to communicate with the system and to transmit his or hers needs and feedback. It is through this interface that the user gets acquainted with the document collection compiled by the system. User interaction with the system occurs in two distinct moments: when specifying information needs and when analyzing search results.

2.5.1 Search engine architecture

The first widely used type of web search service was the search engine (Arasu et al, 2001) – Lycos, (http://www.lycos.com) appearing in 1994 was one of the first. A *search engine* is a system that fetches documents from the Web into a local repository and indexes them based on keywords or terms. Search engines model the Web as a text database, or index, of web contents that is used to answer user queries. User queries are typically composed of a set of keywords – the median web query is two words long (Spink et al, 2001) – that the search engine uses to find relevant documents by matching them with its index. Selected documents are then ranked according to some measure of relevance and presented to the user.

Typical centralized search engine architecture consists of two major functional parts (Wang et al, 2003): one that interacts with the user and is constituted by the query interface and the query processor, and the other that is responsible for searching the Web, fetching and indexing the documents. This is constituted by the crawler and the indexing module. The web image of the search engine, the set of crawled and indexed documents, is kept in a text repository. There are other approaches for search engine architectures but the IR techniques employed are similar regardless of the system's

physical architecture.

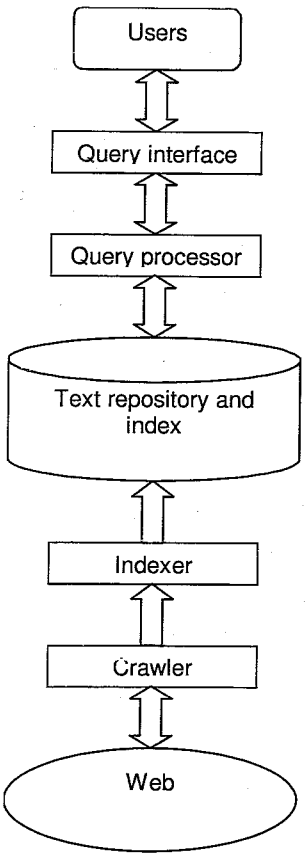


Figure 2.3 – Search engine architecture

Search engines fetch web documents, while traversing the Web by following its link structure, through the execution of a crawling process. Crawling is executed by crawlers, or spiders, which are robot programs that traverse the Web following links extracted from web pages. A crawler starts with a given set of web pages – the root set. From each of these pages the crawler extracts out-links, which are added to a pool of unvisited links, the *crawling frontier*. These pages, in their turn, are processed in order to extract their own out-links and the process continues in an infinite loop. A crucial challenge is to decide on the most interesting link to follow next, starting from the crawling frontier.

In practice, web search engines do not index the full Web and crawling cycles are not infinite; each crawling cycle is stopped when a sufficient number of web pages have been fetched. Crawling cycles may take a few weeks to a few months (Chakrabarti, 2003). Long crawling cycles may compromise index freshness although this problem may be reduced with the establishment of an appropriate index updating strategy, either periodic rebuilding or incremental updating (Wang et al, 2003). Using incremental crawlers – which exclusively visit the pages that are supposed to have changed since last crawling – instead of periodic crawlers – which periodically repeat the same crawling process in batch mode – might improve index freshness (Cho et al, 2000; Cho et al, 1999).

Crawling is based on web structure, materialized by the links among web pages. It is a broad, generalist process of retrieving web pages that merely follows links from page to page, aimed at high web coverage. Recently new paradigms have appeared that try to keep crawling more specific, in order to overcome problems of scale, that arise naturally from web dimension – such as *focused crawling* (Chakrabarti et al, 1999) and *intelligent crawling* (Aggarwal et al, 2001) – or that explore new types of data – as is the case of *collaborative crawling* (Aggarwal, 2004), which explores user behavior registered in proxy server logs.

Crawling the Web raises several low level problems (Chakrabarti, 2003) that must be taken into account when designing a crawler:

- HTTP server access must be planned taking into account that these servers usually assign limited time slots to each session in order to avoid denial of service; this requires that the crawler must control session time and number of downloads for session, while simultaneously trying to optimize throughput;
- bandwidth occupation must be maximized, which requires the establishment of simultaneous multiple sessions in order to reduce the negative impact of delays in the session establishment phase;
- URL resolution, which is the translation of the URL to the corresponding IP address, obtained through a query to a Domain Name Server (DNS) server;
- avoiding unauthorized access; web servers maintain a file, "robot.txt", describing the directories whose access is limited, and crawlers should respect these

indications;

- avoiding redundant fetches by eliminating duplicated URL before establishing the session;
- recognize and avoid spider traps, which are sets of URLs specifically designed to confuse crawlers and retain them in link loops;
- refresh strategy definition;
- compression and storage of fetched pages;
- maintaining a fresh index.

These issues are not directly addressed by our work; our methodology does not require any crawling since we will make use of public search engine indexes to retrieve web pages.

Once the text repository and index database are built, and the appropriate refreshment mechanisms are implemented, the crawling cycles are constantly executed in order to maintain the search engine's partial view of the Web as up-to-date and comprehensive as possible. The other part of the search engine, the query interface, uses the text repository and index database, and is responsible for executing the queries defined by the user and presenting the set of relevant pages, ranked according to some measure of relevance. Some difficulties arise in this area, among which we may refer:

- query language: the user should express his or hers information need in a simple and expressive format; this question is dependent on the type of index that the search engine relies on. If the index is term based, then the query language has to be term based also. Typical search engine queries are sets of keywords/terms that the user believes can express his or hers particular information need;
- document ranking: given the vagueness of information need specifications and document retrieval based on keywords, it is not adequate to measure document interest based on a binary indicator (interesting/not interesting), therefore documents must be ranked according to some interest measure. Although popularity (Brin et al, 1998) and hub and authority (Kleinberg et al, 1998) are the most common interest metrics in use today, there are some other metrics available, such as *novelty* (Zhang et al, 2000) – which measures to what degree one web

page is different from other web pages;

- document presentation: search engines' answers are, in general, very large, in part due to the fact that the user query is usually very broad and ambiguous. The presentation of the result set as a flat list, which is common, does not help the user to understand the relationships among the returned documents and, since users are rather impatient, it is very important that the top links are useful – (Spink et al, 2001) reported that nearly 50% of the users give up exploring after the first 10 to 20 links, whether their information needs have been satisfied or not, and in a similar study (Jansen et al, 2000) report that 58% of the users gave up before analyzing the first 10 links;
- relevance feedback: there is some strong evidence that relevance feedback may improve the quality of the returned result set. The main difficulty is to obtain this relevance feedback from the user, which demands for an extra effort from the user. *Rochio's relevance feedback* algorithm (Chakrabarti, 2003) is a common approach, which refines user query adding positively weighted terms from the interesting documents and negatively weighted terms from the non-interesting documents. This method requires the user to classify documents as relevant or non-relevant. *Pseudo-relevance feedback* is a version of this technique that tries to overcome this drawback by automatically classifying returned documents as interesting – typically the most relevant from the original result set – or non-interesting. This way it operates without requiring the user to define documents' interest. This process may significantly slow down the query response time of the search engine.

2.5.2 Search engine coverage

Different search engines present distinct views of the Web to the users. The number of web pages that are indexed and included in each one's database – web coverage – mainly limits these views.

Commercial search engines struggle for web coverage in a constant effort to be in the top or, at least, to keep updated (http://searchenginewatch.com). AltaVista appeared at the end of 1995 claiming to have a much larger index than the other search engines at

that time; as a consequence competitors made efforts to increase their own. By 1999 search engines where in pursuit of the 150 million page mark. When AltaVista and Northern Light search engines were celebrating hitting that mark, a new competitor appears, AllTheWeb, with an index size of 200 million pages. The pressure to increase coverage was very high again. AltaVista and AllTheWeb alternate the first place in the index size contest, until June 2000, when Google claimed a 500 million pages index. Google keeps the first place until 2002 when AllTheWeb declares to have broken the 2000 million pages. A few months later Google and Inktomi – Inktomi includes several search engines: HotBot, MSN Search, iWon – both claim to be indexing up to 3000 million pages. Search engine's web coverage and index sizes are constantly increasing and difficult to measure accurately; nevertheless Table 2.2 presents an overview of present values for search engines index size, in millions of pages, as claimed by the search engines themselves and estimated by Search Engine Showdown (http://searchengineshowdown.com.), an organism that collects search engines operating statistics.

Search engine	Estimate	Claim
Google	3.033	3.083
AlltheWeb	2.106	2.112
AltaVista	1.689	1.000
WiseNut	1.453	1.500
HotBot (Inktomi)	1.147	3.000
MSN Search (Inktomi)	1.018	3.000
Teoma	1.015	500

Table 2.2 – Search engine's database size

(Bharat et al, 1998) reports the relative web coverage of four search engines, referring to the year of 1997: AltaVista would index about 50% of public web, HotBot indexed a little below 39%, for the Excite search engine the coverage would be approximately 16% and Infoseek had coverage of less than 14%. Search engine coverage relative to the estimated size of the public web has decreased since 1997, with no engine indexing more than 16%, according to (Lawrence et al, 1999b), or 18%, according to (Chakrabarti, 2000), of the estimated size of the public web as of 1999.

These figures must be used with some care since their estimate is a difficult task, biased by many exogenous and uncontrollable factors. Besides, the application of different methodologies may yield substantially different results, which are not comparable. Another relevant aspect related to web coverage is the overlap of different indexes, according to (Bharat et al, 1998) search engines overlap in about 1,4%, that is, only about 1,4% of web pages are indexed by all search engines. This result is based on data from the year 1997. Another similar study, from Search Engine Showdown and related to the year 2002, analyzes 10 search engines and reports that there was an overlap of 2 out of the 141 pages, returned in response to some query; which interestingly yields an overlap of 1,4%. Although these figures are not directly comparable, we may think that if the search engines grew their coverage at a faster speed than web grows then it seems reasonable to expect that the overlap among different search engines should increase; since this overlap keeps the same, one can infer that the Web and search engines coverage grow at approximately the same rate.

Another interesting statistic, reported by Search Engine Showdown, refers to the freshness of search engines indexes (Table 2.3). The age of the oldest pages in a given search engine may be roughly seen as an indicator of its crawling period.

Nielsen estimated the number of people that are online throughout the world, by middle 2004, to be over 900 millions. According to a Search Engine Watch survey, over 75% of them explore the Web through web search services and spend more than 70% of their time searching online.

Search engine	Newest	Average	Oldest
Google	2	30	165
AlltheWeb	1	30	599
AltaVista	0	90	108
WiseNut	133	180	183
HotBot (Inktomi)	1	28	51
MSN Search (Inktomi)	1	28	51
Teoma	41	75	81

Legend: age in days

Table 2.3 – Index freshness

All these statistics must be analyzed with care and, unless they are obtained from the same data and using the same methodologies, they cannot be directly compared; however they give an overview and an idea of the importance that web search services have to users and to web dissemination.

2.5.3 Taxonomy of web search services

In section 2.5.1 we focused our attention on web search engines and some of the major difficulties that web IR pose. Search engines are probably the most widely used search service on the Web (http://www.searchenginewatch.com); however there are other types of web search services.

Topic directories organize documents in a hierarchical structure representing human knowledge. The taxonomy is pre-defined by specialists and is general. Human specialists also classify the documents. This is simultaneously the main advantage and the main drawback of such search systems: the manual classification process is rigorous and guarantees that the document's content is valid and correct but, on the other hand, this classification process is slow when compared to web growing speed. Consequently, topic directories have high precision but suffer from low recall. Another drawback of such systems, relevant to our work, is that the taxonomy is pre-defined and static; if the user is interested in a distinct document organization, or in a specific category not included in the taxonomy, then these search systems are not very helpful.

Search engines automatically retrieve documents from the Web; one difference between topic directory and search engines is that search engines include a crawler. Another difference is that, in general, search engines do not categorize neither organize their documents in any fashion – Tumba! is an exception, and hierarchically organizes retrieved document collections (Silva et al, 2002). Documents are indexed, typically based on their terms, and this index is used to determine which documents are relevant to some query, which is also specified through a set of terms or keywords. In search engines, indexing is not usually a bottleneck, as it happens in topic directories, since this process is automatic; however it is much more difficult to guarantee document's content quality.

Generic search engines depend on large, comprehensive crawls and indices, yet their

coverage is low. Some efforts are being made in order to improve crawlers; specific deep portholes may be more effective than generic portals in satisfying user's information needs. A generic search engine might be replaced by a pool of specialized portholes, which focus their attention in a specific sub-graph of the Web, instead of attempting to crawl the entire Web.

Focused crawling (Chakrabarti et al, 1999) is a technique that attempts to crawl just the relevant areas of the Web, concerning some topic, avoiding crawling irrelevant zones. The goal of a focused crawler is to selectively seek out pages that are relevant to a pre-defined set of topics. The topics are specified not using keywords, but using exemplary documents. The focused crawler contains a component, the *distiller*, which decides the visit priorities of new links based on their relevance to the set of topics of interest. The distiller can filter out irrelevant links, thus avoiding irrelevant web regions.

Meta-search is an approach that tries to explore the small overlap among search engines' indices (Bharat et al, 1998) sending the same query to a set of search engines and consequently merging the answers. These systems do not do any crawling but instead use the public search engine's databases. Meta-search engines provide a virtual view of the Web by integrating partial views from several search engines. The user poses a query to a meta-search engine, as if he was dealing with a singular search, by specifying a set of keywords; next the meta-search engine selects the most adequate search engines and sends the query to them, in the appropriate format; finally the system receives the result sets from each of the search engines and merges them in order to present a unified list to the user, who is kept away from the details of each of the search engines that were used to gather the collection. A few specific problems arise from this approximation: search engine selection, query transformation and result fusion (Wang et al, 2003).

Dynamic search engines try do deal with web dynamics. Such search engines do not have any permanent index or database but instead crawl for their result sets at query time. Dynamic search engines assume that the user query is composed by a set of keywords plus a seed URL. Once a user query is defined the dynamic search engine starts crawling following link paths from the seed URL, fetching relevant documents, according to some measure, until the number of relevant documents, as specified by the

user, is fetched or a user set time limit is reached. This methodology is highly dependent on the seed URLs. Some approaches also work based on keyword queries only; in this case the keyword query is submitted to some general purpose search engine in order to get the seed URLs. A classifier guides the dynamic search engine's crawler, which is responsible for estimating the relevance of frontier URLs and fetched documents. Dynamic search heuristics are required in order to help the crawler to keep in track with the topic and avoid irrelevant regions of the Web in the vicinity of the seed URLs. *Fish search* (De Bra et al, 1994a; De Bra et al, 1994b) is one of the first such heuristics, which uses a binary measure of relevance (a certain URL is either relevant or not relevant at all). This algorithm as been improved later in *shark search* algorithm (Hersovici et al, 1998) with the introduction of a continuous relevance measure, assuming values between 0 and 1, instead of the binary function of the fish search algorithm, and also by making use of information contained in the anchor text of the links.

In *interactive search* (Bruza et al, 2000) a general-purpose search engine is wrapped in an interface that allows the user to navigate towards his or hers goal through a *query by navigation* process. The user poses a keyword query that is submitted to the underlying search engine. The result set of the query is not presented to the user; instead the relevant phrases are extracted from the result set and displayed to the user as candidate refinement possibilities of his or hers original query. The user can then select one of the phrases as the new query and this iterative process continues until the user is satisfied.

Automatic resource compilation may be described as the set of web mining and IR techniques used for retrieving and organizing document collections; we may say that it lies somewhere between search engine and topic directory systems. From our point of view the value of this kind of systems could be much improved if it allowed for automatic personalization and if it was able to automatically follow slightly drifts in user information needs. With these capabilities, an automatic resource compiler could be used as a long-term manager of user specific information needs, able of keeping track of the evolution of user interests and also able of keeping track of the natural evolution of the topic over time.

3 Automatic resource compilation

The Web is a vast repository of information with some characteristics that are adverse to IR: the large volume of data it contains; its dynamic nature; mainly constituted of unstructured or semi-structured data; irregular data quality and coherence.

The user also introduces some additional difficulties in the retrieval process: information needs are often imprecisely defined and hard to specify with the primitives at hand – usually keywords – resulting in ambiguous queries. Despite these difficulties the Web is being used as an information source by an increasing number of people with rather distinct background, motivations and information needs.

Whenever a user wants to keep track of some topic, organizing its references according to some structure reflecting the user informational view, an automatic resource compilation system can be of great value. We may imagine some potential interested parties: organizations, associations and specific interest groups (such as the one, which motivated this thesis, APPIA - Associação Portuguesa Para a Inteligência Artificial), commercial companies trying to gather information on their market (players, prices, products, client reviews, services, press news), news wire services, marks and patents control companies, students interested in some subject, to name a few.

Web mining phases are of major interest to automatic resource compilation systems:

- **Acquisition techniques** feed the system, periodically retrieving documents from the Web to update the resource. During this phase, automatic resource compilation systems submit the same queries in order to collect an updated image of the topics on the Web.
- **Pre-processing** is mandatory because it transforms all documents retrieved from the Web into a unique, standard representation, independently of their original format. This unified document view is required by the next phases and reduces the computational effort required to process the resource.
- During the **learning** phase the system automatically process the resources in order to learn topic specifications and user's information needs.

- **Analysis** of the correctness of the system responses based on implicit user feedback may help the system adapting to drift in user information needs, tracking slightly changes over time. This phase should be carried out without the need to explicitly require user intervention. Ideally, user feedback should be captured while the user explores his or hers resource by an imperceptible process.
- **Presentation** deals with user interface, including both query specification and resource presentation, which is, to our understanding, a fundamental aspect conditioning the quality of the system. Without an appropriate and easy tool to facilitate the exploration of large document collections the final goal of the system may never be achieved, even if all the previous phases have reached their goals. Extracting relevant information from a large set of presumably relevant documents is still a very demanding task, especially if the appropriate tools are not available.

In the next five sections we briefly review the state of the art of these areas and in the sixth section we will review some existing automatic resource compilation systems.

3.1 Acquisition

In automatic resource compilation, the first step is to fetch a data set, which in our case is a document collection retrieved from the Web. This is a problem of routing, which is essentially a particular filtering problem – given that the query is static and the base collection (the Web) is dynamic – that presents the result set ranked according to some measure of relevance (Baeza-Yates et al, 1999).

Our purpose is not to crawl the Web directly. Instead, we intend to use public search services as the providers of information and periodically query them in order to compile our resource and keep it up-to-date. Since single search engine's web coverage is not high and the overlap of their indexes is low (Chakrabarti, 2000; Bharat et al, 1998), it seems appropriate to rely on meta-search techniques in the hope of improving recall. According to (Selberg et al, 1995) no single search engine is likely to return more than 45% of the relevant results to a given query.

In the meta-search process a given query is sent to a set of search services (not necessarily search engines), result sets are gathered, unified and presented to the user

who benefits from several distinct web views without having to directly interact with any of the search engines.

3.1.1 Meta-search

Meta-search can be classified as external or internal meta-search (Aslam et al, 2001). *External meta-search* simply merges the lists of pages returned by the search services and presents the resulting ranked list to the user. *Internal meta-search* goes a little further and tries to incorporate the various sources of evidence – page text and HTML tags, in-links, out-links – available in the result sets returned by the search services. This implies downloading and processing retrieved pages locally, but may substantially improve the overall system performance (Selberg et al, 1995). Meta-search presents some specific problems:

- **Time lag in service results**: different search services may return different images of the same URL, which have been indexed at different moments in time, and the meta-search engine does not have a way to identify which one is the most recent.
- **Ranking algorithms**: different search services apply different ranking algorithms, usually unknown to the public. This makes merging the several result sets a difficult and erroneous task.
- **Heterogeneity of data sources**: search services may be heterogeneous, both in the querying interface and in the query output. Typically, search engines are somehow homogeneous since queries are, very commonly, sets of keywords and results are returned as an HTML web page or a set of HTML web pages.

Meta-search presents as well some potential benefits (Aslam et al, 2001; Selberg et al, 1995):

- **Single unified user interface**: the meta-search provides a centralized console to transparently interact with several distinct search services.
- **Rich feature set**: the meta-search system might provide an extended set of features, independently of each of the search services, if it downloads and locally processes the result set provided by the search services – internal meta-search.

- **Improved recall**: meta-search might improve recall – ratio of retrieved relevant documents to total relevant documents – as long as the base search services retrieve different relevant documents. Given the low intersection among distinct search service's databases this seems a reasonable hypothesis. (Selberg et al, 1995) report a loss of 77% in recall if a single search engine was used instead of their MetaCrawler.
- **Improved precision**: precision is the ratio of retrieved relevant documents to retrieved documents. Meta-searching may improve precision, thanks to the *chorus effect* (Aslam et al, 1998) – different retrieval algorithms retrieve many of the same relevant documents (which in some extend may contradict the previously stated improved recall argument) but different irrelevant ones (Ng, K.B., 1998); fusion techniques that give major importance to common documents may improve precision. In internal meta-search, precision might further be improved by reducing the number of irrelevant or broken links that reach the user; (Selberg et al, 1995) report that over 75% of the links returned by the base search services could be automatically considered irrelevant and removed before presenting the final result set to the user. These two last benefits, improving precision and improving recall, are not achievable simultaneously; there is a compromise between precision and recall, which makes infeasible increasing both with the same technique. Recall is improved with the union of two data sets while precision is improved with their intersection; nevertheless it is not possible to execute them both at the same time. When we refer to improving both precision and recall, we are comparing the answer from a meta-search process – which merges the answers from several search engines – with the answer of one of these search engines.
- **Consistency**: fusion of distinct search engine's answers is an averaging process, which filters out noisy influences from each singular search service, thus providing a more reliable global behavior.

3.1.2 The meta-search process

The meta-search process consists of three main tasks (Kobayashi, 2000):

- **Search services selection**: given a query, search services ability to produce relevant documents is determined and the set of the most promising search services is selected.
- **Query transformation and submission**: the query is prepared and submitted to the selected set of search services.
- **Result fusion**: the result sets from the several search engines are merged, duplicates are purged and the unified result set is ranked.

The first task aims at selecting the set of search services that best can answer a given query. The simplest meta-search engines allow for the user to indicate which search engines to use; in order to be effective this approach requires that the user knows which search services are more adequate for each query. SavvySearch (Howe et al, 1997) learns which search engines to query by keeping a *confidence* measure associated with each pair search engine and query. The confidence measure is increased proportionally to the number of returned links that were explored by the user – *visits* – and decreased if the search engine fails to return links – *no results*. Search engines are ranked based on this confidence measure and on recent search engine *performance* (measured from search engines accessibility and speed of response).

The difficulty of the query transformation task depends on the heterogeneity of the multiple sources available to the meta-search system. A meta-search system submits queries over multiple sources whose interfaces and capabilities may be quite different: the source might be a search engine based on the vector-space model and the query a simple list of keywords; or the search engine may only support Boolean queries requiring the knowledge of the specific query syntax; the search service might allow for the specification of queries based on different attributes. There is a high level of heterogeneity at the interfaces made available by search services and the meta-searcher must know how to translate a given information need specification to each source's syntax.

In the collection fusion task the meta-searcher has to deal with several distinct lists of ranked references and merge them into a single list, which should be ordered by document relevance. The main problems arise from the fact that the reference lists returned by current search engines are ranked with different, independent and unknown

algorithms and, the score of each item in the list is not guaranteed to be available. The main problem that arises is to order by relevance a multiple set of references where each set of references is previously ordered by some unknown measure, whether related to relevance.

3.1.3 Fusion techniques

Most of the fusion techniques in use are based in linear combinations of the individual scores from each of the search engines. This approach may be negatively influenced by the fact that scores in search engines are determined locally (to the search engine) based on measures and features that are specific to each search engine. Besides, the distribution of the relevance may vary from search engine to search engine – for instance the second ranked document from one search engine might be much less relevant than the fifth ranked document from another search engine, and just by looking at the ranking scores one cannot guess this. Another relevant problem may appear in search services based on vector space, and particularly on TF×IDF document models: since search services' document databases include different sets of documents (each one has its own view of the Web) it is possible that two distinct search services rank the same document differently even if they use the same ranking algorithm – it suffices that one of the databases includes more documents that contain a given keyword, present in the query, than the other. In such case, even if the meta-search system knows the ranking algorithm of the sources it cannot rigorously merge the results if it does not know each of the sources index database (Gravano et al, 1998). Under these circumstances the linear combination of these scores may lead to erroneous results.

The fusion of ordered lists might be done based on the *min*, *max*, *sum*, *median* or other statistics of each of the documents normalized scores over the set of lists (Aslam et al, 2001) – usually mapped to the interval [0, 1] in order to make scores from different sources directly comparable. It is also generally accepted that the number of sources that return a given document positively reinforces its relevance. (Fox et al, 1994) calculate the aggregate relevance of documents based on:

$$rel(d) = n_d^a \sum_{i=1}^{S} rel_i(d) \qquad [3.1]$$

with a = {-1, 0, 1}, n_d is the number of sources that returned document d, S is the number of sources and $rel_i(d)$ is the score of document d provided by source i.

If a = -1 then the aggregate relevance is the average over the sources that returned the document, a system called CombANZ. If a = 0 the result is the sum of the singular relevancies, CombSUM. When a = 1, the result is CombMNZ, a common fusion algorithm. A major problem with this form of combining results is that the aggregation is based on the relevance scores from the sources, which are not always available.

(Aslam et al, 2001) organize some fusion techniques according to the need of relevance scores and training: CombMNC requires relevance scores but does not need training data; linear combination models, originated from equation [3.2], are the most demanding, requiring both relevance scores and training data (to learn the coefficients c_i); Borda fuse is a technique that does not need neither relevance scores nor training data, fusing the results based only on the ranks of each of the reference lists.

$$rel(d) = \sum_{i=1}^{S} c_i \times rel_i(d) \qquad [3.2]$$

The Borda fuse technique is a voting model where each of the sources is a voter. Each voter ranks a fixed set of C candidates in order of preference; then it assigns C points to the first candidate, C-1 to the second, and so on; finally the candidates are ranked in order of the sum of points they have obtained from the voters.

The meta-search main difficulties are greatly due to the heterogeneity of the existing search services, which do not provide a common interface. The STARTS proposal (Gravano et al, 1997) is an attempt to normalize search service's interfaces in order to facilitate the tasks that a meta-search system performs. STARTS is a protocol that establishes what information needs to be exchanged between sources and meta-search systems in order to facilitate the meta-search task. It intends to be a general protocol, adaptable by any source, so it specifies a minimal functionality that is simple enough for all sources to comply with and also provides optional features that can be used by more sophisticated search services.

3.1.4 Resource refreshment

(Cho et al, 2000a) define *database freshness* as the average number of up-to-date elements, over all the elements in the database, and define *database age* as the average age computed over all the elements. An element's age is 0, if the element is up-to-date, otherwise equals the time that has elapsed since the last known modification. In order to improve freshness, while controlling bandwidth increase, it is necessary to define a refreshment strategy based on document change frequency. (Cho et al, 2000a) model document changes as a Poisson process and measure database freshness and age while applying three distinct refreshment strategies: *fixed-order* strategy, where all elements are analysed in the same order at fixed time intervals, *random-order* strategy, where all elements are synchronized in each cycle but the order is random, and *purely-random* strategy where the next synchronized element is selected at random from all the collection.

In order to simplify the synchronization process, the database elements may be associated to a frequency category, which characterizes the ideal frequency of refreshment. This way, instead of dealing with one independent synchronization process per document, it is sufficient to deal with one synchronization process per frequency category, thus simplifying the global process and reducing system workload.

3.2 Pre-processing

In this section we will consider the work that needs to be done in order to obtain a suitable web document representation, valid for the subsequent automatic learning phase. This phase includes data preparation and data representation.

3.2.1 Data preparation

In the text mining area, the pre-processing phase includes several steps that sequentially process documents in an attempt to eliminate non-informative features. This phase usually includes (Baeza-Yates et al, 1999):

- *lexical analysis* – eliminating punctuation, accents, extra spacing, convert to lower/upper case;

- *stop-word removal* – the process of removing irrelevant terms, depends on language and may also depend on query or topic of interest, which is hard to control; requires a list of all the words to eliminate (stop-words);
- *stemming* – reducing words to their semantic root; Porter algorithm (Porter, 1980) is probably the most well known stemming algorithm;
- *indexing* – define the index terms, the features that will be used for document modeling.

The application of these pre-processing tasks must be careful because the predictive power of words is highly dependent of the topic of interest (Chakrabarti et al, 1998). Another essential aspect to consider is the language in which the document is written, which determines the list of stop-words and the stemming algorithm to use. Besides, stop-word removal always reduces information contained in the document; to avoid this loss some search services, like CiteSeer (Lawrence et al, 1999a), do not remove any words from the documents to be indexed.

Web documents written in HTML usually require a previous step that removes HTML tags in order to obtain the document effective content text.

In the web environment some other problems have to be dealt with, as, for instance, document duplicates that may exist and which should be detected by the system. While exact match between two documents is easy to detect, with a checksum code, for instance, the near duplicates problem is harder to deal with. Duplicate web pages may have slight differences between them – a date, the home URL or editor, to name a few, which raise difficulties to automated duplicate detection.

It is possible to estimate with a reasonable confidence the characteristic distribution of a set of features when the number of training examples is substantially higher than the number of distinct features, which is not the case in text classification. In text mining the number of features is typically much larger than the number of available training examples and, if care is not taken undesirable over-fitting may arise. Feature selection is desirable not only to avoid over-fitting but also to keep the same level of accuracy while reducing feature space dimension, and consequently reducing the necessary computational effort. Feature reduction or feature selection techniques may be heuristic

– governed by linguistic principles or specific rules from the universe of discourse – or statistical.

Feature selection techniques, in text classification, are usually applied following the process (Chakrabarti, 2003):

1. compute, for each feature, a measure that allows to discriminate categories,
2. list features in decreasing order of that measure and
3. keep the subset of the features with the highest discriminative power.

The *curse of dimensionality* (Koller et al, 1996) requires reducing the feature space dimensionality in order to improve any reasoning over this space. The high feature space dimensionality can be reduced with techniques that might be categorized as feature selection or re-parameterisation techniques (Aas et al, 1999).

Feature selection attempts to remove non-informative words from documents in order to improve categorization effectiveness and reduce computational complexity while *re-parameterisation* is the process of constructing new features, as combinations or transformations of the original ones. Feature selection approaches are usually further classified as *wrapper* or *filter* techniques depending on whether they explore the learning algorithm to select the most appropriate features or not. Among common feature selection techniques we may include (Yang et al, 1997):

- **document frequency threshold** – uses the inverse document frequency, that is the number of documents where the feature is present, and eliminates features whose inverse document frequency fells off some pre-defined threshold; the application of this technique is simple, inexpensive and has been providing good results, although it requires some care in the specification of the threshold levels;
- **information gain** – features are ordered, in decreasing order of their information gain, and the most informative features are retained while the least informative are removed from the feature set;
- **mutual information** – measures the association between each feature and each category, based on a two way contingency table; features with the highest mutual information measure are selected;

- **chi-square** – uses the same contingency table as mutual information but performs a chi-square statistical test to infer on the independence between each feature and category; the major advantage of this method, compared to mutual information, is that, in this case, the test statistic is a normalized value allowing for comparisons across features for the same category;
- **term strength** – significantly different from the above, this method computes each term strength independently from the document category. It assumes that documents that share many common words are similar and further, that the common words are informative. This method estimates term importance based on the conditional probability of a term appearing in a certain document given that it appears in another similar document;
- **Markov blanket criterion** (Koller et al, 1996) – can be used to reduce the feature set, incrementally excluding the least relevant features until the reduced subset is satisfactory;
- **Latent semantic indexing**, LSI, (Deerwester et al, 1988) – is a re-parameterisation technique, which uses the singular value decomposition of the document×term matrix to reduce the dimension of feature space.

It should be stressed that words in the document are not the only features in web pages. Hypertext documents contain other types of features, such as links to and from other web pages and HTML tags, which may be explored, in isolation or in conjunction with others, constituting abstract features (Gani et al, 2001) which may hold significant predictive power.

3.2.2 Data representation

After the data preparation phase each document is reduced to its representative features – words and any other relevant characteristic that might have been considered – and the next step is to encode this view of the document into a specific format, ready for the learning stage. The learning algorithms require documents to be represented according to some model, specifically designed to facilitate the application of such techniques.

IR modeling can be organized according to the following taxonomy (Baeza-Yates, et al, 1999):

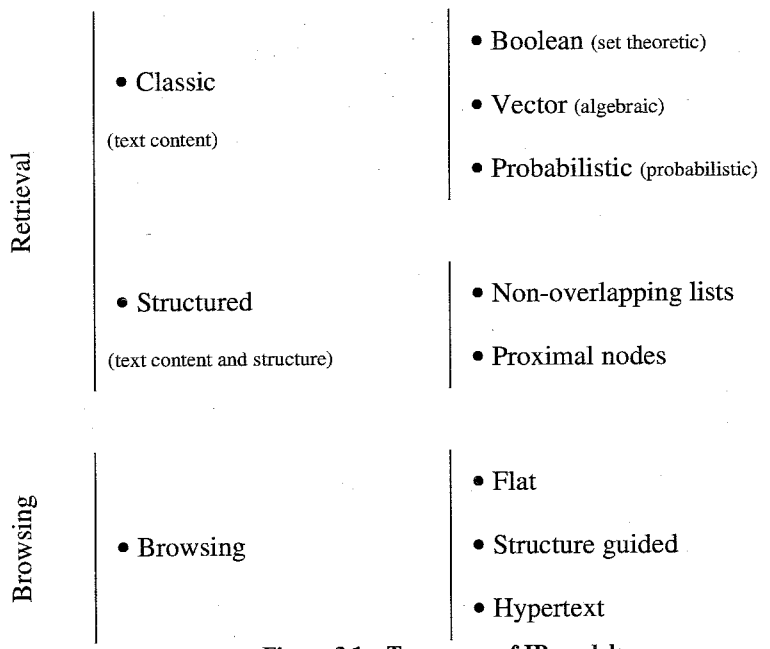


Figure 3.1 – Taxonomy of IR models

The user interaction with the Web may be classified as browsing or retrieval/searching, according to his or hers purpose while exploring the document space. These different user tasks may require different document models, has depicted in Figure 3.1.

For the description of document models we will apply the following notation (Baeza-Yates, et al, 1999):

$K = \{k_1, k_2, \ldots, k_t\}$ is the set of all index terms, k_i, with cardinality t, the number of index terms in the system,

$w_{ij} \geq 0$ is the weight associated with each index term k_i of a document d_j; $w_{ij} = 0$ if term k_i is not present in document d_j,

$\vec{d}_j$ *is the index term vector associated with the document* d_j*; it is represented by* $\vec{d}_j = (w_{1j}, w_{2j}, \ldots, w_{tj})$,

g_i is a function that returns the i-th weight of $\vec{d}_j$ vector, $g_i(\vec{d}_j) = w_{ij}$,

N is the total number of document in the collection,

n_i is the number of documents in which the index term k_i appears,

$freq_{ij}$ is the absolute frequency of term k_i in document d_j (the number of times the term k_i appears in document d_j),

f_{ij} is the normalized frequency of term k_i in document d_j, given by:

$$f_{ij} = \frac{freq_{ij}}{\max_l (freq_{lj})}$$

where $max_l(freq_{lj})$ is computed over all terms which are mentioned in document d_j.

3.2.3 Classic models

Classic models assume that each document is described by a set of representative keywords called index terms. An *index term* is a word, or phrase, appearing in the document, whose semantics help in remembering the document's main themes. The index terms are usually assumed to be independent.

Boolean model

A query is a Boolean expression using the three connectives "and", "or" and "not". The weights associated with index terms are binary, $w_{ij} \in \{0, 1\}$. There is no notion of partial match; each document d_j is either relevant or not relevant at all to a given query q; the lack of a relevance notion makes this a data retrieval instead of a real IR model.

Simplicity and the clean formalism are the main advantages of this model while its main disadvantages come from the fact that exact matching may lead to retrieval of too many – poor precision – or too few – poor recall – documents.

Index term weighting can improve information retrieval performance leading us to the vector space model.

Vector model

This model assigns non-negative weights to index terms in documents and queries. The degree of similarity of the document d_j with regard to the query q, $sim(d_j, q)$, is

evaluated as the correlation between the vectors $\overrightarrow{d_j}$ and $\vec{q}$, which can be, and usually is, quantified by the cosine of the angle between the two vectors:

$$sim(d_j, q) = \frac{\overrightarrow{d_j} \otimes \vec{q}}{|\overrightarrow{d_j}| * |\vec{q}|} = \frac{\sum_{i=1}^{t} w_{ij} * w_{iq}}{\sqrt{\sum_{i=1}^{t} w_{ij}^2} * \sqrt{\sum_{i=1}^{t} w_{iq}^2}} \qquad [3.3]$$

and, $0 \leq sim(d_j, q) \leq 1$

In the vector model index term weights are usually obtained as a function of two factors: the *term frequency* factor, TF, a measure of intra-cluster similarity, and an *inverse document frequency*, IDF, a measure of inter-cluster dissimilarity.

In vector space model documents are represented by vectors in a multi-dimensional Euclidean space. Each dimension in this space corresponds to a relevant term/word contained in the document collection. One given document collection is represented by a matrix W, with elements w_{ij} that represent the weight of the term k_i in document d_j. This model requires identifying the relevant terms – the index terms – and computing the corresponding document coordinates w_{ij}. For the computation of these weights there are several proposals (Aas, K, et al, 1999):

- **Boolean weighting**: the simplest approach is to let the weight to be 1 if the term occurs in the document and 0 otherwise. In this case the vector space model becomes the Boolean model, which may be viewed as a particular case of the former.

$$w_{ij} = 1, \text{ if } f_{ij} > 0 \text{ and } w_{ij} = 0, \text{ otherwise}$$

- **Word frequency weighting**: another simple approach is to use the frequency of the word in the document. This approach does not take into account the information value of the term (one term might appear very frequently but have a low information value).

$$w_{ij} = f_{ij}$$

- **TF×IDF weighting**: the previous two schemes do not take into account the frequency of the term throughout the entire collection. A well-known approach for

computing w_{ij} is the TF×IDF weighting, which assigns w_{ij} in proportion to the number of occurrences of the word in the document, the TF (term frequency) factor, and in inverse proportion to the number of documents in the collection for which the word occurs at least once, the IDF (inverse document frequency) factor.

$$w_{ij} = f_{ij} * \log\left(\frac{N}{n_i}\right) \qquad [3.4]$$

The coordinates of a document d are determined by two quantities:

Term Frequency TF(d,k) – number of times that the term k occurs in document d, normalized in such a way as to make it independent of document length. There are many distinct ways to normalize *TF(d, k)*, such as dividing by the total number of terms in the document or dividing by the total number of occurrences of the most frequent term in the document.

Inverse Document Frequency IDF(k) – not every term present in the lexicon is equally important, this measure *IDF(k)* tries to weight each term according to its discriminative power in the entire collection; information theory states that the more frequent the term is in the document collection the least informative it is. If D represents a document collection and D_k its subset of documents that contain term k, then a common way to compute *IDF* is:

$$IDF(k) = \log\frac{1+|D|}{|D_k|} \qquad [3.5]$$

There are also many ways of computing *IDF*, in its majority functions of $\frac{|D|}{|D_k|}$.

Document's d_j coordinate in term k_i, w_{ij}, is then given by:

$$w_{ij} = TF(d_j, k_i) \times IDF(k_i) \qquad [3.6]$$

- **TFC-weighting**: the TF×IDF weighting does not take into account that documents may be of different lengths. The TFC weighting is similar to the TF×IDF approach except for the fact that length normalization is used as part of the word weighting formula.

$$w_{ij} = \frac{f_{ij} * \log\left(\frac{N}{n_i}\right)}{\sqrt{\sum_{l=1}^{t}\left(f_{lj} * \log\left(\frac{N}{n_l}\right)\right)^2}} \qquad [3.7]$$

- **Entropy weighting**: based on information theoretic ideas.

$$w_{ij} = \log(f_{ij} + 1) * \left(1 + \frac{1}{\log(N)} * \sum_{j=1}^{N}\left[\frac{f_{ij}}{n_i} * \log\left(\frac{f_{ij}}{n_i}\right)\right]\right) \qquad [3.8]$$

where $\frac{1}{\log(N)} * \sum_{j=1}^{N}\left[\frac{f_{ij}}{n_i} * \log\left(\frac{f_{ij}}{n_i}\right)\right]$ is the average entropy of term i – this quantity is –1 if the word is equally distributed over all documents, and 0 if the term occurs in only one document.

The vector model is simple and fast providing better or almost as good results as other known alternatives. It's main advantages are: term weighting improves retrieval performance; it's partial matching strategy allows retrieval of documents that approximate the query conditions; it's ranking formula – cosine – sorts the documents according to their degree of similarity to the query. The index term independency assumption is probably its main disadvantage.

These models see the document as a bag-of-words, representing it as a vector where each coordinate refers to a given term and is calculated by its frequency in the document. These frequencies are then subject to one of several kinds of normalization, which determine several variants of the model.

Vector space model ignores the context in which terms are included, considering them as isolated and independent features; it is admitted that the relative positioning of terms has a weak discriminative power. Although some models try to explore this relative positioning of words, by considering phrases as dimensions – instead of isolated terms – or statistical measures of co-occurrence, the underlying framework is based on a vector representation.

Some new proposals, distinct from the traditional vector space models, try to explore sequences of characters – text is represented as a set of sequences of characters – using kernel functions to measure similarity between documents; the *string kernel* model.

Text can further be represented as sequences of words, which are linguistically more meaningful than characters, adapting string kernel function to *word kernel* functions, significantly reducing the problem dimension (Nicola et al, 2003).

Probabilistic model

Given a query q, the probabilistic model assigns to each document d_j the ratio $\frac{P(d_j\_relevant\_to\_q)}{P(d_j\_non\_relevant\_to\_q)}$ as a measure of its similarity to the query, which computes the odds of the document d_j being relevant to the query q. This model is also known as the *Binary Independence Retrieval* model.

Index term weights are binary: $w_{ij} \in \{0, 1\}$, $w_{iq} \in \{0, 1\}$. R is the set of document known to be relevant; $\overline{R}$ is the complement of R (i.e., the set of non-relevant documents). $P\left(R \mid \overrightarrow{d_j}\right)$ represents the probability of the document $\overrightarrow{d_j}$ being relevant to the query q, and $P\left(\overline{R} \mid \overrightarrow{d_j}\right)$ represents the probability that $\overrightarrow{d_j}$ is non-relevant to q.

The similarity, $sim(d_j, q)$, of the document $\overrightarrow{d_j}$ to the query q is defined as the ratio:

$$sim(d_j, q) = \frac{P\left(R \mid \overrightarrow{d_j}\right)}{P\left(\overline{R} \mid \overrightarrow{d_j}\right)} \qquad [3.9]$$

Applying Bayes rule, we obtain $sim(d_j, q) = \frac{P\left(\overrightarrow{d_j} \mid R\right) * P(R)}{P\left(\overrightarrow{d_j} \mid \overline{R}\right) * P\left(\overline{R}\right)}$ which, after some simplification and assuming independence of index terms, yields a key expression for ranking computation in the probabilistic model:

$$sim(d_j, q) \cong \sum_{i=1}^{t} w_{iq} * w_{ij} * \left(\log \frac{P(k_i \mid R)}{1 - P(k_i \mid R)} + \log \frac{1 - P\left(k_i \mid \overline{R}\right)}{P\left(k_i \mid \overline{R}\right)} \right) \qquad [3.10]$$

The probabilities $P(k_i \mid R)$ and $P\left(k_i \mid \overline{R}\right)$ are obtained through an iterative process.

This model ranks documents in decreasing order of their probability of being relevant, which is an advantage. Its main disadvantages are: the need to guess the initial separation of documents into relevant and non-relevant sets; just like the Boolean model, weights are binary; just like the vector space model, index terms are assumed to be independent.

Generalized vector space (vector space)

The classic vector model assumes independence of index terms, which means that the set of vectors representing each of the *k* index term form a basis, of dimension *t*, for the indexing space. Usually this independence is interpreted in a more restrictive sense to mean pair-wise orthogonality among the index term vectors i.e., to mean that for each pair of index term vectors, $\overrightarrow{k_i}$ and $\overrightarrow{k_j}$, we have $\overrightarrow{k_i} \otimes \overrightarrow{k_j} = 0$. In the generalized vector space model the index term vectors are assumed linearly independent but are not pair-wise orthogonal. In this model, index term vectors are composed as linear combinations of the orthogonal vectors that form the basis of the space, which are derived from the original index term vectors, and are called the *minterms*.

The generalized vector space model adopts as a basic foundation the idea that co-occurrence of index terms inside documents induces dependencies among these index terms. It is not clear that this framework provides a clear advantage over the classic vector space model since the usage of index term independencies to improve retrieval performance continues to be a controversial issue. However it certainly is more complex and computationally more expensive than the classic vector space model (Baeza-Yates et al, 1999).

Latent semantic indexing (vector space)

The ideas in a text are more related to the concepts described in it than to the index terms used in its description (Baeza-Yates et at, 1999). Thus, the process of matching documents to a given query could be based on concept matching rather than on index term matching. This could allow the retrieval of documents even when they are not indexed by query index terms. Latent semantic indexing (Deerwester et al, 1988) is an approach that addresses these issues.

The main idea is to map each document and query vector into a lower dimensional space, the latent semantic space, which is associated with concepts. The latent semantic space dimensionality should be large enough to allow fitting all the structure in the real document collection, and small enough to allow filtering out all the non-relevant representational details, which are merely noise to the retrieval process but are present in the conventional index term based representation. The dimension of the latent semantic space depends on the correlation between index terms; if the index terms are highly correlated then the latent semantic space dimension might be smaller.

Neural network (vector space)

A neural network (NN) for IR is composed of three layers: one for the query terms, one for the index terms and one for the documents themselves. The query term nodes initiate the inference process by sending signals to the index term nodes. Following that, the index term nodes might themselves generate signals to the document nodes. The document nodes in their turn might generate new signals, which are directed back to the index term nodes and might reach new index terms, not included in the original query terms; the index term nodes might then send new signals to the document nodes, repeating the process.

After the first round of signal propagation a NN model yields the same result as the TF×IDF model; further rounds implement a process analogous to a user relevance feedback cycle which naturally allows retrieving documents which are not directly related to the query terms.

Bayesian networks (probabilistic)

Allow for the combination of distinct sources of evidence in support of the rank for a given document.

3.2.4 Structured models

Classic models for text retrieval view a document in its most primitive form: for these models a text is a bag-of-words; the eventual text structure is not explored. Retrieval models, which combine information on text content with information on the document structure, are called structured text retrieval models. We are interested in representing

and processing hypertext documents. Hypertext contains a set of features, not found in plain text documents, which can reveal to be highly informative. Hypertext features on web pages include: HTML tags, URLs, IP addresses, server names contained in URL, sub-strings contained in URLs, links from the current page to other pages – *out-links* – and links from other pages to the current page – *in-links*.

At the end of the 1980s and throughout the 1990s, various structured text retrieval models have appeared in the literature. We refer two models that explore the logical structure of the text – which we have defined as the kind of structure imposed by the author – non-overlapping lists and proximal nodes, and another two that explore the physical structure – which we have defined as the kind of structure imposed by the technology used to produce the document – path prefixing and relational model.

Model based on non-overlapping lists

The text of each document is divided in non-overlapping text regions, which are collected in a list. Since there are multiple ways to divide a text in non-overlapping regions, multiple lists may be generated. These lists are kept as separate and distinct data structures. While the text regions in each list do not intersect, text regions from distinct lists may overlap. To allow searching for index terms and for text regions, a single inverted file is built in which each structural component stands as an entry in the index. Associated with each entry, there is a list of text regions as a list of occurrences. The query language allows the specification of queries concerning index terms and text regions.

Model based on proximal nodes

This model allows the definition of independent hierarchical indexing structures over the same document. Each of these indexing structures is a strict hierarchy composed of nodes. To each of these nodes is associated a text region. Two distinct hierarchies might refer to overlapping text regions. An inverted index for words/terms is associated to each hierarchy of structural components. The query language allows the specification of regular expressions (to search for strings), the references to structural components by name (to search for chapters, for instance) and a combination of these.

Path prefixing

One simple way of exploring information contained in the structure of HTML pages consists in prefixing each term – word or phrase – with the sequence of HTML tags that are associated with the term, separated by some given token or special character, such as ".". For instance "resume.publication.title.surfing" would represent the term "surfing" in the HTML code:

```
<resume>
    <publication>
        <title> Statistical models for web surfing </title>
    </publication>
    ...
```

This simple technique is enough, in its own, to guarantee significant improvements in some cases (Chakrabarti, 2003); however, it generates representations that are very specific and inflexible. Specific hypertext characteristics are better represented in a more uniform framework consisting of a set of relations and relational schemas to represent hypertext documents.

Relational model

Relations, such as:

Classified(domNode, label)
ContainsText(domNode, text)
PartOf(domNode, domNode)
LinksTo(scrDomNode, dstDomNode)

can be used to model web pages; over this relational model it is possible to apply inductive classifiers, such as FOIL that find rules from the relational schema.

3.2.5 Browsing

When the user is not interested in posing a specific query to the system but, instead, prefers to explore the document space, looking for interesting references, we may say that the user is browsing – not searching.

Models for browsing refer to the organization of the document space, rather than to the representation of each document.

Flat browsing

In this model the document space has a flat organization. The documents may be represented as elements in a list with a single dimension. The user randomly visits the documents in the list, eventually looking for correlations between neighbor documents or looking for keywords, which might improve his or hers original query – relevance feedback.

Structure guided browsing

The documents are organized according to a hierarchy of classes which groups documents covering related topics – topic directory. Some public search engines provide, besides the standard search interface, a hierarchical directory, which can be used for browsing – Yahoo! is probably the best known.

Hypertext model

A hypertext is a high level interactive navigational structure, which allows us to browse text non-sequentially on a computer screen. It consists basically of nodes, which are correlated by directed links in a graph structure. To each node is associated a text region which might be a chapter in a book, a section in an article, or a web page.

3.3 Learning

Automated learning techniques, generally adapted from the machine learning and statistics fields, may improve the performance and the functionality of automatic resource compilation systems in a wide variety of forms, ranging from document classification to user behavior modeling and IE.

Document classification or categorization is the task of assigning one or more pre-defined categories to documents. Typical classification problems are binary in nature: a given example either belongs to some specified class or not. In the web environment we are usually interested in sets of classes with more than just two distinct

classes: the classification problem is a *multi-class* problem. In data retrieval, the data domain is structured and so it is possibly to determine with certainty the set of positive examples related to a given query, which is specified as a set of conditions based on concrete features and corresponding values that must be verified by the examples to be considered positive. In IR, documents and queries are represented in high dimensionality spaces and two distinct documents may even have disjoint feature sets. The number of features included in the representation of a document is usually much smaller than the feature space dimension. In this scenario it is not possible to determine with certainty if a given document is relevant to a given query or not; consequently query responses are typically constituted by a set of documents that are sorted in decreasing order of their relevance to the query. Besides, web documents are often related to more than one category, the *multi-label* characteristic. The problem of classification in IR is a multi-class and multi-label problem. The common approach to multi-class, multi-label problems is to use a set of binary classifiers, each one responsible for determining the relevance of the document as to one specific category. The relevance scores of each one of the individual binary classifiers are then combined in order to provide the final answer.

Text classifiers, from text mining field, deal primarily with flat text documents and do not take advantage of other potentially relevant features present in web documents. Web pages contain other features, besides flat text, like hyperlinks, content of neighbors and metadata that might help to improve classifiers' performance.

Web page classifiers may be categorized according to the data they explore (Quek, 1998):

- Classifiers that use *data on the web page itself*: typically text mining algorithms applied to the web page content text – like Naïve Bayes and k-Nearest-Neighbors, for instance – and classifiers that try to explore HTML tags – like title and headings – which usually carry significant semantic value.
- Classifiers that rely on the *hyperlinks* among web pages: the Web is a kind of social network and its structure naturally reveals clusters or groups of strongly interconnected pages, which are usually associated to some kind of common

interest; these evidences can be leveraged in order to improve the classifiers performance.

- Classifiers that examine *metadata* about the page: metadata can be generated from previous evidence; for instance, the URL of the page may contain some patterns that might help the classification task.

This is merely a conceptual partition of the evidential sources for the classification task, which does not prevent classifiers that combine information from several sources.

Several text mining algorithms, derived from the machine learning field, have been applied to the web document classification task. Although the performance of the text classifiers depends heavily on the document collection at hand (Yang, 1999), some classifiers, particularly Support Vector Machines and k-Nearest-Neighbors seem to outperform others in the majority of the domains. (Joachims, 1998) points out a few properties of text – high dimensional feature spaces, few irrelevant features, document vectors are sparse, most text categorization problems are linearly separable – that justify why Support Vector Machines should perform well for text categorization.

3.3.1 Flat text classifiers

When applied to web pages, classical text mining methods treat each page independently, ignoring links between pages and the class distribution of neighbor pages.

Rochio

Rochio's algorithm (Joachims, 1997) is a classic method for document categorization in IR. In this method the training examples are used to build a prototype vector for each class. The prototype vector for each class is computed as the average vector over all the training document vectors that belong to the class. A new document is classified according to the distance measured between the document vector and the class prototype vectors.

Naïve Bayes

Naïve Bayes methods (Joachims, 1997) use the joint probability of words/terms, k_i, and categories, v_j, to estimate category probabilities given a document (bag-of-words), $P(v_j \mid k_1, k_2, ..., k_n)$. Dependences between words are ignored, that is the method assumes that the conditional probability of a word given a category is independent of the conditional probability of any other words given the category; this is the naïve assumption. Assuming word independence, the conditional probability of document d, given class v_j can be obtained by:

$$P(d \mid v_j) = \prod_i P(k_i \mid v_j) \qquad [3.11]$$

Given a new document, the algorithm computes a probability vector, which contains the document relevance scores regarding each one of the classes, and assigns to the document the most probable class v_{NB}:

$$v_{NB} = \arg\max_j P(v_j) \prod_i P(k_i \mid v_j) \qquad [3.12]$$

The algorithm application requires, essentially, the definition of a way to estimate the conditional probabilities $P(k_i \mid v_j)$, which may be obtained by:

$$P(k_i \mid v_j) = \frac{n_k + 1}{n + \mid vocabulary \mid} \qquad [3.13]$$

In equation [3.13], n is the total number of words in all training documents belonging to category v_j, n_k is the number of times that word k_i appears in that n word set and $\mid vocabulary \mid$ is the total number of distinct words present at the training document collection, it is the lexicon cardinality (Fang et al, 2001). The a-priori probabilities of the classes, $P(v_j)$, may be estimated by the relative frequencies of training documents. The value 1, added to the numerator, and $\mid vocabulary \mid$, added to the denominator, are necessary to avoid the effect that one word k_i not appearing in the training documents of class v_j would have, forcing $P(d \mid v_j) = 0$.

k-Nearest-Neighbors

k-Nearest-Neighbors (kNN) is an instance based classifier which has been demonstrating good performance in pattern recognition and text categorization problems (Yang, 1997; Yang et al, 1994). This method classifies a document based on

the characteristics of its closest k neighbor documents. Documents are represented in the traditional vector space model and the similarity measure is the cosine between document vectors. The categories that have a relevance score above a given threshold are assigned to the document. (Yang et al, 2002).

Decision Trees

Decision trees are decision structures built from a base, root node, which contains all the training examples (Witten et al, 2000; Lewis et al, 1994). The set of examples in any specific node are subdivided in a partition represented by its descendant nodes. This split is made with the objective of minimizing the diversity of categories present at each node and it is carried out until no further reasonable improvement is possible. At a given node, the split is made as to assure that the sum of the diversities at the child nodes is much less than the diversity at the present node without splitting. The goal is to maximize:

$$diversity(before\_split) - \sum_i diversity(child_i) \qquad [3.14]$$

Care must be taken into account in order to avoid over-fitting, which is usually made by pruning the decision tree. There are two common pruning approaches: *post-pruning*, or backward pruning, and *pre-pruning*, or forward pruning (Witten et al, 2000). In the pos-pruning approach the tree is expanded as much as possible and then it is pruned by removing nodes, which cannot significantly improve the homogeneity, hence the predictive power, of the tree. In pre-pruning approach one tries to evaluate, during the tree construction process, when to stop developing sub-trees. The nodes at the bottom of the tree are the leaf nodes. Any training example belongs to a certain leaf node. Each leaf is then assigned to a class and the error rate of the leaf is the probability of examples in that leaf node being misclassified. The global tree error rate is a weighted sum of all the leaf nodes error rates.

A key issue in decision trees is to decide which features allow for the best split at each node, the one that generates the most homogeneous partition, and the definition of the diversity measure to use.

One of the most common diversity measures is entropy:

$$\sum_{j=1}^{K} p(c_j \mid t) \times \log p(c_j \mid t) \qquad [3.15]$$

Where $p(c_j|t)$ is the probability of a training example belonging to class c_j given it is located in node t, which can be estimated by the relative frequency of examples of the class c_j that fall in the node t:

$$p(c_j \mid t) = \frac{N_j(t)}{N(t)} \qquad [3.16]$$

$N_j(t)$ is the number of examples of class c_j in node t and $N(t)$ is the total number of examples in node t.

Several algorithms are available to grow decision trees. CART, CHAID and C4.5 are common approaches (Aas et al, 1999).

CART algorithm builds a binary decision tree by splitting the training examples at a given node according to a function of one single feature. The main task is to decide, at each node, which feature performs the best partition and what is the split value.

C4.5 algorithm builds decision trees with various children per node; it is not limited to binary trees (two children per node) as is CART.

CHAID is also a popular algorithm but it is limited to categorical features; thus, if the domain under study has numeric attributes then they must be previously discretized.

Support Vector Machines

Support Vector Machines (SVM) (Joachims, 1998) is based on the intuition that a hyper-plane that is close to several training examples will have a bigger chance of making erroneous decisions than one which is as far as possible from all training examples. The SVM algorithm is a binary classifier that defines a maximum margin hyper-plane between the convex hulls formed by the training examples of each class. The maximum margin hyper-plane is the one that is as far away as possible from both convex hulls – it is orthogonal to the shortest line connecting the hulls, intersecting it half way. This hyper-plane may be defined as a function of the training examples that are closest to it, the support vectors.

$$x = b + \sum_i \alpha_i c_i \vec{d}_i \otimes \vec{d} \qquad [3.17]$$

where the sum is over the set of the i support vector documents, d_i is one support vector, d is the document to classify and b an α_i are numeric parameters to be determined by the learning algorithm. SVM, like all discriminative classifiers, is non-parametric; it does not assume any presupposition on the data distributions besides the trivial presupposition that training and testing examples all come from the same population, thus assuming identical distributions.

3.3.2 Exploring other features besides text

Some algorithms explore other sources of evidence, besides plain text, that are present in hypertext collections. (Yang et al, 2002) define five types of regularities that might be present in a hypertext collection: *no regularity* – the only place that has relevant information about the class of the document is the document itself, *encyclopaedia regularity* – documents with a given class only link to documents with the same class, *co-referencing regularity* – some, or all, of the neighboring documents belong to the same class but this class is distinct from the document class, *pre-classified regularity* – the regularity is present at the structural level where a single page (hub) points to several pages which belong to the same class – and *metadata regularity* – metadata is available from external sources and can be explored as additional sources of evidence generating new features.

The authors then define the types of features that should be used in order to improve the classification task of documents belonging to each of these regularities. However, there is no suggestion as to how to previously determine the type of regularity that is present in a given document collection. According to their experiences different classifier designs should be considered, depending on which of the above regularities holds. With no regularity we would not expect any benefit from using hyperlinks and the suggestion is to use flat text classifiers, based on the text of the document itself. Encyclopaedia regularity suggests augmenting the text of each document with the text of its neighbors, thus increasing the number of words related to the topic present in the document representation. In the case of co-referential regularity, the text of the document should also be augmented with the text from its neighbors but these imported words should be treated as if they came from a different vocabulary, for instance prefixing them with a specific tag. If the collection has a pre-classified regularity then there is no need to look

at the document text, it suffices to look at the pages that link to it and determine their class. When external sources of information are available that can be used as metadata, metadata regularity, we can collect them, possibly using IE techniques.

(Chakrabarti et al, 1998) also tests several feature sets, similar to the ones suggested by (Yang et al, 2002): *local text, local text concatenated with all neighbors text, local text plus neighbors text prefixed with discriminative tags*, to conclude that naïve use of terms in the link neighborhood of a document can even degrade performance. (Yang et al, 2002) have reached the same conclusion, which seems consensual. Although the use of extended sets of features available in hypertext collections – including text from hyperlink anchors, the full text from neighbor documents, HTML tags, category distribution over a linked neighborhood, metadata available from external sources – may provide rich information for the classification task, it is not guaranteed that the use of such features will improve performance, which in general is dependent of the specific document collection.

In document classification, the class labels are frequently organized in a taxonomy where classes are associated through inheritance. This means that if a document belongs to a given class then it also belongs to all their ancestor classes. This specialization/generalization structure may also be explored, improving the classification of documents. Techniques that explore this structure are commonly referred to by *hierarchical classification* (Chakrabarti, 2003).

3.3.3 Performance measures

The evaluation of the classifier performance is an important issue of classification tasks. This evaluation can be based on several measures, each of which evaluates some specific aspect of the classifier. Among them we will refer a few common metrics: recall and precision, which assume a very important position in web mining, F-measure, which combines recall and precision in a single measure, accuracy and error and novelty.

Recall is defined as the ratio between the number of documents correctly classified and the total number of documents in the category.

Precision is defined as the ratio between the number of documents correctly classified and the total number of documents classified in the category – correctly plus erroneously.

Usually a classifier exhibits a trade-off between precision and recall, so these measures are negatively correlated: improvement in recall is made at the cost of precision and vice-versa. It is frequent to have text classifiers operating at the break-even point. The *break-even point* of a classification system is the operating point where recall and precision have the same value.

The *F-measure* combines recall and precision in a unique indicator:

$$F_\beta = \frac{(\beta^2 + 1) \times precision \times recall}{\beta^2 \times precision + recall} \qquad [3.18]$$

where β is a parameter allowing different weighting of precision and recall. At the break-even point recall, precision and F_1 have the same value.

Accuracy and error are complementary measures of the error probability of the classifier given a certain category. *Accuracy* is defined as the ratio of the number of correctly classified examples by the total number of evaluated examples, while *error* is defined as the ratio of the number of incorrectly classified examples by the total number of evaluated examples.

These measures are defined for binary classifiers. To measure performance in multi-class problems there are two common approaches to aggregate these measures evaluated at each singular category: macro-averaging and micro-averaging. *Macro-averaged* measures are obtained by first computing the individual scores, for each individual category, and then, averaging these scores to obtain the global measure. *Micro-averaging* measures are obtained by first computing the total number of documents correctly and incorrectly classified, for all of the categories, and then using these values to compute the global performance measure, applying its definition.

There is an important distinction between these two aggregation algorithms: macro-averaging gives equal weight to each category while micro-averaging gives equal weight to every document.

Novelty is defined as the fraction of the relevant documents retrieved which were

unknown to the user (Baeza-Yates et al, 1999):

$$V = \frac{N_u}{N_u + N_k} = \frac{N_u}{N} \qquad [3.19]$$

where N_u is the number of relevant documents unknown to the user and N_k is the number of relevant documents known to the user.

3.3.4 Unsupervised, Supervised and Semi-Supervised learning

The application of machine learning techniques to classification problems generally requires two distinct stages: the learning stage – when the classifier is learned, that is the algorithm builds a model of the concept to be learned based on training and testing data – and the classification stage – when the model is applied over unseen examples in order to classify them. When the training and testing data sets are fully labeled we are in presence of a *supervised* learning problem; on the other extreme, if none of the training examples is labeled, we are in presence of *unsupervised* learning, or clustering, when the training data is partially labeled, we are in presence of a *semi-supervised* learning problem. (Bennet et al, 1998) further classify semi-supervised learning problems as either *semi-supervised clustering* problems, when the number of labeled examples is small when compared to the dimension of the training data set, or transduction problems, when the number of labeled examples is large when compared to the training data set dimension. The *transduction* problem is to estimate the value of a classification function at a given point, which opposes to the standard *inductive learning* problem of estimating the classification function at all possible values and then using the fixed function to deduce its value at a given point.

In the unsupervised setting there is no prior knowledge on labels, neither on the labels of each document nor even on the labels themselves. Clustering algorithms organize documents in homogeneous groups, based on their similarity, forming partitions of the data set that minimize intra-group variance and maximize inter-group variance. This setting does not seem very adequate to our particular problem, which admits the existence of a previously defined set of labels.

The supervised setting requires the full data set, from where the training and testing samples are obtained, to be labeled or, at least, that there is a large number of labeled

examples from each class. This is the major drawback of this setting, concerning web page classification, because of the high cost of labeling web pages. The process of manually assigning labels to text documents is both time consuming, inaccurate and subject to incontrollable factors arising from human nature (Macskassy et al, 1998) – two users with the same skills may classify the same page differently or the same user may classify the same page differently at different moments of time.

Semi-supervised learning techniques are particularly interesting when the process of labeling training data is expensive and time consuming, as is the case of labeling web documents. In this setting, classifiers are wrapped by some method in order to take advantage of unlabeled documents. Several approaches have been proposed to solve the semi-supervised learning problem:

- **Bootstrapping** is a simple iterative method. At each iteration, labeled examples are used to learn a classifier. This classifier is applied to label unlabeled examples; those where there is enough evidence in favour of a certain label against the others are added to the labeled set. The algorithm proceeds iteratively until convergence (Chakrabarti, 2003; Jones et al, 1999; Riloff et al, 1999).
- Usage of **Expectation-Maximization** (EM) to estimate maximum a posterior parameters for a generative text classification model (Nigam et al, 2000).
- **Co-training** (Ghani, 2001; Blum et al, 1998) is a supervised learning method, particularly useful when it is required to combine sources of evidence originated form very distinct spaces – particularly if they have rather different dimensions and scales, which may bias the aggregation of measures from these distinct sources. With co-training distinct classifiers keep disjunctive, independent feature sets and their estimates are never directly consolidated; instead this method uses the estimates of one classifier to train others. The application of co-training requires the existence of distinct and independent sets of features – (Blum et al, 1998) apply co-training to web document classification, a field where the features are naturally separable into disjoint sets, such as text in the page itself and words appearing in the in-links to the page, and two classifiers, one for each feature set, can be built.

- **Error Correcting Output Code** (ECOC) is a method that converts a K multi-class problem in a set of L binary problems (Witten et al, 2000). Any binary classifier can then be used to learn these L problems. ECOC assigns to each class a unique binary string, the *code word*, where each bit is estimated from one of the binary classifiers. The predicted class is the one whose code word is closest to the code word produced by the set of the L binary classifiers. The distance between code words is computed by the Hamming distance, which is calculated as the number of different bits in both code words. (Ghani, 2001) describes a method for semi-supervised learning that uses the ECOC method bundled with co-training techniques in order to learn the binary classifiers.
- **Transduction** (Seeger, 2001; Joachims, 1999; Gammerman et al, 1998) is naturally related to instance-based learning. In transduction we are interested in the classification of a particular example rather than in a general rule for classifying future examples. S^3VM (Bennet et al, 1998) improves SVM for the semi-supervised setting.

In the special case where there are no negative examples – only labeled or unlabeled examples are available – a generic method, consisting of two steps, may be followed (Liu et al, 2003b): the first step consists in identifying a set of reliable negative examples from the unlabeled set; in the second step, a battery of classifiers is built, iteratively applying a classification algorithm, and one of them is selected. These two steps together can be seen as an iterative method of increasing the number of negative examples while keeping the positive ones correctly classified. Several algorithms where proposed (Liu et al, 2003a; Li et al, 2003), differing on the specific techniques they apply at each of these steps.

Coaching (Tibshirani et al, 1995) is a technique that may also be applied in the semi-supervised setting (Seeger, 2001). The coaching technique applies when we are in the presence of two distinct sets of predictive variables, say X and Z, which are used to predict the label (in the case of classification), Y, of examples, but only one of these sets, say X, will be available on the future examples to classify. Coaching techniques use the set of predictive variables Z to coach the set X how to improve prediction of Y in the absence of Z.

In the previous methods, just referred, the training set is obtained by a process that is exogenous to the learning process. In opposition to this passive learning process, *active learning* (Callison-Burch, 2003; Cohn et al, 1996) gives the learner the ability to select which examples should be included in the training set. Active learning reduces the amount of labeling that needs to be done though selective sampling of the unlabeled data. In active learning the learner examines a collection of unlabeled examples and selects the most informative, requiring the human annotator to label the selected examples, and iteratively re-trains on the augmented set of labeled training examples.

(Seeger, 2001) discusses the hypothesis of considering the semi-supervised learning setting as one of *missing or uncertain data.* He rejects the hypothesis, concluding it is probably not a good approach, given the specialization of the label attribute.

3.4 Analysis of resources

In current search engines it is typical that the user specifies his or hers information need with a set of keywords. Although this form of interaction is very simple and relies on universally understood specification language and protocol, keywords are weak primitives to specify an information need (with high imprecision and subject to each user's interpretation), resulting in ambiguous, incomplete or excessive specifications and, consequently, poor result sets. These result sets are analysed and validated by the user. The user implicitly analyses the retrieval system performance when he explores the result set. It is reasonable to expect that if a user archives or downloads many of the returned documents, then they probably have some relevance to him or her; and, on the other hand, if the user does not download a given document it is probable that this particular document is not very relevant to the user information need. The user may even be required to explicitly indicate to the system his or hers interest in a given set of documents. This feedback from the user, either explicit or implicit, might be used by the information retrieval system in order to improve its performance, particularly as to the relevance of the returned documents; this problem is known as *relevance feedback.* Relevance feedback techniques usually produce some query modifications, by adding, removing or changing terms, originating internal queries that are different from the original user query and which are expected to be more representative of the user's

information need. The common approach is to retrieve an initial set of relevant (and eventually also non-relevant) documents and discover correlated features that can be used to narrow down the original query, improving precision.

Rochio's method (Chakrabarti, 2003) reflects user feedback through a simple technique that simply adds to the query vector a weighted sum of all the relevant document vectors and subtracts a weighted sum of all the irrelevant document vectors:

$$\vec{q}' = \alpha\vec{q} + \beta\sum_{D^+}\vec{d} - \gamma\sum_{D^-}\vec{d} \qquad [3.20]$$

where q and q' represent the query vector, before and after query modification, respectively, D^+ and D^- represent the set of positive and negative document vectors, respectively, and α, β and γ are adjustable parameters.

The set of documents initially returned may be classified manually, by the user, or automatically. In the last case the process is known as *pseudo-relevance feedback*. In pseudo-relevance feedback the set D^+ is usually a subset of the positive documents, consisting only of the most relevant, and γ is usually set to zero, that is, only the positive reinforcement is used.

(Mitra et al, 1998) describe a pseudo-relevance approach, which discovers correlated features exclusively from the most relevant documents, based on the intuition that a more effective query modification can be achieved by using only the documents that are closest to the original query.

(Glover et al, 2001) propose a relevance feedback system, in operation at the Inquirus meta-search engine, which is aimed at improving recall, given a minimum precision threshold, and whose results are intended to belong to some specific category besides being relevant to the query keywords (for instance the user may be interested in documents about "artificial intelligence", the keywords, but only those that are about "international conferences", the category). They use an SVM classifier, which explores, besides flat text, several other structural and metadata features to obtain with high accuracy a list of positive examples, which are then used to extract several candidates to query modification. The pairs (query modification, search engine) are ranked based on the number of valid documents returned in response to some representative test queries submitted to the search engines.

3.5 Presentation of resources

In IR systems the user typically interacts with the system in two distinct moments: when specifying his or hers information need and when analysing the results.

Query specification requires the user to describe information needs as a set of keywords and, eventually, some other characteristics to be met, such as: the web domain to explore and the document language and file format. In traditional search engines the user specifies the query in one single step, while interactive query systems (Bruza et al, 2000) require the user interaction during the initial tuning phase where the user iteratively refines his or hers query by following suggestions from the system.

The analysis of the results is a more challenging, interesting and potentially valuable aspect of search services, however, public search services have not been given it much of importance. Results returned from a search engine are usually presented to the user as a flat ranked list of documents, which raises difficulties if the user needs to get an overview, a global mental picture, of the entire document collection and its intrinsic structure.

Retrieval systems are a black box from the user point of view – the user submits a query and receives an answer without having any knowledge on the decisions, processes and data analysed to provide the answer – which obstructs the analysis of the returned document collections. Besides, in a flat list, users do not have a way of analysing a global perspective of the entire collection neither of analysing a single object particularities while still keeping a global image of all the collection.

Several difficulties may arise from these facts (Cugini et al, 1996; Olsen et al, 1992):

- the user probably does not know how the query relates to the source database so he does not know whether the query has an appropriate scope;
- the user probably is not aware of the structure of the source database;
- the user does not understand why a given document is included, or excluded, from the answer, since he does not know the ranking algorithm used;

- ordering the result set in a flat list hides the structural organization and any relations that might exist among documents, and between the documents and the query, eventually based on other characteristics besides relevance;
- relevance is highly dependent on user specific needs, thus it is an ambiguous metric to sort documents;
- the returned list may be large and the user has to scan through it, deciding which documents are interesting and which are not;
- the user may find an interesting document and wish to look for similar ones.

A visualization tool might help in solving these problems, providing user with functionalities that allow for selecting valuable information from large document collections without much effort, mainly because it will help in:

- understand large collections of objects. The flat list approach is very restrictive when the object collection is large (Carey et al, 2000); this approach subordinates the user to a slow, intensive mental process of reading through the reference list while in a graphical system the user just has to recognize patterns on a visual display, which is a much faster and less demanding process;
- getting an overview of an object collection and allowing analysing any particular object without loosing a global perspective of the full object collection and further being able to get a global view of the entire collection, which stresses relationships among similar objects.

In a traditional library the user can wander around, looking at how the books are stored, reading titles and authors, opening a few books that seem especially relevant. In a very short period of time the user gets some idea of the coverage, contents and organization of the library. It is then much easier to realize if his or hers information need may be fulfilled within the library's resource and also to locate relevant documents in it. Search services usually do not provide the user with any means that allow obtaining such an overview of the retrieved document collections. A flat list is not the most appropriate framework because the user looses all structural information embedded in the collection and so will not be able to realize how the entire collection is organized. Visualization

techniques – human visualization is the process of forming a mental image of a domain space (Olsen et al, 1992) – might help.

Graphical descriptions can show structural patterns, clusters and relationships among a number of objects in a very economic manner, which is much quicker and less demanding in memory requirements, than a linguistic approach. Given a large set of data points on several variables, it is impossible for a human user to look at them in a flat list display and perceive any relationships among them; an adequate graphical representation of the same set can be easily interpreted by the same human user.

When documents are represented in the vector space model, an immediate visualization approach seems appropriate: define a multidimensional Cartesian coordinate system and position document vectors in that space. Each element of the feature set originates one dimension and the weight of that term in the document is used as the document coordinate in the corresponding dimension. Document vectors in this multidimensional space can then be projected into a two or three dimension space. The position of the document is representative of its content and other attributes of the icon used to represent the document in the visual display, such as colour and shape, could be used to represent other relevant characteristics of the document. One major drawback from this approach is that the loss of information due to the projection might hide the relationships and patterns of interest since the required reduction in the number of dimensions is dramatic. Nevertheless IR requires visualization techniques that might provide some desirable functional characteristics (Olsen et al, 1992):

- the position and other graphical attributes of a document's icon should intuitively give information on a document;
- data reduction may be necessary, but the most important document attributes, as defined from the user (or automatically inferred), should be retained in the display;
- the display should give an overview of the complete document collection retrieved, as seen from the user's perspective;
- users should be able to identify single documents for retrieval of additional information;

- users should be able to control the display interactively, for instance, by relating document attributes to graphical features and by viewing documents from a new perspective.

Alternative post-retrieval visualization techniques, besides the common ranked flat list, may be categorized, according to their main goal, in the following categories (Zamir et al, 1999):

- those that stress inter-document similarities, usually based on document content, and which help the user getting an overview of the full document collection as well as visualizing clusters of topically related documents, thus helping to rapidly find interesting documents and locating similar documents to a given one;
- those that aim to display additional information about retrieved documents, which might be predefined document attributes (such as size, date, age, author or source) or user-specified attributes (such as predefined categories).

Although public search services rarely explore post-retrieval visualization techniques, there are many proposals present in the literature (some of them are explored in search services like Envision and webVIBE). We briefly review some of them:

- (Carey et al, 2000) claim that an **unified approach** – allowing the user to explore the document collection through several different visualization paradigms, which should be integrated, in order to allow for cross comparisons and evaluations – will be valuable. They include Sammon Maps, Tree-Map visualization and Radial visualization in this consolidated framework.
- **Sammon maps** try to represent *n*-dimensional objects in a 2-dimensional space while attempting to preserve the pair-wise distances between objects; instead of singular documents, cluster's centroid vectors are used as main objects – computed as the normalized arithmetic mean over document vectors belonging to the cluster – which can be detailed through a drill-down operation. Each cluster is represented as a circle, whose radius and colour are indicative of the number of documents that it contains, and is labeled with its most frequent keyword. The distance between any two circles in the display is indicative of the similarity of their respective clusters.

- The **Tree-Map** visualization represents clusters as rectangles arranged in such a way as to fit inside a larger rectangle. The area of a rectangle is proportional to the size of the cluster it represents. Similar clusters are grouped together in *super-clusters*. Black lines delimit Super-clusters while white lines separate inner clusters. Cluster rectangles have different colours and are labeled with their most frequent words. Several functionalities are available for the user to explore the document collection; it is possible, for instance, to view a list of cluster keywords, listed by descending order of cluster document frequency.
- **Radial visualization** is a common paradigm where keywords are represented as nodes, which are homogenously displayed, usually around a circumference, in the display space. Documents are then placed as if they were connected to the relevant keyword nodes by forces that attract them proportionally to the weight of the word in the document.
- **VIBE system** (Olsen et al, 1992) explores this radial visualization paradigm by defining *Points Of Interest* (POI) as a set of representative keywords working as anchor nodes relatively to which documents are placed in a bi-dimensional space. Based on the keywords defined for each POI, the system computes a score for the document on that POI as the sum of the frequency counts over the POI keywords. These scores determine the influence from each POI on a document. Documents are positioned by a positioning function that relies on the document score vector and on the POI coordinates.
- (Spangler et al, 2002) represents relevant categories, associated to the query result set, in a **Radial graph**, which allows the user to understand which categories are strongly related to the query, and associates this view with a binary tree that allows to refine a query, once the user as chosen a given category. This binary tree divides the set of documents present at a given node (initially the documents that belong to a given category) in two subsets based on the keyword that most evenly divides them.
- (Viji, 2002) describes a visualization tool that uses **springs** to represent documents semantic correlation. The documents are represented by masses interconnected by springs. The length of a spring connecting two documents is proportional to their

correlation. Documents are initially placed at random positions in the view space and the "correlation springs" exert their force to drive documents to their equilibrium positions. The stronger the correlation between two documents, the closer they will be their equilibrium positions in the view space. Correlation between two documents is based on two components: textual content (measured by the dot product of both document vectors) and link based information (which explores direct links, that is links between both documents, and shared links, that is out-links to common target documents or co-cited documents).

- **Document Spiral** paradigm, proposed in (Cugini et al, 1996) displays documents along a spiral in a two dimensional space. Documents represented at the centre of the spiral are the most relevant and relevance decreases as one travels along the spiral. They refer to *keyword sliders*, a common functionality in this type of interface, also present in VIBE for instance, that allows the user to enhance keywords relative importance and analyse the impact in the collection display.
- **Kohonen maps** apply neural network principles to self-organize document collections into a *galaxy* – a galaxy is simply a group of clusters represented in a two-dimensional space, one of the simplest forms of cluster mapping (Chewar et al, 2001).

3.6 Comparative study

An automatic resource compiler is a system that, given a topic, seeks and retrieves a list of the most authoritative web documents, as perceived by the system, for that topic (Chakrabarti et al, 1998). This is a very broad definition, under which many distinct types of systems may be considered, including, for instance, search engines. In our work we are interested in an automatic resource compiler, which, given a topic, has the responsibility of building and managing a collection of relevant documents in a continuous effort to keep the collection up-to-date.

Many automatic resource compilation systems and methodologies have been proposed in the last few years, exhibiting many interesting ideas and characteristics.

iVia (http://infomine.ucr.edu/iVia) (Mitchell et al, 2003) is an open source virtual library system supporting Infomine (http://infomine.ucr.edu), a scholarly virtual library collection, from University of California, Riverside, of over 26000 librarian created and 80000 plus machine created records describing and linking to academic Internet resources. It is a hybrid system that collects and manages resources, starting with an expert-created collection that is augmented by a large collection automatically retrieved from the Web. iVia automatically crawls and identifies relevant Internet resources through focused crawling and topic distillation approaches.

Personal WebWatcher (Mladenic, 1999) is a system that observes users behavior, by analysing page requests and learning a user model, and suggests pages potentially interesting to the user. The system operates offline, when learning user models, and at query time, when proposing interesting pages to the user. When a user downloads a web page the system analyses out-links and highlights those that seem interesting given the specific user model. Similar agents may communicate and exchange information on similar users, leveraging particular experiences, through a collaborative or social learning process. The system learns by exploring requested pages: a page requested to the server is considered to be a positive example of user interest and any links not selected is considered to be a negative example. In this way user relevance feedback is obtained without the need to explicitly request it to the user.

Metiore (Bueno et al, 2001) is a search engine that ranks documents according to user preferences, which are learned from user historical feedback depending on the user objective. Queries might be based on content and also other document attributes, such as title, author and year. The user must define an objective for each search session. User models consist of the documents that the user has classified in previous sessions. Relevance or interest feedback is explicitly required from the user, who can classify each returned document in one of the following categories: "ok", "known", "?", "wrong". By default all documents are classified as *normal*, standing for *not classified.*

Letizia (Lieberman, 1995) is a user interface agent that assists a user browsing the Web. Somehow similar to Metiore, Letizia also suggests potentially interesting links for the user to follow. Interest in a document is learned through several heuristics that explore user actions, user history and current context. In Letizia the notion of interest is global

to the user; it is not conditioned by the user objective, so there is no need to specify any session objective.

Personal View Agent, PVA, (Chen et al, 2001) is another personalization system that learns user profiles in order to assist them when they search information in the Web. This system organizes documents in a hierarchical structure – the *personal view*, which is user dependent and dynamic, automatically adapting to changes in user's interest. Relevance feedback is also obtained in implicitly, by analysing information from proxy server log files; in particular, documents whose visit time is larger than 2 minutes are considered positive examples. All specific personal views (hierarchical structures of categories that represent user interests) are derived from the *world view*, a generalist pre-defined taxonomy used by default as a starting point of user interests. Personal views are updated by merging and splitting nodes (two crucial operators of the system) in the hierarchical structure according to the perceived interest in each node.

Thesus (Halkidi et al, 2003) allows for the users to search documents in a previously fetched and classified document collection. In this system documents are classified based on document contents and on *link semantics*; authors claim that in-link semantics might improve document classification. The system includes four components: the document acquisition module, which starts from a set of seed URLs and crawls new documents following hyperlinks that carry specific semantics; the information extraction module, which extracts keywords from incoming hyperlinks from the documents in the collection and maps them to concepts in the predefined ontology; the clustering module, which partitions the document collection into coherent subsets based on keywords and also on the semantic tags associated to documents in the previous phase; and, finally, the query module, which allows for the user to explore the document collection.

The **ARC** system (Chakrabarti et al, 1999), a part of the Clever project, compiles a list of authoritative web resources on any topic. The algorithm has three phases. In the first phase, search and grow, a query is submitted to AltaVista and a root set is constituted with the top 200 returned documents. This root set is expanded by adding direct neighbors (both in-links and out-links); this expansion step is executed twice so the final expanded set contains a radius 2 neighborhood from the initial root set (in the HITS this

expanded set just contains the root set and their direct neighbors). In the second, weighting, phase the anchor text is extracted from the documents in the expanded set and links are weighted by the relevance of the terms in the anchor text vicinity. In the third, iteration and reposting, phase an iterative process is carried, just like in HITS, in order to compute authority and hub measures for each document in the expanded set. The documents with the fifteen highest scores of authority are returned to the user as the authority pages to the topic and fifteen highest scores of hub measure as the topic hub pages.

WebLearn (Liu et al, 2003) is a system that retrieves documents related to a topic, which is specified through a set of keywords, and then automatically identifies a set of *salient topics*, by analysing the most relevant documents retrieved in response to the user query that describes the topic. The identification of these salient topics is a fully automatic process that does not allow for user interference.

Grouper (Zamir et al, 1999), operated by University of Washington between 1997 and 2000, was a document clustering interface, which clustered document collections, at run time, as they were returned by HuskySearch meta-search engine (HuskySearch and Grouper were retired from service in 2000). By generating clusters with simple descriptions Grouper provided a way of organizing search results into collections for ease of browsing. *Tumba!* (Silva, 2002) is another example of a search engine (http:///www.tumba.pt) that might organize query output into clusters. *NorthernLight* (http://www.northernlight.com) is a commercial search engine, mainly oriented to business, that presents many innovative search capabilities, including the so called *custom search folders*, which is, again a way to cluster the output of a query.

In the next chapter we will describe our methodology, which shares some of these functional characteristics, although the mechanisms that are applied to guarantee them are distinct. Like PVA the resource organization is dynamically adapted to drifts in user information needs. PVA, Letizia and Personal WebWatcher explore implicit relevance feedback based on the actions performed by the user during search sessions. These user actions are mainly related to server requests for web pages and the time between requests, a requested web page is considered to be a positive example. PVA also organizes resources according to a user specific taxonomy. This taxonomy is always

derived from an initial static taxonomy, global to all users. PVA does not allow for the explicit definition of topics, each user has its own taxonomy independently of any specific topic. webTOPIC also exhibits a few characteristics that are not present at any of the previously discussed systems among which we may stress:

- the presentation phase that we consider fundamental to help the user take advantage of his or her resources;
- the definition of corrective and preventive procedures, to be automatically executed in order to keep system performance at acceptable levels, responding to present or foreknowable drifts in user information needs;
- topic organization defined by the user through examples.

4 webTOPIC: the proposed methodology

The aim of our work is the development of a methodology to assist filtering or, more specifically, routing tasks on the Web. Stated more precisely, our methodology should assist the user in the continuous process of maintaining an up-to-date document collection on some specific user-defined topic. The documents are retrieved from the Web and organized according to a concept hierarchy, also defined by the user, which reflects the desired ontological structure of the resource. The methodology should also detect drift in user interests, adjusting resources in order to keep them according to the user's current interests. This methodology should leverage the advantages of both search engines and topic directories: similarly to a search engine, it should return a set of ranked documents in response to a user query; like a topic directory, it should organize these document collections according to a given taxonomy. webTOPIC methodology is partially implemented in a prototype that we have used to conduct the experimental study. We will refer to this prototype and to a hypothetic implementation of the methodology as the *system*.

At this point it is convenient to define a few concepts:

- *User*: someone who has some specific information need, and wants to satisfy it through the webTOPIC methodology. A user is associated to at least one topic.
- *Topic*: specification of the user information need. It has four main properties: name, representative keywords, taxonomy and a set of exemplary pre-labeled documents. The user initializes the values of these properties. A topic is dynamic since its properties are periodically updated by the methodology, reflecting the evolution of user's interest regarding that specific topic. A topic is always associated to a specific user. The methodology deals with topic/user binomials rather than isolated topics or isolated users.
- *Resource*: document collection that satisfies a specific user information need or topic; each resource belongs to a given topic. A resource is an instance of a topic. Resources are also dynamic since they reflect current web content. The methodology may hold a retrospective view of the topic, which will be composed

by several resources, each one with reference to a specific period of time. The current view of a given topic is its most recent resource. We will refer to resource either as an instance of a given topic at a given instant of time or as the set of all the instances of the topic along time.

- *Document*: element of a resource. A resource is a set of documents. In the current implementation of the methodology all resource elements are HTML pages.

webTOPIC methodology requires the user to specify a topic. Once this is done, topic keywords or key-phrases are used to build queries that are submitted to public search engines. The answers to these queries, which are list of URLs, are presented to the user who has the responsibility of classifying some of them. These exemplary pre-labeled examples are then used to train classifiers for the topic taxonomy, using semi-supervised techniques.

Periodically these queries are submitted and new documents are automatically classified (applying previously learned classifiers) and added to the resource. This way the resource is kept up-to-date. New resources are instantiated periodically at fixed time intervals or whenever appropriate.

User interaction with the resource is facilitated by a graphical interface that also records user's actions in a log file. The information recorded in this log file is analysed in order to detect – or anticipate – drift in user's interests, triggering adequate corrective – or preventive – procedures. This way the methodology continuously adapts to changes in user's interests, without requiring explicit feedback, and tries to keep the resource quality above some acceptable pre-defined level. The methodology keeps records of resources' quality, measured on a specific quality space, which is automatically monitored and registered at run-time.

4.1 User's point of view

From the user's point of view, webTOPIC is a methodology that provides a set of services requested by him or her (Figure 4.1). The main services directly available to the user allow explicitly defining a new topic, classifying exemplary documents – which are part of the topic specification – and exploring resources. The user explores his or

hers resources during a session that may be aided by a visualization tool that simultaneously produces an actions' log file, where user actions are recorded. This user actions' log will be used to learn and adapt to evolutions on user interests.

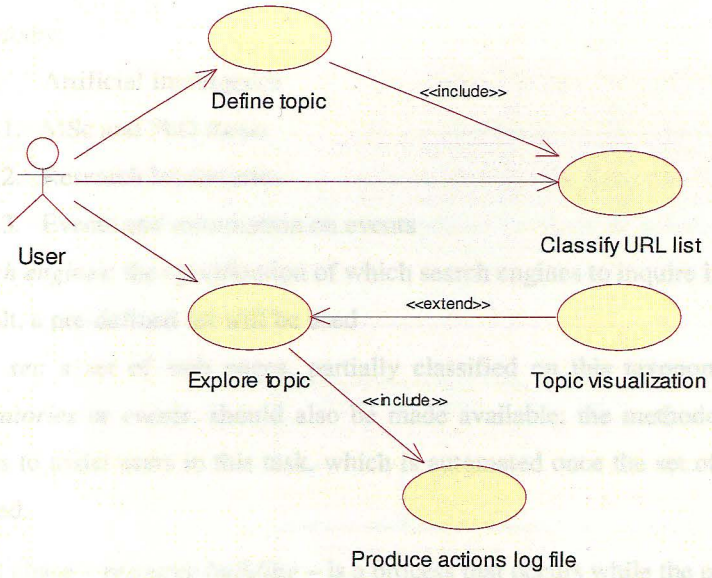


Figure 4.1 – webTOPIC from user's point of view

The user interacts with the methodology during two distinct phases: in the first phase the user defines the topic and specifies its characteristics, in the second phase the user explores the resources compiled by the system (Figure 4.2).

The first phase – named *topic specification* – occurs once per topic and is concentrated on a short period of time. The specification of a new topic includes:

- topic name;
- set of representative keywords or key-phrases;
- taxonomy, an hierarchy of concepts specifying the ontological structure of the resource;
- the set of search engines to inquire;
- a partially classified data set: a set of partially labeled documents, according to the

given taxonomy, allowing to characterize through examples each concept of the topic taxonomy;

As an example we present the following topic specification for *Artificial Intelligence*:

Topic name: Artificial Intelligence

Keywords: {"Artificial", "Intelligence"}

Taxonomy:

1. Artificial Intelligence
1.1. MSc and PhD thesis
1.2. Research laboratories
1.3. Events and information on events

Search engines: the specification of which search engines to inquire is optional; by default, a pre-defined set will be used

Data set: a set of web pages, partially classified on this taxonomy, as *thesis*, *laboratories* or *events*, should also be made available; the methodology has the means to assist users in this task, which is automated once the set of keywords is defined.

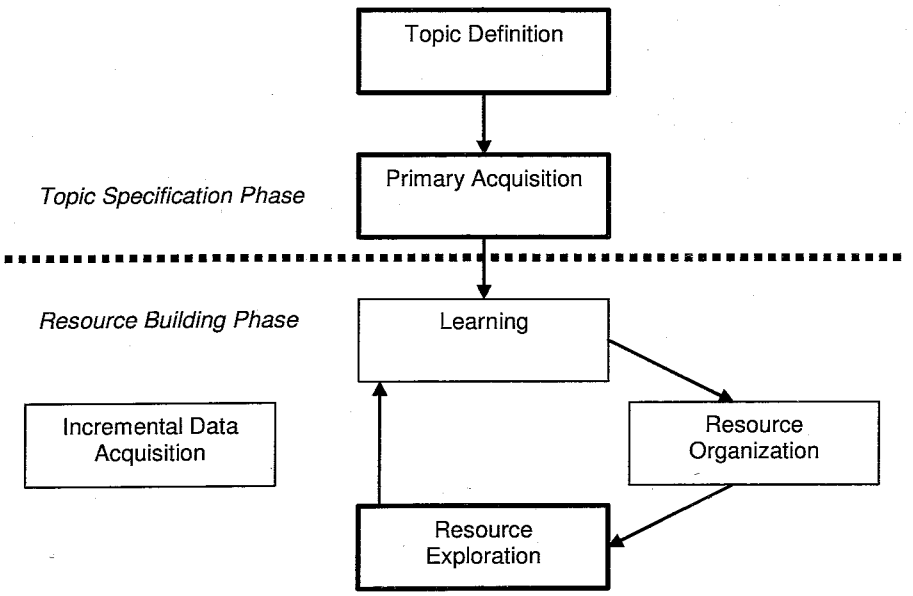
The second phase – *resource building* – is a process that occurs while the user maintains his or hers interest in the topic. Once the topic is fully defined, the system enters this second phase. In this phase the system learns the topic, from the examples that the user has classified while in the initial state, and learns user preferences, from user actions, while exploring the resource, automatically building the resource and keeping it up-to-date.

During this second phase the user interaction is recorded and the gathered data is used to improve system quality, detecting and adapting to changes in user's interests. A visualization framework helps the user exploring the topic; this may be helpful mainly when the resources are large document collections, which is common in web IR.

During the *Topic Definition* task the user specifies a new topic. Next, at the *Primary Acquisition* task, the system uses the topic keywords to build the query strings that are submitted to their respective search engines gathering a set of documents, which we will call the *primary data set*. This primary data set is partially classified by the user who labels a few exemplary documents according to the previously defined taxonomy.

When this task is finished the system enters the Resource Building phase and learns classifiers for each of the categories in the taxonomy, instantiating a model for the topic/user binomial. This model is used to organize the resource and it will be continuously updated, based on data that the *Learning* module receives from the *Resource Exploration* task.

The data used to keep models lined up with user's interests is gathered, in a way that is transparent to the user, by the *Resource Exploration* module. In parallel, at moments defined by the meta-search scheduler, which are dependent on the topic/user binomial, the *Incremental Data Acquisition* task retrieves documents from the Web trying to acquire new, previously unseen, documents, updated versions of previously acquired documents and identify broken links, in order to keep the resource up-to-date.



Legend: bold border boxes represent tasks that require user intervention; the others tasks are fully automatic

Figure 4.2 – Operational phases

4.2 Functional description

In this section we refer to the dynamic details of the webTOPIC methodology, defining how it operates.

4.2.1 Topic specification

Topic specification is accomplished through its designation, a set of keywords and taxonomy. This taxonomy represents the way the user wants to organize his or hers resource. The concepts in the taxonomy are characterized through examples.

Category specification

Once the user has defined the topic taxonomy, one possible approach would be to request him to specify a set of keywords for each one of these taxonomy categories, instead of exclusively specifying a set of keywords global to the topic, as prescribed in our methodology. Although this seems also reasonable, we believe it is more restrictive and laborious than our approach. It is more laborious because it requires the user to specify a set of keywords for each class in the taxonomy, instead of specifying just a single set corresponding to the topic itself. Besides, although it seems straightforward for the user to specify a representative set of keywords for the topic that he is interested in, it does not seem so easy to require the user to specify a representative set of keywords for each of the categories. The taxonomy is just an organizational artifact and, although the user knows the way he or she wants to organize the resource, he or she probably had never given much attention to the specific terms that are representative of each of the organizational concepts in the taxonomy. Several other reasons seem to support our approach:

- the user may not know the representative keywords for each concept;
- the concepts may not be easily characterized by a set of keywords;
- the topic contamination effect may negatively influence query results, specially when keywords are not well chosen; this problem may be efficiently solved by using positive examples instead of defining keywords (Chakrabarti, 1999).

Taxonomy

Topic taxonomy is a hierarchy of concepts, whose root is the topic itself. The specification of a taxonomy, which is merely a way of structuring the resource, seems appropriate because of the multi-faceted nature of web documents (Buntine et al, 2002); we believe that it is not possible to know which particular facet the user is interested in

unless he or she specifies it.

Several users, or even the same user at different moments in time, may be interested in organizing the same document collection in several different ways. It is not possible to know what the desired structure is just by looking at the topic or at the resource; the same resource may be organized according to many distinct structures. We believe this is a key question that makes automatic clustering techniques inappropriate for our methodology. The user has to explicitly declare which is the most adequate way of structuring the resource at the moment. Unsupervised automatic classification, or clustering, organizes documents according to latent prominent characteristics of the document collection and does not take into account user preferences. The user may be interested in the topic *computers* and the taxonomy *(hardware, software, book)* or the taxonomy *(Windows, Unix, AS400)* or any other that it is not possible to know which one is he or she interest in advance, unless the user states it.

4.2.2 Resource acquisition

An automatic resource compilation system should provide the user with an up-to-date document collection. We intuitively believe that in this type of search service the freshness of the resource is of greater importance than its recall, provided that the recall is above some critical minimum threshold. As the web size grows, and since it is not feasible to intend reaching it all, it is important to get to the best pages in first place (Castillo et al, 2004) and to guarantee that they are current. In this hypothesis, the resource construction is a continuous process of data acquisition from the Web designed with two goals: fetch new pages as soon as possible and keep previously fetched pages up-to-date.

Document retrieval

While operating in the Topic Specification phase there is a primary data set acquisition step during which the resource is initialized. During this resource initialization step specific tools assist the users in the partial classification of this primary collection. The primary collection of a topic is the consolidation of the result sets returned by the first meta-search cycle executed for the topic. The methodology does not conduct any crawling; it relies on the answers from the base search engines. We believe that this

process might be improved with crawling capabilities, for instance, by adding to the primary data set their out-linked neighbor documents.

Further on, at the Resource Building phase, the resource will be automatic and periodically updated. The Web presents high dynamics requiring special attention to keep a fresh view of its content. The process that has the responsibility to keep the freshness of the resource, accomplishes the following tasks:

- detect and retrieve new pages, that are not yet part of the resource;
- keep previously fetched pages updated, by retrieving them, analyzing their characteristics and decide whether they have changed since last retrieval cycle; if they have changed them the document version is incremented and its previous version is sent to history, otherwise the crawl cycle of the document is incremented (see Figure 4.17); this information is useful for future analysis on the frequency of change of the resource elements;
- purge broken links; web pages that are no longer available online must be excluded from the resource or, at least, marked as broken links.

Whenever a web page is fetched for the first time, its content changes or it is considered discontinued, the resource log is updated in order to keep a record of the resource evolution. The crawling scheduler uses this data, to improve the meta-search strategy, which defines the moments of execution of topic queries.

Meta-search

At the initial search cycle for a new topic the system submits a set of queries to the search engines that are either specified by the user or automatically selected by the system – by default all the available search engines will be selected for the first search cycle. The answers returned from these search engines are merged, applying the Borda fusion technique (Aslam et al, 2001), resulting in a list of URLs that constitute the primary data set.

Information on the submitted queries is recorded. In particular the numbers of positive, negative and invalid URLs are stored. This data will be later used to determine each search engine's ability for each query. The selection of which search engines are the most adequate for a specific query is based on two criteria computed from this data:

- the ratio between the number of positive documents returned by the total number of documents returned, which is the search engine local precision as to the present query;
- the number of positive documents returned, despite the result set size, this criterion gives more importance to recall.

Whether we use one or the other criterion to rank search engines, the subset to be selected may be defined by some absolute measure, either as the first *n* search engines in the ranking order or by defining edge thresholds for each criterion, which should be observed by all the search engines to be selected.

Once the set of search engines to inquire is selected the system must generate the query strings to be submitted to each one, based on the specific search engine query syntax. The query syntax specification is merely a character string with special tokens to be replaced by the specific query properties.

Merging the answers returned from each of the search engines is the final step in the meta-search process. After submitting the queries to their respective search engines, the answers are processed to extract URLs into a text file. In these text files, each line – consisting of the fields url, status, status date and class – represents a document. These text files, one from each search engine, are then merged into a unified list applying Borda fusing. This consolidated list of URLs, the result of a retrieval cycle for a specific topic, must be classified. If it comes from the primary acquisition step the classification is a manual and partial process; otherwise it is automatic.

Cross-classification

Human classification of web pages is highly subjective, inconsistent and erroneous (Kobayashi et al, 2000; Macskassy, 1998). Minimizing human classification errors at the primary data set is of crucial importance for the accuracy and global performance of the methodology. To accomplish this, webTOPIC methodology allows performing what we have called *cross-classification*: instead of relying on a single user, the primary data set may be simultaneously classified by a team of specialized users.

This list of URLs is submitted to a team of users who have the responsibility of classifying a few documents from the list on the topic taxonomy. Each of the users in

the team will receive a copy of the list to classify. The methodology does not need the full list to be classified, since it applies semi-supervised learning techniques. It just requires a few documents from each category in the taxonomy to be labeled. We will experimentally determine the relation between the labeling coverage of the primary data set, a measure of the effort required to the user, and the accuracy of the classifier.

Each of the users in this team is assisted in the manual classification task by the manual classifier tool (Figure 4.3). This tool reads the URL list file and the topic taxonomy – where it adds the special categories *unlabeled* and *negative* – and allows for the user to preview and assign categories from the topic taxonomy to documents from the URL list. Assigned categories are recorded in the URL list file. By default, all documents in the URL list are automatically assigned to the unlabeled category.

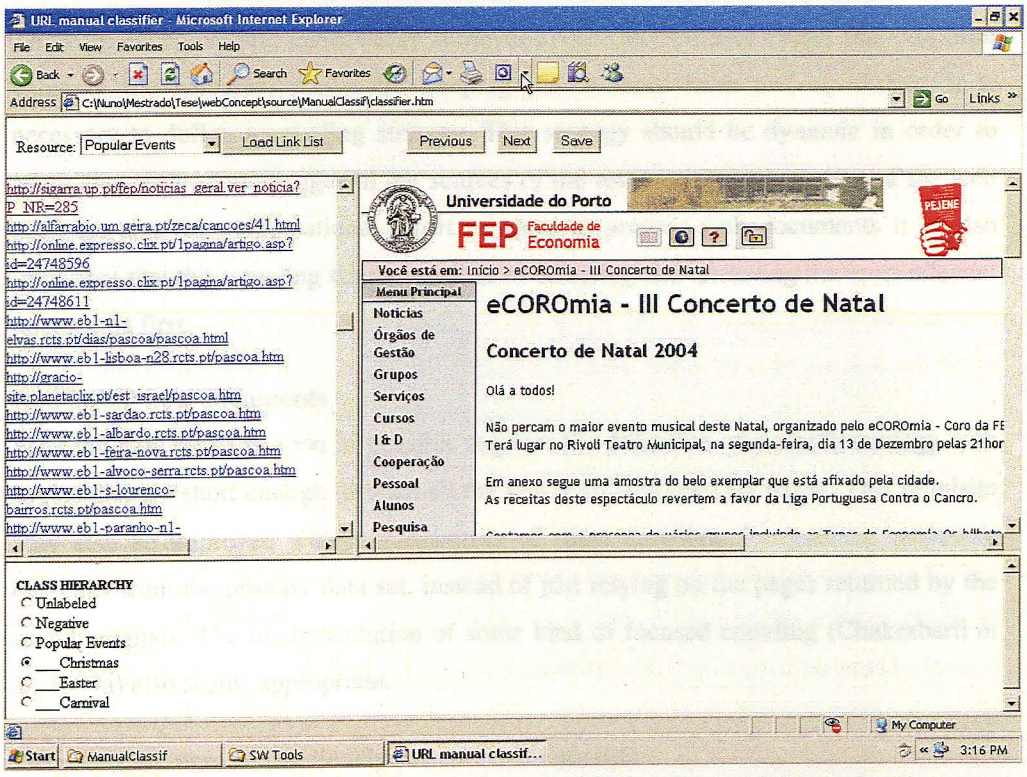


Figure 4.3 – Manual classifier tool

Once the labeled lists have been received from each of the users, or when the Topic

Specification phase reaches its deadline, the available lists have to be validated in order to detect ambiguities – an ambiguity occurs when different users have assigned more than one distinct class to the same URL. The process responsible for this step generates a list of validated URLs. In the current prototype no attempt is made to correct ambiguities.

Several simple correcting heuristics can be easily implemented. A voting mechanism might be used to eliminate ambiguities by choosing the category that has received more votes. The organizational structure of the resource might also be explored by associating an ambiguous document to the upper common category of the different user defined categories. Once this phase is concluded the system archives the set of pages in the validated URL list.

Refresh strategy

To avoid excessive and unnecessary retrieval cycles and retrieval cycles that add little value to the resource, while still keeping the resource fresh and up-to-date, it is necessary to define a crawling strategy. This strategy should be dynamic in order to adapt and respond to changes in the sources of the resource. Given the size of the web database and the computational effort required to process web documents it is also important that this crawling strategy is able of detecting and retrieving the most relevant documents first.

Retrieving new documents

Fetching new pages as soon as possible requires the definition of a search strategy with cycles that are short enough to fetch all the newly created relevant pages. This requisite may also be improved with the execution of some crawling, for instance retrieving out-links from the primary data set, instead of just relying on the pages returned by the search engines. The implementation of some kind of focused crawling (Chakrabarti et al, 1999a) also seems appropriate.

Detecting changes in previously retrieved documents

Updating previously fetched pages requires the definition of a revisiting strategy aimed at detecting changes in pages content – based on experimental evidence (Lim et al, 2001) conclude that more than 90% of web pages changes are different by less than

20% of words – as well as broken links – pages removed from the server since last fetch. Revisiting frequency for each document is computed, for each resource document, from the number of retrieval cycles without change. The estimation of the frequency of change may be facilitated by a regular access interval (Cho et al, 2000c). Thus, we start with a regular access interval for all the documents in resource and keep lowering this interval – increasing sampling frequency – until we catch the frequency of all changes.

Resource instantiation

The instantiation of a new version of the topic – a new resource – depends on the current topic update mode. Two distinct modes are defined:

- *Content*; in this mode a new resource is activated whenever the number of accumulated changes – retrieving of a new document, change in the content of a document, detection of broken link – that were identified at the resource since it's instantiation, reaches a pre-defined threshold. In the limit, if this threshold is set to 1, a new resource is instantiated whenever any change occurs in any document on the active resource. The initial value of this threshold is pre-defined but it may be dynamically updated by the system.
- *Time*; in this mode a new resource is activated periodically; the period is defined by the number of retrieval cycles it contains. In this mode a new resource is activated after every n retrieval cycles; the initial value of n is pre-defined but it may be dynamically updated by the system.

4.2.3 Pre-processing

Once archived, each document must be transformed into a model that is adequate for the automatic processing that will be further required. This pre-processing phase is responsible for transforming an HTML document into its representation in the selected modeling framework. It includes a *data preparation* step, which extracts the required set of features from HTML files, and a *data representation* step, which builds the document model from the set of features previously extracted from the document file.

Data preparation

The data preparation goal is to eliminate any linguistic or structural symbols, which are required by the syntactic rules of the content language and also of the programming language used to build the web page, but that do not add any semantic value to the document. Eliminating these symbols from the document reduces the feature set size, thus contributing to reduce the computational effort and avoiding confuse the learning stage with irrelevant features. The process described in Figure 4.4 executes this task.

The first activity – HTML parsing – parses HTML documents and build the corresponding document's DOM tree view – *DOM, Document Object Model* is a platform-neutral interface that allows programs to dynamically access and update the content, structure and style of documents (http://www.w3.org/DOM/Group/IG). This view is then manipulated in order to extract structural features – title and headings – in an attempt to explore the semi-structured nature of HTML.

Once this is done we may remove HTML tags, thus reducing the document exclusively to its content text. This step returns document text, free from HTML tags. This view of the document object is identified by document status "AF" corresponding to document full content text (see Figure 4.17).

This view is used for the detection of duplicated documents. The detection of duplicated documents is important for two reasons:

- we want to detect copies of documents, that may be present on the Web at mirror sites or as integrating parts of distinct sites, avoiding to include them on the resource, which should keep just unique, distinct, documents;
- we want to detect if a given document, already present in our resource, has changed since it was last retrieved, or if it keeps unchanged.

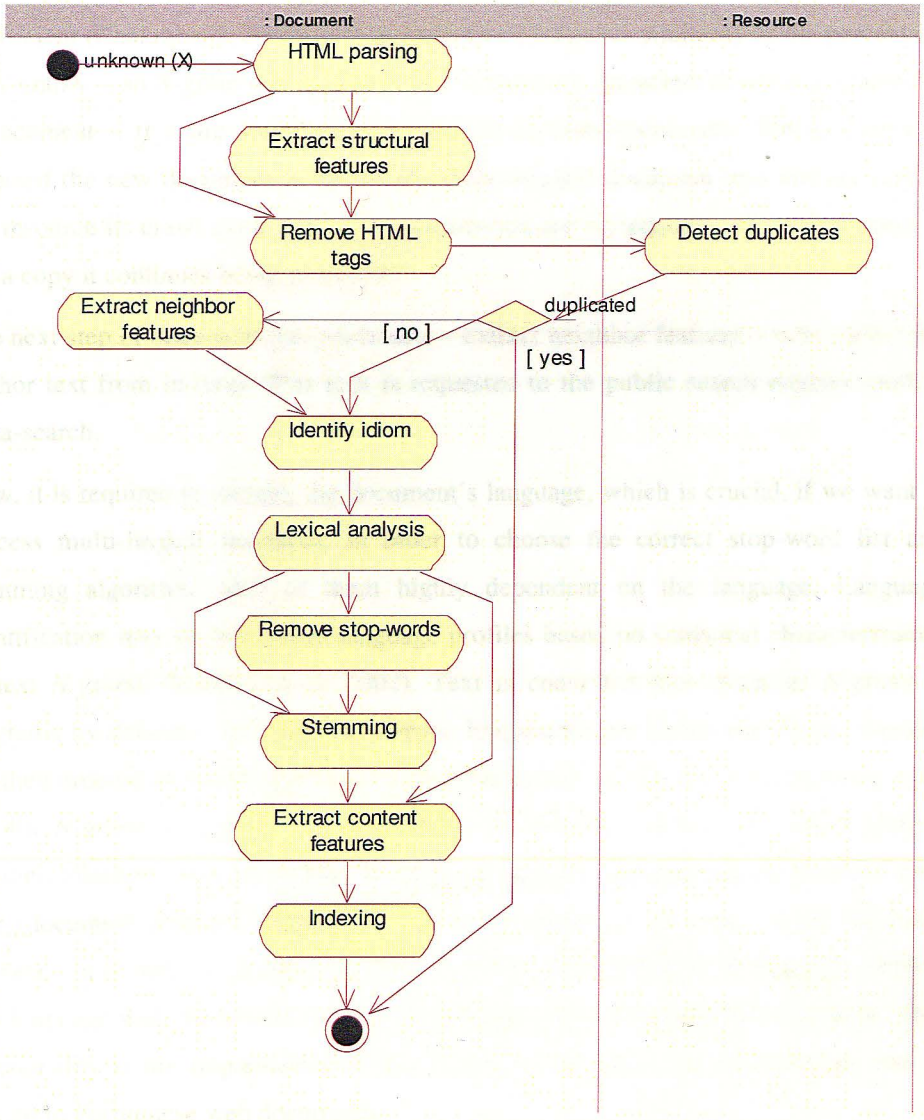


Figure 4.4 – Document pre-processing, data preparation step

Document similarity may be very restrictive, if we are looking for full match, or more permissive, if we are just interested in near match. For what concerns our purpose we believe that full match – the detection of exact copies – does not help much and may even be harmful since it distinguishes two copies of the same document that just differ by details that, on some circumstances, might be irrelevant, such as, for instance, a date.

To detect near copies we may use a document similarity measure based on Jaccard coefficient (Chakrabarti, 2003), which quantifies documents similarity as the percentage of *N*-grams – an *N*-gram is a sequence of *N* contiguous characters or terms contained in a document – that are simultaneously present at both documents. When a copy is detected the new document is discarded; if the original document was already part of the resource its crawl cycle and copies parameters are incremented. If the document is not a copy it continues being processed.

The next step of document pre-processing – extract neighbor features – is to collect the anchor text from in-links. This task is requested to the public search engines used in meta-search.

Now, it is required to identify the document's language, which is crucial, if we want to process multi-lingual resources, in order to choose the correct stop-word list and stemming algorithm, both of them highly dependent on the language. Language identification may be done from language profiles based on statistical characterization of text *N*-grams (Martins et al, 2002). Text is converted into character *N*-grams – tri-grams by default – and their occurrence frequencies are computed. These *N*-grams are then ordered by decreasing order of their frequency and the top most frequent – the top 400 *N*-grams by default – are recorded as the document profile. Document profiles are then matched to the characteristic language profiles and the distance between each pair (document profile, language profile) is measured as the sum of the distances between out of order *N*-grams in document profile when compared to language profiles. The language with the lowest value of this measure is associated to the document. This functionality is not implemented in the current prototype of the methodology that is limited to Portuguese web documents.

The full content view of a document is, usually, still a very noisy representation that can be improved by removing punctuation, extra space and special characters, converting all letters to their lowercase or uppercase form and eventually removing digits that represent numbers. This next step – lexical analysis – removes punctuation, extra space and special characters (such as linefeed and carriage return) and converts characters to lowercase, generating a new view of the document.

After this step we remove stop-words. Depending on the language used in the document

text the methodology would select the appropriate stop-word file, which contains a list of words to be removed from the text. These are words that, usually, do not carry any semantic value, such as articles. The execution of this task eliminates stop-words from the document and produces a new document view. This view is the set of words that constitute the document's content text. This view is then converted into another one consisting of the set of distinct terms present in the document and their respective frequencies.

This document representation, after stop-word removal and elimination of duplicated terms, may still be further reduced, by conflating words to their common semantic root; this is accomplished in the stemming step. This step is also dependent on the document's language. Porter's algorithm (Porter, 1980) is probably the most common stemming technique in use, however, although the rules may be adapted to other languages, it was proposed for the English language. We have decided to implement in our prototype a stemming algorithm, specifically developed for the Portuguese language (Orengo et al, 2001) and claimed to perform significantly better than the Portuguese version of the Porter algorithm (Chaves, 2003). The document is now transformed into the stemmed view, which contains the document stems and their frequencies that are computed by summing the frequencies of the terms that were conflated to the same common stem. In this view the terms are in their most simplified form. It is at this moment that the topic lexicon is updated: terms that are present at this document view but that do not yet exist in the topic's lexicon are included. Whenever the topic lexicon is updated the active resource lexicon is reinitialized as a copy of topic lexicon. Finally, the new document may be added to the resource.

Data representation

We propose a document model combining four document description levels – content, structure, metadata and neighborhood – that are potentially valuable sources of evidence concerning the classification of HTML documents. Each of these complementary aspects is characterized by a set of features:

- **Content** description level is characterized by text words (terms, stems).
- **Structure** description level is based on HTML tags including the features title and

headings.

- **Metadata** includes first fetch date, current version date, URL, status, status date, number of retrieval cycles without change, number of detected copies, automatic and user defined labels and respective dates and content language.
- **Neighborhood** description level tries to explore hyperlink web structure and is constituted by anchor text from in-links.

Document content is represented in the vector space model. For each document a TF×IDF vector is generated from the term frequencies of the document (TF factor) and each term's inverse document frequency (IDF factor). The TF×IDF model seems adequate because it is simple and seems to outperform other models with general collections (Baeza-Yates et al, 1999). Besides, it relies on the simplest form of document, the bag-of-words, being extremely flexible concerning the large spectrum of documents that is supports.

The TF×IDF representation of a document is a vector where each element represents the weight of the corresponding term in the document. Each document TF×IDF vector is computed in three steps:

1. Compute the maximum term frequency in the document, $max_j(TF_{ij})$; this is required because documents do not have all the same size and this allows the normalization of the TF factor, eliminating the effect of document size.
2. For each term, compute its IDF factor as $\log\left(\frac{N}{DF_j}\right)$, where N is the number of documents in the resource and DF_j is the number of documents containing term t_j.
3. For each term, compute the weight w_{ij} as $w_{ij} = \frac{TF_{ij}}{\max_i(TF_{ij})} \times \log\left(\frac{N}{DF_j}\right)$, the product of the normalized TF factor by the IDF factor

Each of these vectors is inserted into a document-by-term matrix, where each row represents one document. Once the resource is fully computed this matrix contains one row for each of the document that constitutes the resource and one column for each term in the resource lexicon. The row number is an index for the document collection,

identifying the document, and the column number is an index for resource lexicon, identifying the term. Each matrix element w_{ij} is the weight of term t_j in document d_i. Each document is now represented by the set of unique stems that occur in the document and their absolute frequencies and also by its final TF×IDF model.

We are now in conditions of selecting relevant text features. Text terms are not all equally important, concerning their discriminative power, and selecting the most relevant ones may significantly reduce the feature space by eliminating those terms that do not positively contribute to the classification task. We apply a method for the selection of text features that is based on document frequency techniques, which are simple, do not require many computational resources and have good performance (Yang et al, 1997).

By default we select as text features all the terms in the resource lexicon having an absolute frequency in the resource greater than 1. The final weights matrix considers only the columns that correspond to those terms; other columns are ignored – although all the weights w_{ij} are always computed allowing the application of any other feature selection techniques.

Finally our document model is complete:

- text features – content words – originated the vector space representations,
- structural features – HTML title and headings,
- metadata features – fetch and version date, URL, status, status date, retrieval cycle, number of copies, labels and respective dates and language, and
- neighborhood features – anchor text from in-links.

4.2.4 Learning

Our document model includes four distinct sets of attributes: attributes from document content, attributes derived from external document properties, attributes from HTML structure and attributes from the neighboring pages. The prototype we have implemented explores exclusively the text features descriptive level; the document model is thus reduced to its TF×IDF vector.

As previously discussed, learning the topic taxonomy in an unsupervised manner, by

applying clustering techniques, does not seem appropriate. The user may be interested in an organizational structure substantially different from the one obtained with unsupervised techniques. On the other hand, a supervised learning scheme requires a large number of labeled examples from each category. This is a major drawback since the manual classification of web pages is very demanding.

We propose a semi-supervised solution, requiring the user to classify a few examples from the primary data set at an initial phase. The system will then learn a classifier for each category in the taxonomy, based on the exemplary pre-labeled documents, using semi-supervised techniques. It is only required that the set of pre-labeled examples covers all the taxonomy categories.

webTOPIC uses a SVM classifier – SVM are currently the most accurate classifiers for text (Chakrabarti, 2003) – wrapped in a simple semi-supervised algorithm called *bootstrapping* (Jones et al, 1999). In this method, the classifier is wrapped in a process that iteratively labels unlabeled documents and adds them to the labeled set. This cycle is executed until a certain stopping criterion is met. One of these stopping criteria is based on the concept of classification gradient, which we have defined as a way of measuring model improvement between iterations.

Classification gradient is defined as the percentage of labels that change from one iteration to the next. It is computed as the number of documents that have been given, in the current iteration, a different label from the one that has been assigned at the last iteration, divided by the total number of training documents.

This is used as one of the stopping criteria for the semi-supervised classifier since, at this setting, it is not possible to define stopping criteria based on the evolution of error rates once the labels in the majority of the training documents are unknown. We assume that if a classifier, generated at a given iteration during the semi-supervised process, does not produce significantly different labels with respect to the previous iteration, the discriminative power of the data set has already been captured by the classifier. This stopping criterion may trap the classifier at local stationary points.

Our methodology implements four stopping criteria, controlled by four parameters: *max.iterations*, *min.unlabeled*, *min.gradient* and *min.posterior*:

1. number of iterations is greater than or equal to *max.iterations*, which is unlimited, by default;
2. the number of unlabeled documents in the training data set is less than *min.unlabeled*, this parameter defaults to 0;
3. the classification gradient is below the minimum threshold defined by *min.gradient* (by default one document);
4. none of the predictions has a posterior probability greater than *min.posterior*, by default *min.posterior* is computed as the inverse of the number of classes in the topic taxonomy.

The bootstrapping cycle stops on the first iteration that meets any one of these stopping criteria. The last generated SVM model will prevail and will be used to classify future documents to be included in the resource. This model is then used to classify the documents at the unlabeled data set. These SVM classifiers produce for each document a vector of probabilities containing the posterior probability of each category, given the document.

For each unlabeled document, the algorithm compares the posterior probability of the most probable category with a pre-defined minimum (*min.posterior*). If the category probability is higher than this pre-defined minimum the predicted label is accepted and the document is marked as *newly-labeled*, meaning that it has been labeled at the current iteration; otherwise we assume that there is not enough evidence to accept the predicted label for that document. Once all the unlabeled examples have been processed the documents marked as *newly-labeled* are added to the labeled data set and removed from the unlabeled data set.

Figure 4.5 represents the bootstrapping algorithm implemented at the current version of the webTOPIC methodology. After an initialization step, where the labeled and unlabeled data sets and the required control variables are instantiated, the algorithm enters an iterative process where it will remain until any of the stopping criteria is met.

At the beginning of each iteration the set of stopping criteria is verified; if none of these stopping criteria is met the iteration is executed, otherwise the algorithm returns. At each iteration a SVM classifier is learnt for each category in the topic taxonomy. The

classifier is trained with the set of documents that constitute the labeled data set

Finally the classification gradient for the current iteration is computed and control variables are updated. The current iteration ends and the stopping criteria are verified to decide whether a new iteration should be performed or not. This algorithm returns the SVM classifier that has been generated at the last iteration.

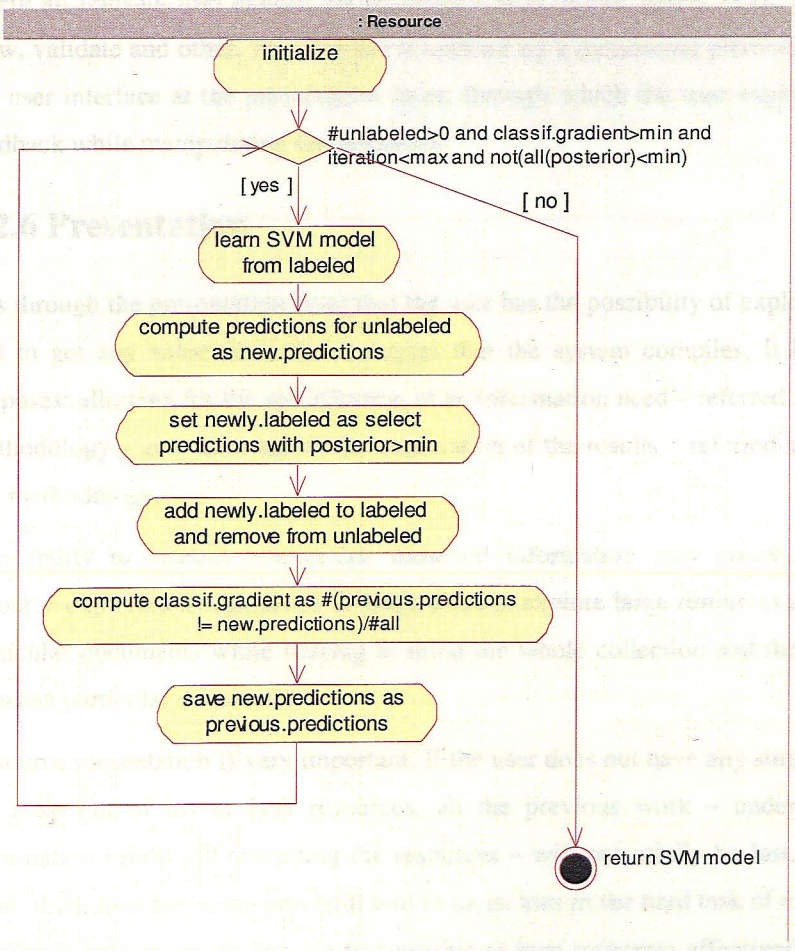


Figure 4.5 – Bootstrapping SVM

4.2.5 Analysis

Our methodology defines an automatic process to analyze the performance of the

system while in operation. This analysis is based on user's feedback, which is automatically gathered directly from the user's actions. This way the user provides valuable information to the system, about classification accuracy, without being explicitly required to do so.

This process that permanently analyzes both system classification accuracy and user information needs, which might naturally change over time, is fed through a log file where all relevant user actions are recorded – save, move, delete, print, open, browse, view, validate and other. This log file is updated by a monitoring process, embedded in the user interface at the presentation layer, through which the user implicitly provides feedback while manipulating the resources.

4.2.6 Presentation

It is through the presentation layer that the user has the possibility of exploring the topic and to get any value from the resources that the system compiles. It has two main purposes: allowing for the specification of an information need – referred as topic in our methodology – and allowing for the exploration of the results – referred as resources in our methodology.

The ability to visually manipulate retrieved information may greatly improve the resource exploration experience. It helps users to explore large resources and to analyze particular documents while bearing in mind the whole collection and the relationships between particular documents.

Resource presentation is very important. If the user does not have any simple way to get the most out of his or hers resources, all the previous work – understanding user information needs and compiling the resources – will potentially be lost. On the other hand, if the user has some powerful tool to assist him in the hard task of exploring large document collections, he may in fact use his or hers resources effectively, without the feeling of being disoriented or of missing important documents in the collection. This layer should work like an informed cicerone, like a librarian, able and available to guide users in the exploration of their resources.

This presentation layer provides two distinct views of the topic: the *organizational view* and the *exploratory view*.

Organizational view

The organizational view is a simple interface between users and resources directed for resource manipulation. Experienced users, familiarized with the resource, should use this interface; it is mainly oriented for users who know the topic and have been exploring it for some time and are aware of its internal structure. In this view the resource is seen as a directory tree where each node represents a concept of the topic taxonomy; the root directory stands for the topic itself. Each directory contains the set of documents labeled with the corresponding category (Figure 4.6).

Users can use this interface to manipulate the resource, for instance, by moving documents from one directory to another if they want to redefine the document classification. Changes in the resource – newly retrieved documents, documents that have changed and broken links – are identified by distinct colors: a newly added document may have its name printed in green until the user opens it for the first time, documents that have changed since last access will be printed in yellow and documents that have become broken links may be colored red.

Document icons show the labeling mode – *M* for manual or *A*, followed by the posterior probability of the label, when the label has been automatically assigned. Other properties – essentially metadata attributes like document's URL, classification score vector, labeling data – are directly available through the document icon. This interface implements a set of functionalities, which allow the user to reorganize the topic according to his or hers current interests:

- Topic taxonomy may be updated or re-defined by creating new concepts at any node in the taxonomy, moving nodes from one location to another, removing nodes, as if the user is manipulating directories in his or hers hard disk. By managing topic taxonomies through this interface the user is also providing data that will be used by the learning module to re-adjust document classification and resource organization. Whenever a new node is created in the topic taxonomy, and new documents are added to it, the system refreshes the topic taxonomy and rebuilds the document classifiers, based on the new taxonomy and using the associated documents.
- Moving a document from one node to another is assumed by the system as explicit

user re-labeling, corresponding to the manual classification of that document. When this happens, the system increments one counter that will be used to trigger the learning task. Classifiers for the topic taxonomy are re-trained when this counter reaches a pre-defined threshold value. The evaluation module dynamically updates this threshold.

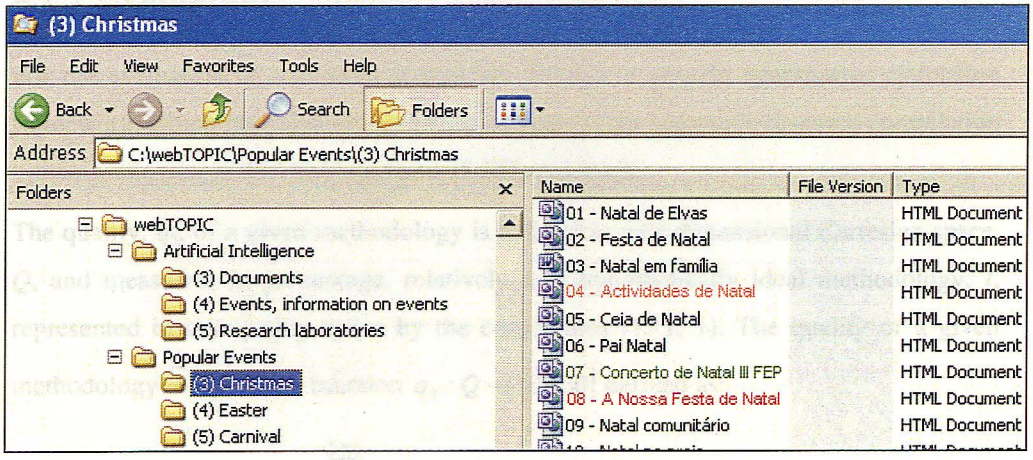


Figure 4.6 – Resource presentation, organizational view

Exploratory view

The exploratory view is meant for users who want to understand the internal structure of the resource, its cohesion, and how are topic categories covered and related. It will probably be preferred for the first sessions, when the user is starting to get acquainted with the resource, or after a long period of user inactivity during which the resource is supposed to have suffered extensive changes. This view is inspired by the Vibe system (Olsen et al, 1993) and implements a set of functionalities aimed at assisting users with large document collection's details and structure:

- characterization of points of interest based on several properties from documents, such as: label, text features, structural features, metadata and neighbor features or combinations of them;
- size, color and shape of the icons representing documents may be associated to several features;

- allow for the definition of the relative importance of points of interest;
- eliminate all the selected – or all but the selected – documents from the display;
- view all the documents that share some specific feature, or set of features, with any given document or set of documents.

4.2.7 Evaluation

For the evaluation of our methodology we propose a generic quantitative evaluation framework, which may also be applied to evaluate other automatic resource compilation systems, allowing a direct comparison of different tools.

The quality, q_S, of a given methodology is defined in a tri-dimensional Cartesian space, Q, and measured, in percentage, relatively to a hypothetically ideal methodology, I, represented in our quality space by the coordinates (1, 1, 1). The quality of a given methodology $\vec{S}$, q_S, is the function $q_S : Q \rightarrow [0,100]$ defined as:

$$q_S = \frac{\|\vec{S}\|}{\|\vec{I}\|} \times 100\% \qquad [4.1]$$

where $\|\vec{I}\|$ is the norm of the vector $\vec{I}$ = (1, 1, 1), representative of the ideal methodology.

A small simulation study, developed to evaluate the adequacy of this measure, showed that equation [4.1] gives relevance to high values, close to 1, in one of the coordinates, independently of the value of the other two – for instance, the system (1,1,0) has a quality of 82%, when we would ideally expect 2/3, and the system (1,0,0) has a quality of 58%, when we would expect 1/3, provided all the dimensions are equally important, as we assume, they should all equally contribute to the system quality measure.

Thus, it seems more adequate to define the quality of the system $\vec{S} = (s_1, s_2, s_3)$, q_S, as:

$$q_S = \frac{\log(1+s_1) + \log(1+s_2) + \log(1+s_3)}{3 * \log(2)} \times 100 \qquad [4.2]$$

Applying logarithms to the coordinates, before summing them, as in equation [4.2] may

reduce the impact of higher values in a given dimension – we sum the quantity 1 to each coordinate before applying the logarithm to avoid indeterminations of the type 0/0 – and presents quality values that seem more adequate than equation [4.1] – the system (1,1,0) has a quality of 67% and the system (1,0,0) has quality of 33%.

Quality dimensions

The quality space, *Q*, aggregates, in three dimensions – *Automation Level*, *Efficacy* and *Efficiency* – a set of indicators that measure the relevant characteristics of an automatic resource compilation methodology.

The *Automation Level* dimension reflects the automation level of the system. The automation level is understood as a complementary aspect of workload; it is measured indirectly, as the complement of the workload required to the user. The user workload is directly measured during system operation. We do not directly use the measured workload because we intend that all dimensions represent positive concepts – concepts to maximize – and the workload is to be minimized.

The *Efficacy* dimension is the aggregation of two measures common in IR: precision and accuracy (see section 3.3.3). Through them we can measure to what extent the resource is efficacious – whether it is focused and able to present accurate views of the topic.

The *Efficiency* dimension aggregates recall, freshness and novelty (see section 3.3.3). Through this dimension we measure the system's ability for presenting accurate views on the topic with minimum effort – i.e., with high recall – that are also fresh and frequently contain new documents, unknown to the user – high novelty.

The coordinates of vector $\vec{S}$ may be obtained through the application of one of several aggregation forms. We will compute these coordinates as the average of the indicators that contribute to it; the average is simple and gives the same relevance to all indicators. Quality dimensions are based on the following indicators:

Automation Level (*A*)

- User Workload (*W*)

Efficacy (*E*)

- Precision (P), Precision is computed over the full document collection that constitutes the resource, and not, as is usual, over a crawling cycle
- Accuracy (X)

Efficiency (C)

- Recall (R), recall is computed over the full document collection that constitutes the resource, and not, as is usual, over a crawling cycle.
- Freshness (F)
- Novelty (N)

Computing system coordinates in Q

We define the following terminology:

M is the number of documents manually classified by the user,

D is the total number of documents returned by a meta-search cycle,

C is the number of documents correctly classified in the respective categories of the topic taxonomy,

N is the total number of documents in the resource, equals the number of documents that are associated to the topic (the hierarchy root),

U is the estimate of the total number of documents associated to the topic that are present on the Web,

Y is the total number of documents that are up-to-date (see Appendix I).

The coordinates (A, E, C) of a given system S in space Q, are computed from:

$$A = 1 - W = 1 - \frac{M}{D} \qquad [4.3]$$

$$E = \alpha P + (1-\alpha)X = \alpha\frac{N}{D} + (1-\alpha)\frac{C}{N} \qquad [4.4]$$

$$C = \beta R + \gamma F + (1-\beta-\gamma)V = \beta\frac{N}{U} + \gamma\frac{Y}{N} + (1-\beta-\gamma)V \qquad [4.5]$$

and, if we consider all measures to be equally relevant to their respective dimensions,

then parameters α, β e γ will assume, respectively, the values 1/2, 1/3 and 1/3 and equations [4.4] and [4.5] will be simplified to:

$$E=\frac{1}{2}\frac{N}{D}+\frac{1}{2}\frac{C}{N}, \text{ if } \alpha = 1/2 \qquad [4.6]$$

$$C=\frac{1}{3}\frac{N}{U}+\frac{1}{3}\frac{Y}{N}+\frac{1}{3}V, \text{ if } \beta = \gamma = 1/3 \qquad [4.7]$$

Novelty is computed from equation [3.19]. We admit that any new document, which has been retrieved for the first time, is unknown to the user.

Preventive and corrective behaviors

All these measures, as well as the system quality, may be obtained and recorded by the system itself, while in operation, which can thus continuously evaluate its position in the quality space, its own quality, and try to correct any eventual degradation on a specific dimension or at a global level. Measured values are stored to analyze the evolution of the system quality and specific corrective procedures are pre-defined to be automatically executed when the system quality falls off some established limit thresholds. The application of forecasting techniques allows the execution of preventive measures, which are also defined. The methodology computes and records the evolution of the system quality and executes the preventive and corrective procedures, whenever necessary.

4.3 Structural description

In this section we describe the structure that supports our methodology. We introduce the methodology architecture, which describes the relationships and the interfaces between the main tasks that are required, and give an overview of the main entities that implement our system.

4.3.1 Architecture

webTOPIC methodology is composed of four main blocks, or sub-systems: *Acquisition*, *Organization* and *Presentation*, which are responsible for managing and processing resources, and *Evaluation*, which is responsible for evaluating system performance and

to trigger corrective and preventive actions that are eventually necessary to keep service quality at a reasonable level.

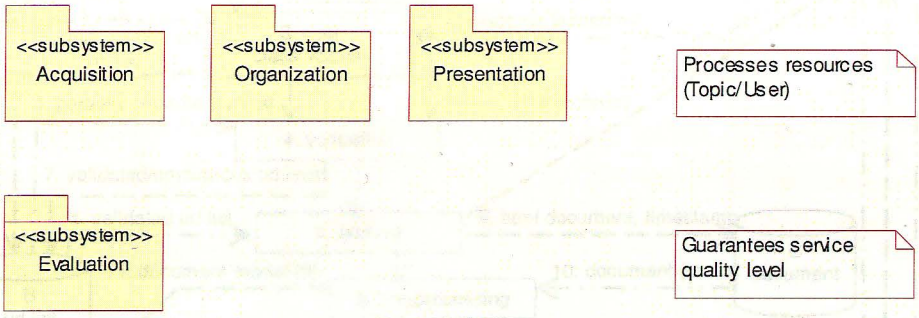


Figure 4.7 – webTOPIC modular organization

Figure 4.8 represents an internal view of the methodology specifying the tasks that are performed, their interfaces and the required flows of control and data.

The user defines the topic, at the task *1.topic definition*, specifying the set of keywords or key-phrases that are representative of the topic, the concept taxonomy associated to the topic and, optionally, a set of search engines to inquire. Using this information, the system, in task *2.acquisition*, constructs the required query strings, submits them to the corresponding search engines and merges the answers into a list of URLs, called the *primary data set*. This list of potentially interesting documents is partially classified by the user, in *3.user classification*, who may associate to each document a category of the topic taxonomy. The manual classification process that may be carried out by one or several distinct persons (see Cross-classification) has to be validated in order to detect eventual ambiguities, minimizing, this way, human classification errors. This validation is performed automatically by *4.validate*. The validation process generates a list of validated URLs and a list of ambiguous URLs; the validated URL list is retrieved and locally archived during *5.archive*. After this step this topic/user binomial leaves the Topic Specification Phase, entering the Resource Building Phase (Figure 4.2). The following step, *6.pre-processing*, converts HTML documents, previously archived, to the adequate representation model for the learning task.

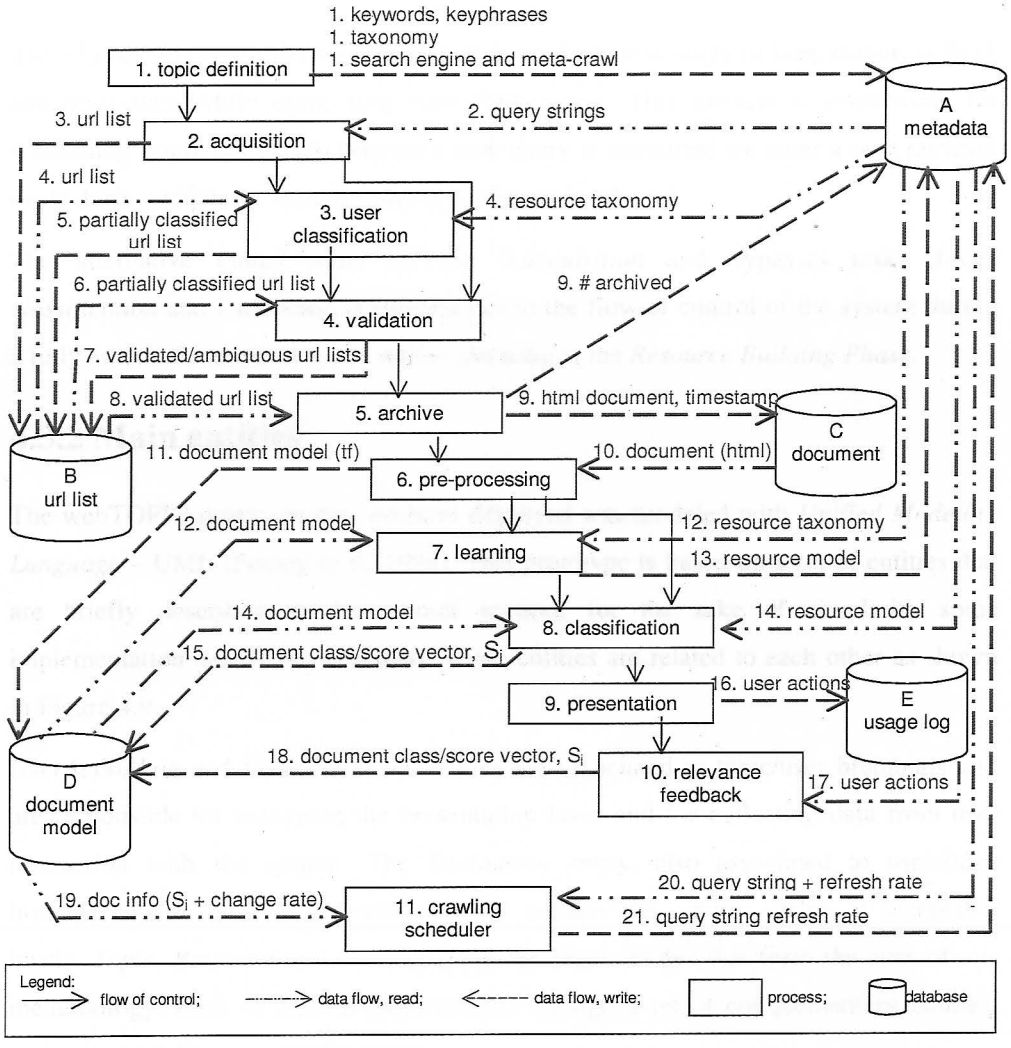


Figure 4.8 – webTOPIC architecture, internal view

In *7.learning* the system builds a classifier for each category of the topic taxonomy and in *8.classification* applies these classifiers to label the resource documents. Once this first resource instance is compiled the user can explore it, through *9.presentation*. It is expected that the user actions will provide evidence to improve the learning task and help to keep aligned with user interests. For that purpose, the task *10.relevance feedback* maintains a resource log where it registers all user actions – recording for each one, the topic/user binomial identification, the action that has been performed and specific action parameters including the target document and the time of occurrence.

The *11.crawling scheduler* module tries to infer the best strategy to keep resources fresh and up-to-date while optimizing bandwidth usage. This module is responsible for scheduling future queries. Whenever a new query is submitted we enter a new retrieval cycle during which the system follows an alternative thread.

The alternative thread starts at task *2.acquisition* and bypasses tasks *3.user classification* and *7.learning*; it corresponds to the flow of control of the system during a fully automatic retrieval cycle, while operating in the *Resource Building Phase*.

4.3.2 Main entities

The webTOPIC prototype that we have deployed was modeled with *Unified Modeling Language* – UML (Pooley et al, 1998). This prototype is based on a set of entities that are briefly described at the current section; for the sake of simplicity some implementation details were omitted. These entities are related to each other as shown in Figure 4.9.

UserActionLog and *VisualScenario* entities are associated to topic/user binomials and are responsible for managing the presentation layer and for collecting data from user interaction with the system. The *Evaluation* entity, also associated to topic/user binomials, is responsible for analyzing and keeping the service quality at acceptable levels. *Topic*, *Resource* and *Document* are the main entities that form the core of our methodology. Each of them is implemented through a set of complementary entities, each one modeling a particular aspect.

Topic entity

The Topic entity has the purpose of storing and processing user defined topics. A given topic relates to a user – the topic/user binomial – has its own taxonomy and lexicon, which is updated whenever new documents are added to the topic, and a set of resources. Each resource is the topic's image regarding a certain period of time. Topics are implemented at our prototype through the structure presented in Figure 4.10.

The topic is the starting point for our automatic resource compiler. Users specify topics that will subsequently be instantiated as resources. Once a new topic is defined a set of queries is prepared and submitted. Answers returned from these queries are merged into

a resource. As to its internal representation, a topic consists of the attributes and methods shown in Figure 4.11.

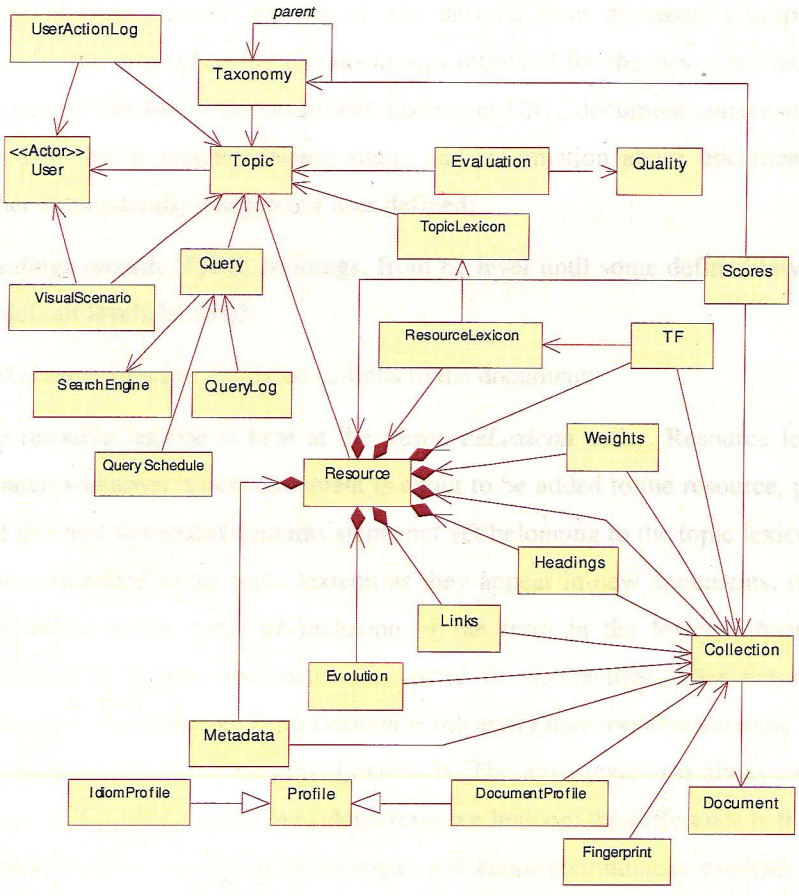


Figure 4.9 – webTOPIC entities

Resource entity

The purpose of webTOPIC is to collect and manage up-to-date, comprehensive resources. A resource is a document collection that must be mapped onto some informational structure that will allow for persistency and automatic processing and management of all document model features (see Data representation). webTOPIC prototype represents documents and resources through a set of nine entities, each of them modeling a complementary aspect of the resource and its documents (Figure 4.12):

- *Collection* entity identifies the documents that constitute the resource;
- *Metadata* entity contains relevant information about each document that was obtained from external sources or was derived from document's properties. It includes the date when the document was retrieved for the first time, the date of the current version of the document, document URL, document current status and the date that it entered current status and information about document labels, either automatically assigned or user defined;
- *Headings* records HTML headings, from h1 level until some defined lower level, by default levels h1 to h3;
- *Links* contain anchor text from in-links to the document;
- The resource lexicon is kept at the *ResourceLexicon* entity. Resource lexicon is updated whenever a new document is about to be added to the resource, provided that this new document contains stems not yet belonging to the topic lexicon. New terms are added to the topic lexicon as they appear in new documents; this way, term index is the order of inclusion of the term in the lexicon. New terms, originated in a new document, are sorted in alphabetical order before being included in the resource. *TopicLexicon* is the entity that contains the topic lexicon; it is updated whenever ResourceLexicon is. The topic lexicon is always a current image of the most recent, up-to-date, resource lexicon; the difference is that while TopicLexicon is associated to the topic, and keeps continuously evolving as long as the topic exists, ResourceLexicon is associated to a given resource, which may be substituted by another version. When this happens the previous resource version is frozen along with its lexicon;
- *Evolution* entity registers any state transition that affects documents from the resource;
- *TF* is a document-by-term matrix containing the absolute term frequencies occurring in each document. Row and column indexes correspond to the Collection and ResourceLexicon indexes of the respective document and term;
- *Weights* is another document-by-term matrix that contains TF×IDF weights. Each row of this matrix is the vector space representation of the corresponding

document. Row and column indexes correspond to the Collection and ResourceLexicon indexes of the respective document and term;

- *Scores* is a document by category matrix containing the posteriori probabilities for each category in the topic taxonomy associated to each document. Row index corresponds to the Collection index of the corresponding document.

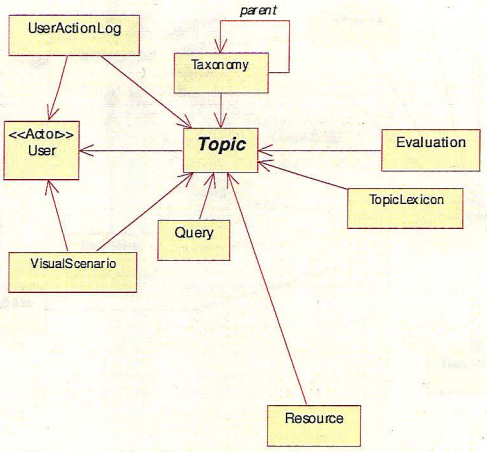


Figure 4.10 – Topic entity relationships

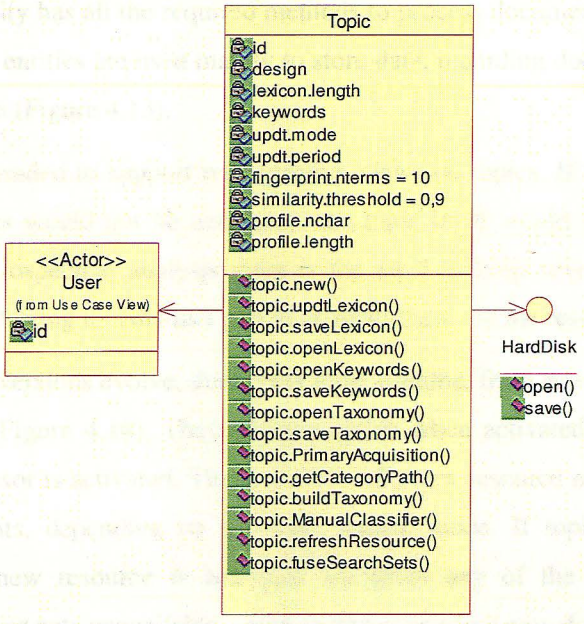


Figure 4.11 – Topic entity, attributes and methods

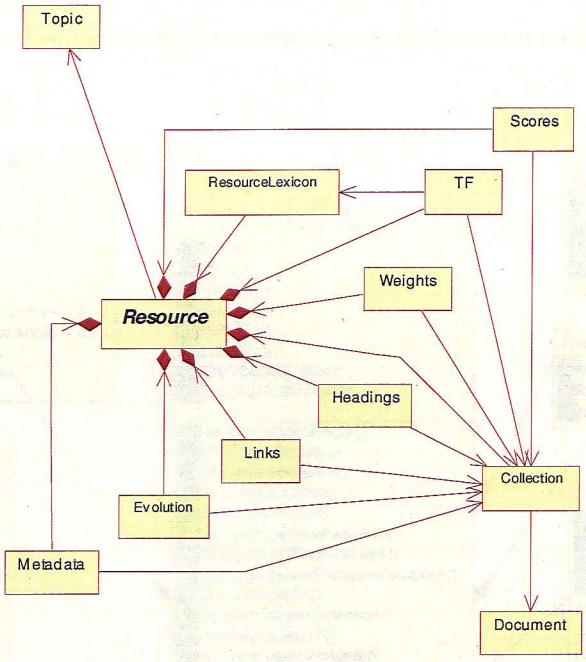


Figure 4.12 – Resource entity relationships

The Resource entity has all the required methods to process document collections while the other satellite entities are used mainly to store data, regarding document models and their classification (Figure 4.13).

Resources are intended to support retrospective views on topics. If it were not for this purpose, resources would not be necessary; the topic itself would suffice. In order to allow for this retrospective analysis there is the need to keep several pictures of the topic, each one referring to a distinct period of time: these are the resource versions.

Distinct resource versions evolve, during the topic lifetime, from *new* to *active* and from there to *history* (Figure 4.14). They become active when activated and go to history when their successor is activated. The activation of a new resource may be triggered by two distinct events, depending on the topic update mode. If topic update mode is *content*, then a new resource is activated whenever any of the documents in the collection change or gets unavailable – broken link – or when new documents are added to the resource. This may happen in any retrieval cycle. If topic update mode is *time*,

then a new resource is activated periodically; the period is defined in number of retrieval cycles. In this mode a new resource is activated every *n* retrieval cycles.

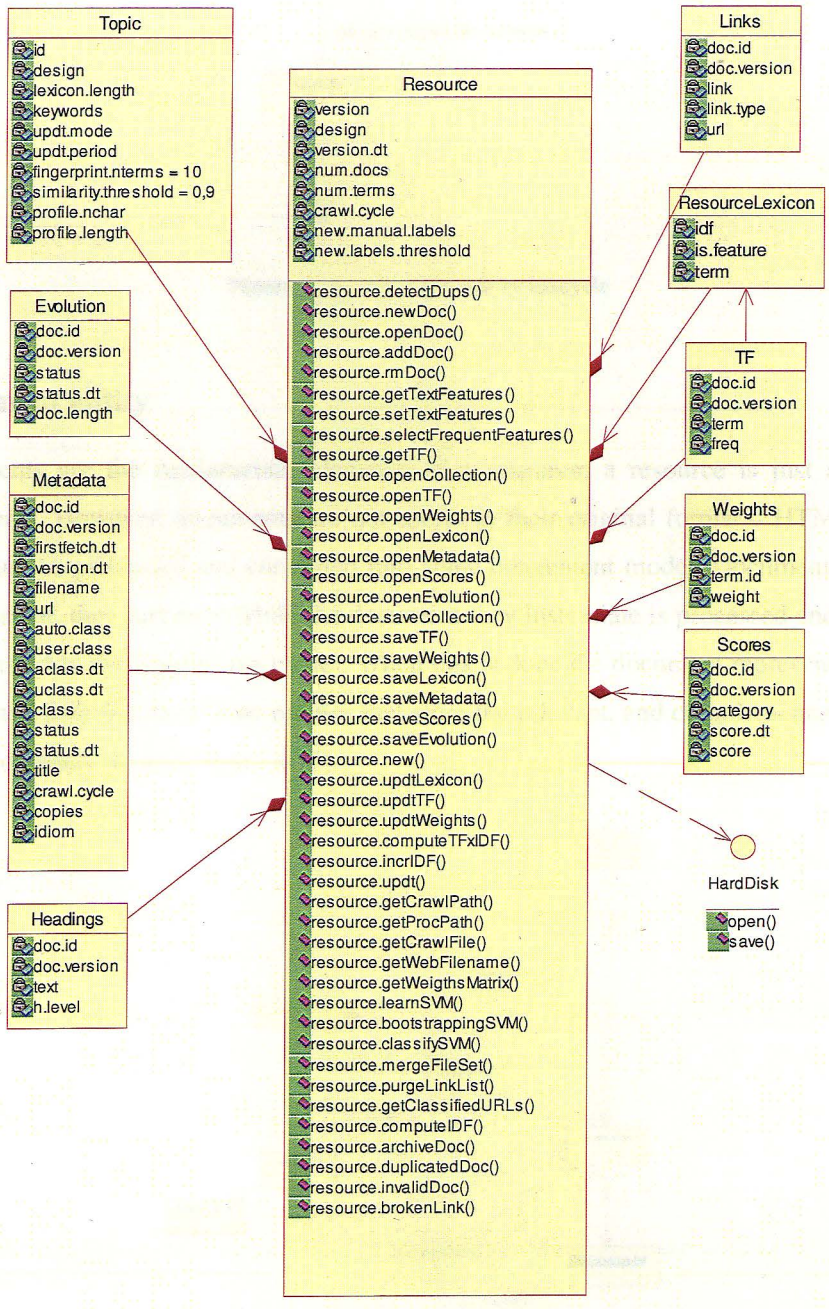


Figure 4.13 – Resource entity, attributes and methods

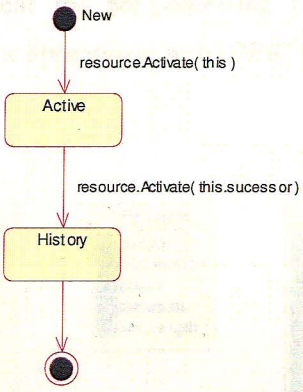


Figure 4.14 – Resource entity lifecycle

Document entity

Documents are the fundamental elements of a resource; a resource is just a set of documents. However, documents are not useful in their original format – HTML files; they must be processed and converted into some convenient model. Document objects are transient, they just exist while the document they instantiate is processed and until it is converted to the appropriate model. When this is done the document representation is distributed over the set of nine entities that store the relevant, and complementary, parts of the document model (Figure 4.15).

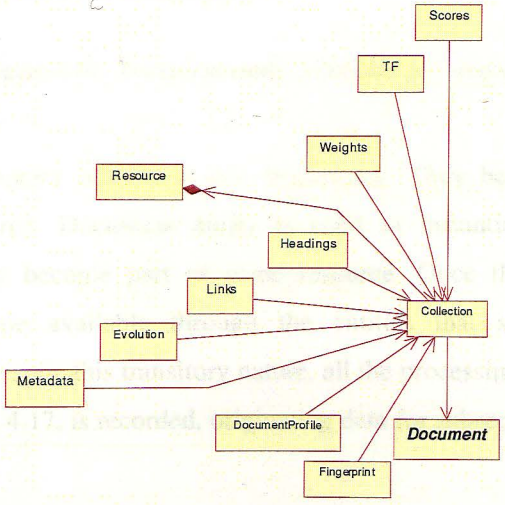


Figure 4.15 – Document entity, relationships

Document methods implement the pre-processing phase and are responsible for converting an HTML file to the appropriate webTOPIC document model.

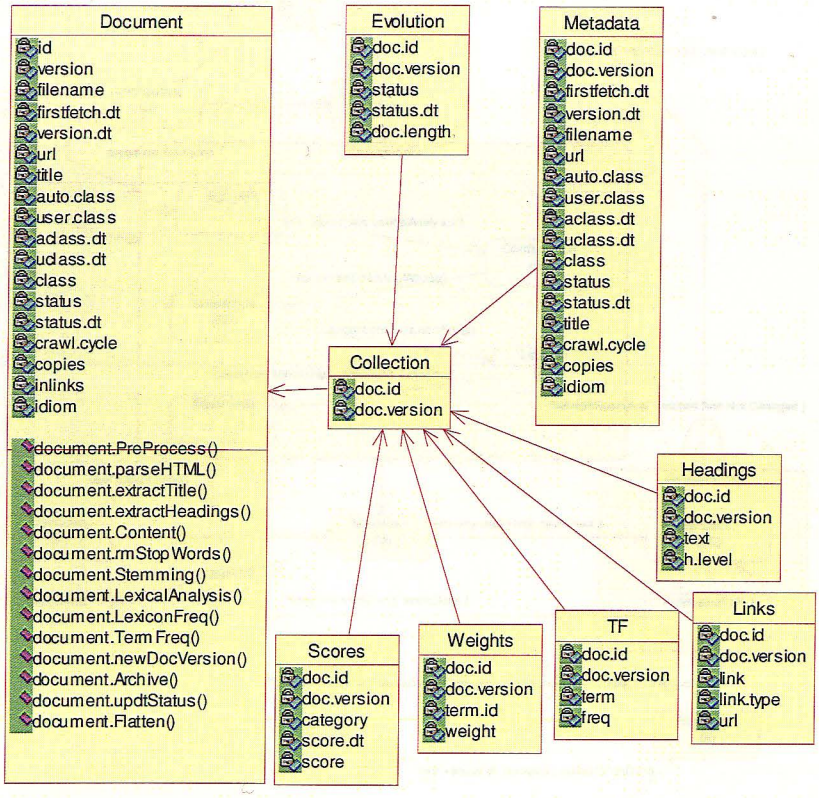


Figure 4.16 – Document entity, attributes and methods

Documents are temporary objects in our framework. They become persistent once included in a resource. Document entity is used to instantiate and process new documents until they become part of some resource. Once they are added to the resource they become available through the entities that store and manipulate document's model. Despite this transitory nature, all the processing they undergo during their lifecycle, Figure 4.17, is recorded, originating data for subsequent analysis.

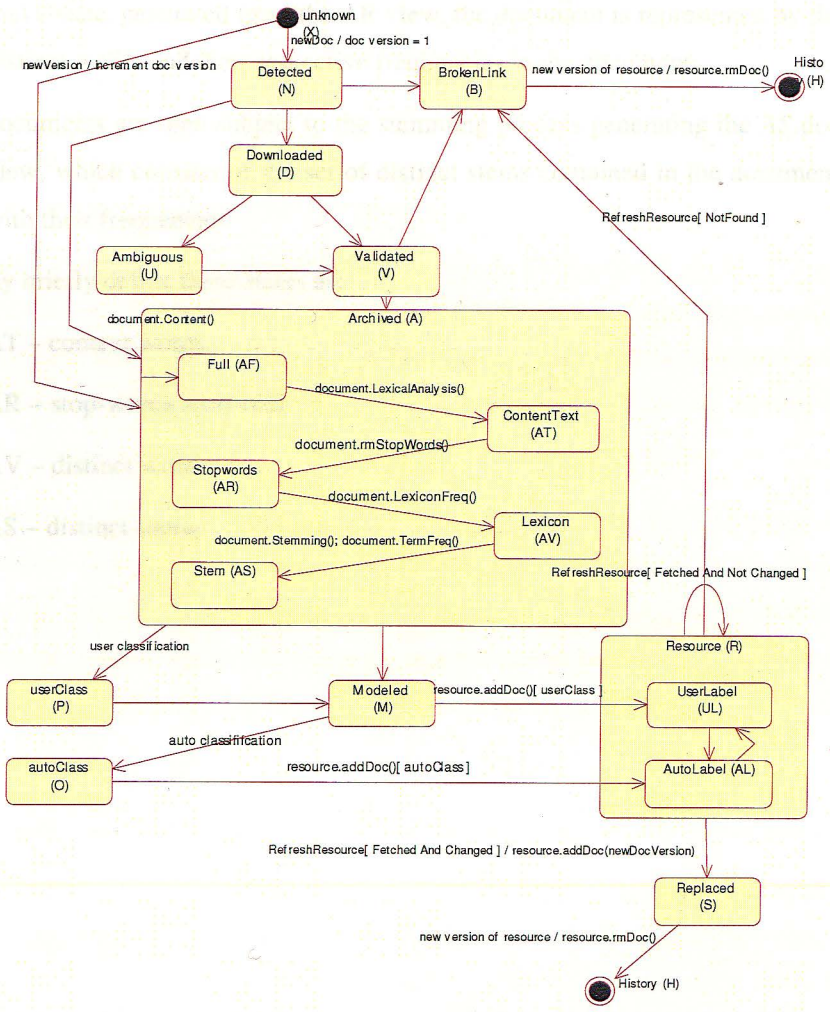


Figure 4.17 – Document entity lifecycle

During the pre-processing task the internal view of the document changes a few times. Each distinct view corresponds to a different state in the document lifecycle and also to a different representation of the document, which is consecutively simpler:

- in *AT* state the document is represented by all of its content text words; HTML tags have been removed;
- the document *AR* representation is generated from the AT document view by removing stop-words;

- in *AV* state, generated from the AR view, the document is represented by the set of distinct words and their respective frequencies in the document;
- documents are then subject to the stemming process generating the *AS* document view, which consists of the set of distinct stems contained in the document along with their frequencies.

We may briefly define these states as:

- AT – content words;
- AR – stop-words removed;
- AV – distinct words;
- AS – distinct stems.

5 Experimental evaluation

At this stage it seems important to gain some insight into one of the most critical tasks of the webTOPIC methodology: the learning task. The accurate classification of web pages and, consequently, the correct organization of the resource, based on a small set of just a few manually labeled examples, are crucial to the global performance of the entire system that depends mostly on the learning component. Without a reliable classifier for the web environment – which demands for expensive human effort – resources cannot be conveniently organized in a semi-automatic fashion.

As previously discussed, the application of semi-supervised methods may improve classification tasks that are based on just a few labeled examples. In this chapter we describe the experimental study that has been conducted with the goal of evaluating the adequacy of semi-supervised learning for the automatic resource compilation setting. We will try to analyze two fundamental aspects of the classification task:

- accuracy of the classifier on its own and
- robustness against human errors during the initial pre-classification task.

The learning methods used are based on the SVMLight classifier (Joachims, 1999; http://svmlight.joachims.org/) and on a semi-supervised bootstrapping algorithm.

5.1 Experimental setup

We start by describing the resource acquisition process that has been employed and by characterizing the data sets we have obtained. Next we proceed with a preliminary analysis of the properties of the data sets, mainly exploring term distributions and document compression rates that were achieved during the pre-processing phase. Then we will compute an estimate for the classifier accuracy in a supervised setting, which will be used as our reference. From this reference, we evaluate the semi-supervised bootstrapping SVM capabilities in an automatic web resource compilation scenario.

Our experiences will be conducted as follows:

- Experiences 1 – in this set of experiences, described in section 5.5, we intend to determine the accuracy of our classifier in a supervised setting. These results will serve as a reference.
- Experiences 2 – described in section 5.6, determine the classifier accuracy when operating in a semi-supervised setting. Accuracy will be computed for both random and stratified training samples.
- Experiences 3 – these experiences, described in section 5.7, are intended to evaluate the robustness of supervised and semi-supervised classifiers, considering random and systematic errors in human classification, committed at the pre-classification stage.

5.2 Data set acquisition and characterization

We have chosen two distinct topics to perform our experiences: *Artificial Intelligence*, which is of relevance for us, and *Popular Events*, which is a topic that easily provides document collections. We are interested in organizing these topics according to the following taxonomies:

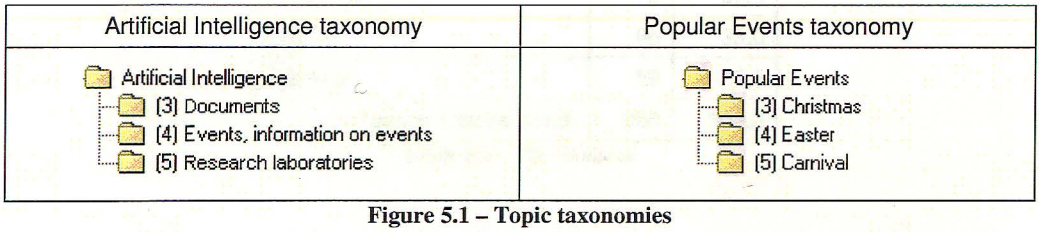


Figure 5.1 – Topic taxonomies

The topic definition still requires the specification of a set of keywords. All the documents from the Artificial Intelligence (AI) resource were retrieved with a unique query, defined by the keywords "*inteligência*" and "*artificial*". To retrieve the documents on the Popular Events (PE) topic we have submitted three queries, each one with a single keyword: "*Natal*" for the query submitted to obtain documents for the category Christmas, "*Páscoa*" for the Easter category and "*Carnaval*" for the Carnival category. With these definitions, primary data sets for each topic were obtained and

manually classified using our webTOPIC prototype capabilities. These document collections constitute two resources, one for each topic. The resource on the topic AI has 66 documents and the resource on PE is constituted by 200 documents. All documents at both resources are required to be written in Portuguese, since we are using a stemming algorithm and a stop-words list for the Portuguese language; this requirement became a major drawback when acquiring resources, especially on the topic AI.

Both resources were fully labeled on their respective taxonomies, observing the following distributions:

Category	#	%
Documents	22	33,3
Events, information on events	23	34,8
Research laboratories	21	31,8
Artificial Intelligence (total)	66	100,0

Table 5.1 – AI resource

Category	#	%
Christmas	75	37,5
Easter	66	33,0
Carnival	59	29,5
Popular Events (total)	200	100,0

Table 5.2 – PE resource

These documents, from both topics, were processed by the webTOPIC prototype generating two resources that are internally supported by a set of informational structures, among which four of them that are of special relevance at the moment:

- a document-by-term matrix, stored at the entity *Weights*, where each row is the TF×IDF model representation of a document in the resource;
- another document-by-term matrix, entity *TF*, where each row is the term frequency representation of a document;

- the resource lexicon, managed by *ResourceLexicon* entity, which contains the vector of the distinct stems in the corpus – the resource vocabulary – and a vector with the the inverse document frequency of each stem – the IDF vector;
- the evolution of each document belonging to the resource, through its lifecycle (Figure 4.17) is recorded by the *Evolution* entity.

During our experiences we will rely exclusively on the information that is generated and processed by the webTOPIC prototype.

5.3 Exploratory data analysis

Before carrying out the experiences, we have performed a brief study on the pre-processing task and also on the characteristics of our resources. Since the pre-processing task is very demanding, requiring a significant computational effort, we wish to analyze the compression rates that this task is capable of.

5.3.1 Pre-processing compression rates

We have evaluated the value of the pre-processing task in the global process based on the number of terms that are present at each subsequent document view generated during the document lifecycle phases (see section Document entity):

- AT – content words
- AR – stop-words removed
- AV – distinct words
- AS – distinct stems

We have collected a few statistics on the number of terms that constitute each document's view through pre-processing and present them at Table 5.3 and Table 5.4.

From these data we observe that the dispersion of the number of terms by document is high at both resources. This is expected since web pages may be very short on content text as well as they may contain heavy content text. For instance, we have detected on our PE resource some pages that just had a picture and no text at all, except for the title, while others have several text pages, for instance, telling Christmas tales.

Artificial Intelligence	AT	AR	AV	AS
Mean	181,3	137,3	97,1	87,5
Standard deviation	42,6	57,0	58,3	59,7
Max	253	236	222	221
Min	78	42	5	3

Table 5.3 – AI resource terms per document

Popular Events	AT	AR	AV	AS
Mean	228,8	141,6	99,4	93,0
Standard deviation	185,3	111,9	76,6	69,3
Max	908	535	383	337
Min	13	9	5	5

Table 5.4 – PE resource terms per document

We will focus on the mean number of terms per document and on its evolution during pre-processing. The reduction in the number of terms required to model documents at the different states of their lifecycle is notorious, falling down to less than half its initial value at the end of the pre-processing stage (Table 5.3, Table 5.4). The distribution of the number of stems per document can be analyzed through their histograms (Figure 5.2).

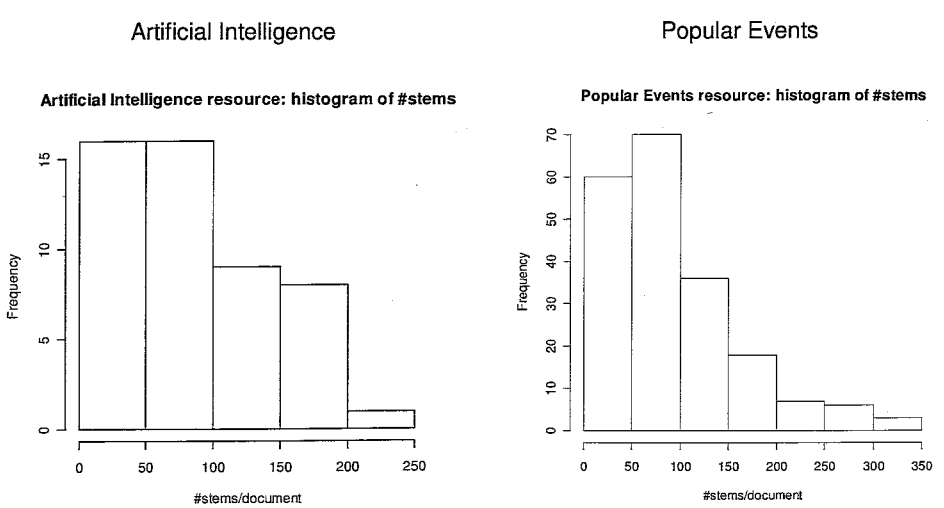


Figure 5.2 – Stems per document histograms

The AI resource may be divided into three groups concerning the stem distribution: one consisting of nearly half the 66 documents of the resource that have less than 100 stems, another one with documents that have between 100 and 200 stems and a last one with a single document with more than 200 stems. The PE resource also has the majority of the documents with less than 100 stems; however it presents a longer right tail than the one we have observed on AI resource.

The distribution of the number of stems per document – AS document view, stratified according to topic's taxonomy, may be made clearer with a set of Box-Whiskers plots:

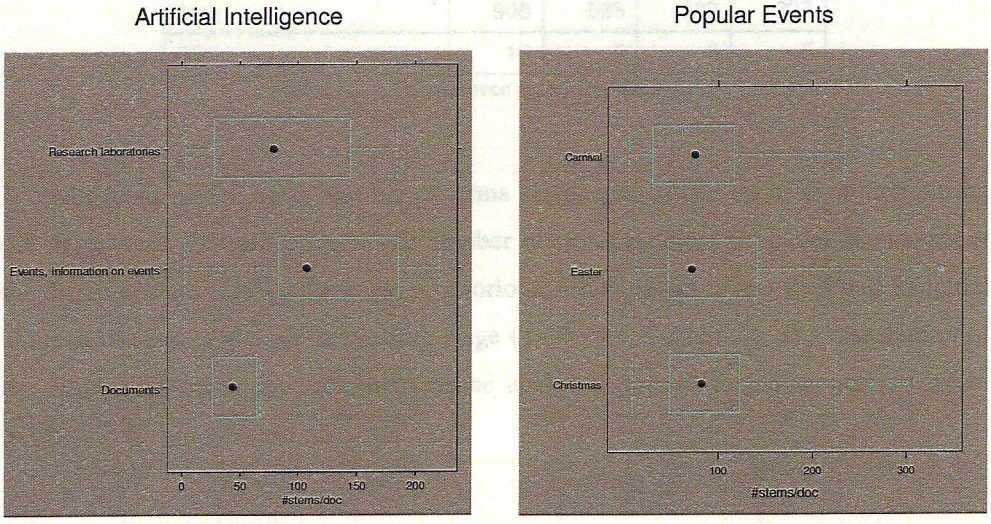


Figure 5.3 – Stratified distribution of stems per document

All the categories of PE taxonomy present very similar distributions. Although the majority of documents have less than 100 stems, these distributions have long right tails corresponding to a few outliers with around 300 stems. Medians for all the categories are very close to each other. On the AI resource the median number of stems per document exhibits a higher dependency on the category than on the PE topic. On the other hand the dispersion is generally lower on the AI resource.

Table 5.5 and Table 5.6 present compression rates that are obtained on the document's representation at each pre-processing state – compared with the previous state – as well as accumulated compression rates – compared to the AT state.

Compression rate from previous state	AT	AR	AV	AS
Documents	1,000	0,636	0,653	0,827
Events, information on events	1,000	0,868	0,744	0,938
Research laboratories	1,000	0,760	0,705	0,913
Artificial Intelligence (total)	1,000	0,757	0,707	0,901

Accumulated compression rate from AT state	AT	AR	AV	AS
Documents	1,000	0,636	0,416	0,344
Events, information on events	1,000	0,868	0,646	0,606
Research laboratories	1,000	0,760	0,536	0,489
Artificial Intelligence (total)	1,000	0,757	0,536	0,483

Table 5.5 – Compression rates achieved at AI resource

Compression rate from previous state	AT	AR	AV	AS
Christmas	1,000	0,624	0,68	0,932
Easter	1,000	0,605	0,719	0,931
Carnival	1,000	0,629	0,716	0,945
Popular events (total)	1,000	0,619	0,702	0,936

Accumulated compression rate from AT state	AT	AR	AV	AS
Christmas	1,000	0,624	0,424	0,396
Easter	1,000	0,605	0,435	0,405
Carnival	1,000	0,629	0,451	0,426
Popular events (total)	1,000	0,619	0,434	0,406

Table 5.6 – Compression rates achieved at PE resource

Every step in the pre-processing task reduces the feature space dimension. In the final document representation, the AS view, the number of text features has been reduced to 40% of its original value on PE and to 48% on the AI resource. Stemming is the least significant step; it reduces feature space around 3% on PE and around 5% on AI, while the other steps – removing stop-words and eliminating duplicated words – reduce the feature space to 60% and 40%, respectively, on PE, and to 75% and 50%, on AI.

These observations illustrate the relevance of the pre-processing task. Although it demands significant computational resources, it substantially reduces the effort required by the subsequent tasks. In particular, the learning task benefits from the reduction of the feature space.

5.3.2 Lexicon growth

Topic lexicon grows as new documents are added to the resource. We have recorded lexicon growth for both topics. Figure 5.4 presents the evolution of the size of the topic lexicon as a function of the number of documents in the resource.

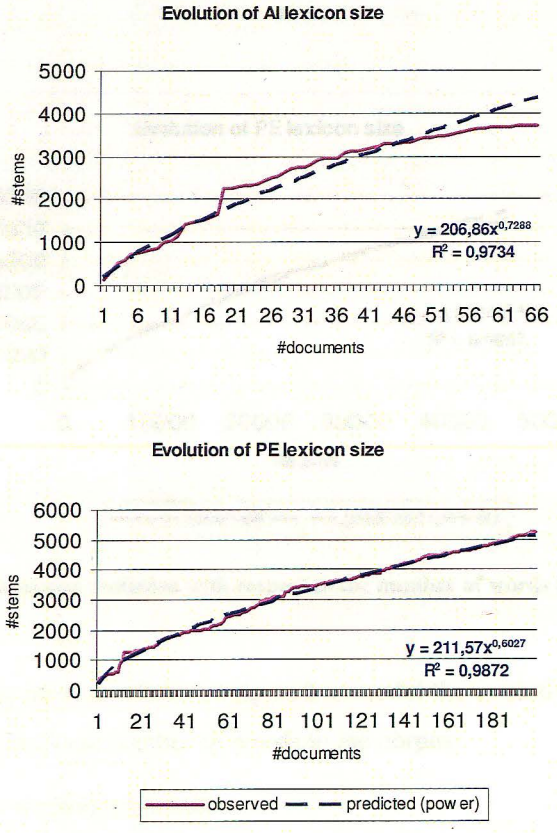


Figure 5.4 – Lexicon size evolution with respect to the number of documents in the resource

The evolution of lexicon size may be predicted with a power law, which allows predicting the dimension of text features space as new documents are added to the resource.

Figure 5.5 presents the evolution of the size of the topic lexicon as a function of the number of words in the resource.

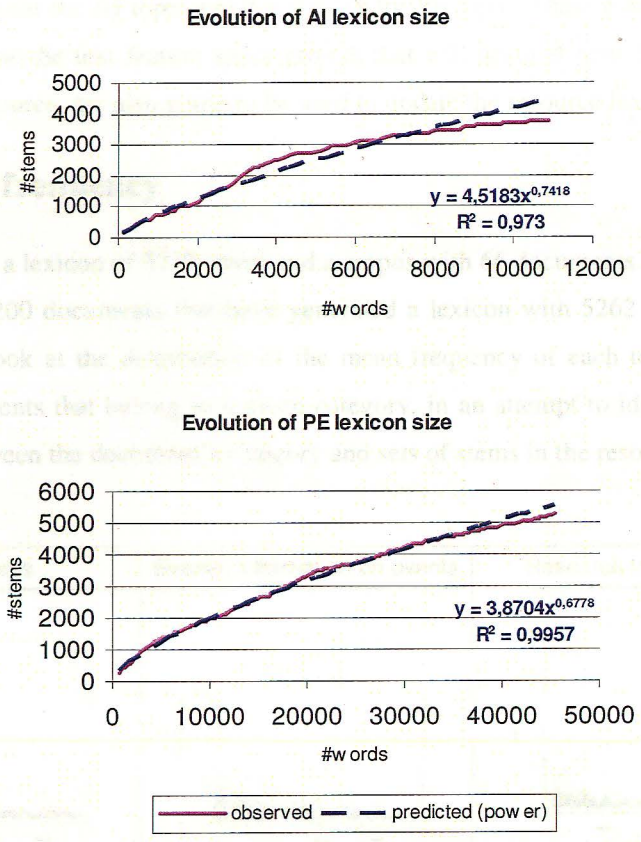


Figure 5.5 – Lexicon size evolution with respect to the number of words in the resource

Power law trend lines have also been computed to model the evolution of topic lexicon size, as a function of the total number of words in the corpus:

- for the AI topic we have obtained:

$$Y = 4{,}518 * X^{0{,}7418}, \text{ with a } R^2 \text{ value of } 0{,}973$$

- for the PE topic we have obtained:

$$Y = 3{,}870 * X^{0{,}8778}, \text{ with an } R^2 \text{ value of } 0{,}996$$

Y is the estimated lexicon size for a corpus with X words.

Both R^2 values are higher than 97%, confirming the validity of these estimators. Heap's law (Baeza-Yates et al, 1999) – an empirical rule that establishes that a text of X words has a vocabulary of size, Y, proportional to X^{β}, for $0 < \beta < 1$ – applies to both topics, with $\beta = 0{,}7418$ for the AI topic and $\beta = 0{,}8778$ for PE topic. These estimators may be used to anticipate the text feature space growth that will occur if new documents, just added to the resource, are also going to be used to update the resource lexicon.

5.3.3 Term frequency

AI resource has a lexicon of 3732 stems and a corpus with 66 documents. PE resource is constituted by 200 documents that have generated a lexicon with 5262 distinct stems. We will now look at the distribution of the mean frequency of each term, computed over the documents that belong to a given category, in an attempt to identify eventual correlation between the document's category and sets of stems in the resource lexicon.

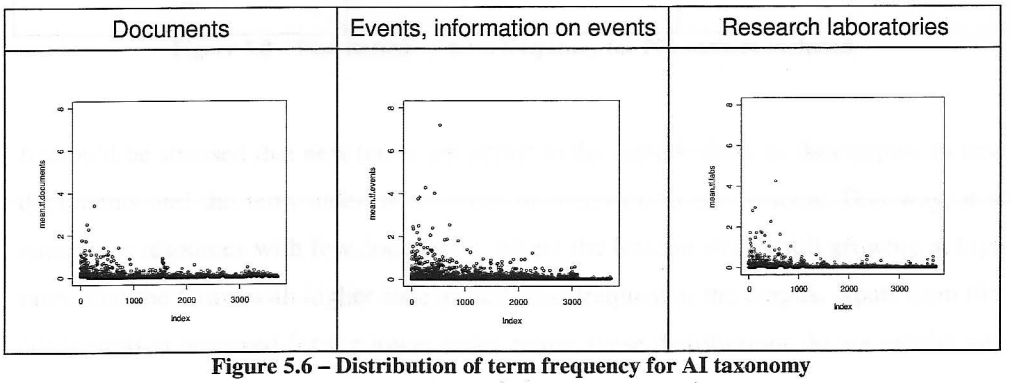


Figure 5.6 – Distribution of term frequency for AI taxonomy

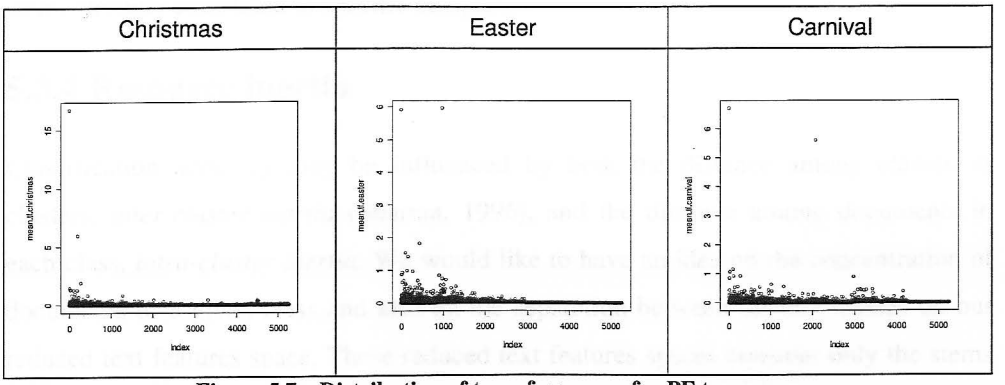


Figure 5.7 – Distribution of term frequency for PE taxonomy

We have computed the mean number of occurrences of each stem over all the documents from a given category. Figure 5.6, for AI resource, and Figure 5.7, for PE resource, show these figures for each stem that is represented by its index in the lexicon.

The distribution for PE can be better perceived if we eliminate the outliers, which hide low frequency terms. We have generated the same plots but considering exclusively the terms that have mean IDF lower than 2.

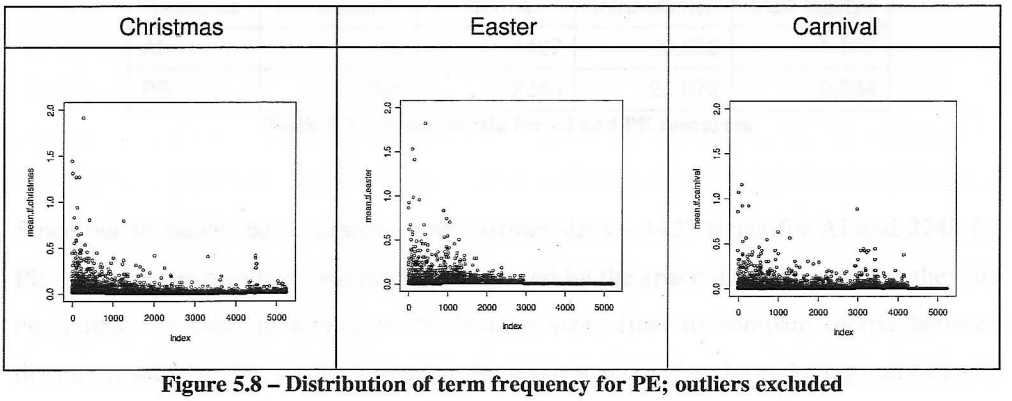


Figure 5.8 – Distribution of term frequency for PE; outliers excluded

It should be stressed that new terms are added to the topic lexicon as they appear in new documents and the term index is its order of inclusion in the lexicon. This way, it is natural, in resources with few documents, where the lexicon size is still growing at high rates, that the terms with higher indexes are least frequent at the corpus. Apart from this concentration observed for the lower index terms, these distributions do not exhibit any evident pattern, thus we may not assume that in our resources there are any subsets of stems strongly correlated to specific categories.

5.3.4 Resource inertia

Classification accuracy may be influenced by both the distance among classes or clusters, *inter-cluster inertia* (Sharma, 1996), and the distance among documents in each class, *intra-cluster inertia*. We would like to have an idea on the concentration of documents of a given class and also on the separation between distinct classes on our reduced text features space. These reduced text features spaces consider only the stems that have an IDF value greater than 1 – 1427 and 2240 terms, respectively for AI and

PE resources. We can measure these resource characteristics with intra and inter-cluster inertia, respectively. To compute them we will define document similarity as the square of the Euclidean distance between the TF×IDF vectors that represent each document on its text features space. Class gravity centers are the mean vectors computed over the document that belong to each class in the topic taxonomy.

			Inertia	
Resource	#documents	#terms	Intra-cluster	Inter-cluster
AI	66	1427	20,256	1,107
PE	200	2240	20,072	0,864

Table 5.7 – Topic inertia for AI and PE resources

Since our resources have lexicons with distinct sizes – 1427 terms for AI and 2240 for PE – and inertia measures are highly influenced by the space dimension where they are computed, we wish to eliminate the lexicon size effect to compare inertia between distinct resources. We have eliminated the lexicon size effect by reducing inter-cluster and intra-cluster inertia to a normalized length of 1000 terms; thus, absolute values were divided by 1,427 for the AI resource and by 2,240 for PE resource.

	Inertia	
Resource	Intra-cluster	Inter-cluster
AI	14,195	0,776
PE	8,961	0,386

Table 5.8 – Normalized resource inertia

Ideally intra-cluster inertia should be minimized while inter-cluster inertia should be maximized. We observe that normalized intra-cluster inertia of PE is nearly half the value that is observed for AI; this means that documents of a given class are closer to each other on the PE resource than on the AI one, which results in smaller class radius – more concentration – on the PE resource. However, the PE taxonomy is composed of a set of classes that are much closer to each other than those at the AI taxonomy;

inter-cluster inertia is much greater at the AI resource, which means that classes in this resource are farther apart than on PE.

5.3.5 Inverse document frequency

We now study the *inverse document frequency*, IDF, of the terms in the resources' lexicon. To analyze the evolution of IDF we have computed the number of terms appearing in any document – lexicon size – and the number of terms with IDF values between 1 and 30.

Figure 5.9 shows how many terms that appear in 1 to 30 documents. There we can observe a great reduction in the number of terms that have an IDF value greater than 1 – the set of terms that appear in more than one single document. So, we select as text features just those terms that have an IDF greater than 1, reducing 57% of text features on PE topic, whose lexicon is reduced from 5262 to 2240 terms, and 62% on AI, reducing the lexicon from 3732 to 1427 terms. This allows for a significant reduction in the feature space dimension without loosing much discriminative power since terms that appear in just one document are irrelevant for the classification task (Yang et al, 1997). This way the effective dimensions of document-by-term weight matrices become 200×2240 for PE and 66×1427 for the AI topic.

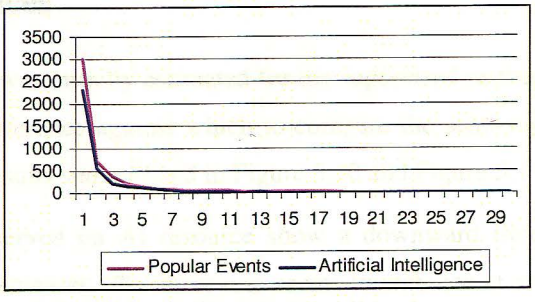


Figure 5.9 – IDF distribution

5.4 Accuracy

After this exploratory stage we computed our classifier's accuracy on both resources. From the different aspects on the document model of webTOPIC – textual, structural,

neighborhood and metadata features – we will only use the text features subset where documents are represented by TF×IDF vectors. Given the characteristics observed in the exploratory step, we will work on the reduced text features space (only terms with IDF greater than 1) – this way the resource on AI is represented by a 66×1427 weights matrix and the resource on PE is represented by a 200×2240 weights matrix. All documents in both resources are fully labeled.

5.5 Supervised accuracy

5.5.1 Training and testing data sets

To estimate the accuracy of our classifier we have generated several partitions of the resource, obtained by random sampling. Each partition divides the resource into two subsets: one of them is used to train and the other one to test the classifier. Training data sets are constituted by a number of documents that ranges from 3 to 60, in the case of AI resource, and from 3 to 180, in the PE resource in multiples of three – the maximum train size is 90% of the resource size, thus guarantying at least a minimum of 10% of the documents to test. We have generated 10 random samples for each cardinality of the training data set and estimate error rates by the mean computed over these 10 samples.

5.5.2 Experiences

Classifier accuracy was initially estimated for the supervised setting; this will serve as a benchmark and a reference against which to compare the semi-supervised setting. We have obtained the results summarized in Figure 5.10 and Figure 5.11.

The error rates observed on AI resource show a downward trend as the number of training examples increases. The error rate seems to stabilize at around 25% when the training data set has 42 examples. From this level on, increasing the number of training examples up to 51 does not improve the error rate that is kept between 23% and 26%. Training with 54 and 57 training examples reduces the error rate to 20% and 14%, respectively. The last experience was made with 60 training examples and 6 testing examples; in this case the error rate increases from 14% to 18%.

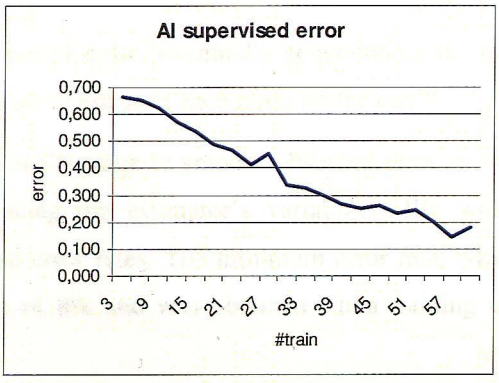


Figure 5.10 – Supervised error rate for AI resource

Given this behavior we will assume as our reference for the supervised error on AI resource the value of 25%, which has been obtained with 42 examples to train and 24 examples to test. Although this is not the lowest error obtained, it appears to be an inflexion point in the curve of the error evolution. Besides it corresponds to an experimental configuration with 2/3 of the available examples used to train and 1/3 to test, which is a typical configuration and prevents over-fitting.

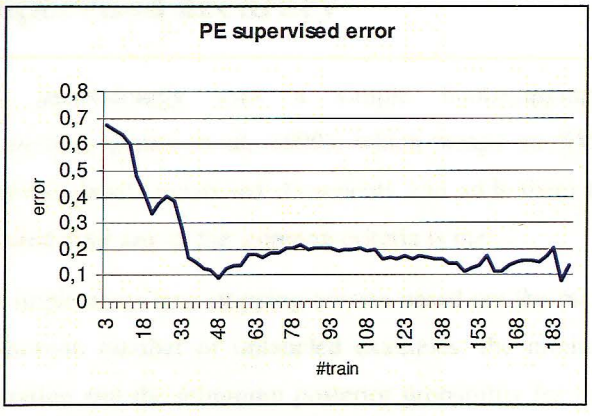


Figure 5.11 – Supervised error rate for PE resource

The error rates observed at PE resource (Figure 5.11) have a typical behavior: error decreases until a certain minimum and from there on it inverts this tendency and starts increasing, due to over-fitting. Although this evolution is not surprising, observed error

rates denote the presence of noise, eventually generated by the number of experiences, (10), that were performed to estimate each point on the curve.

We have applied moving average to smooth observed error in order to eliminate high frequency noise, reducing the estimator's variability. We will use these estimates instead of the observed error rates. The minimum error rate, which we will use as our reference, has a value of 8% and was obtained when training with 45 examples and testing with 155.

We will then consider the following reference values, obtained on the supervised setting:

	Artificial Intelligence	Popular Events
Generalization error	25%	8%
#Training data set	42	45
#Testing data set	24	155

Table 5.9 – Supervised accuracy

5.6 Semi-supervised accuracy

The webTOPIC methodology uses a simple bootstrapping algorithm for semi-supervised learning (Jones et al, 1999), which wraps an SVM classifier in a process that iteratively labels unlabeled documents and adds them to the labeled set. This cycle is executed until any of the stopping criteria is met.

Our methodology implements four stopping criteria based on: the maximum number of iterations, the minimum number of unlabeled examples, the minimum classification gradient at the iteration and the minimum posterior probability for the new labels. We will now apply this algorithm to compute the semi-supervised accuracy for both resources.

5.6.1 Training and testing data sets

The reference values we have previously obtained at the supervised setting – an error rate of approximately 25% with 42 training and 24 testing documents, for the AI resource, and an error rate of approximately 8% with 45 training and 155 testing documents, for the PE resource – will enable us to evaluate the semi-supervised setting. So, we will assess the semi-supervised classification algorithm on several training data sets that will be derived from 45 training examples and 155 testing examples, for the PE resource, and from 42 training examples and 24 testing examples, for the AI resource. For each experience the training data will be split in two parts: one where document labels are made available to the classifier and another one where document labels are hidden from the classifier. To obtain the evolution of the error rate by workload – proportional to the number of labeled examples – the number of documents whose labels are known to the classifier is successively increased. This way, we have prepared the following experiences for both topics – for each experience there are 10 distinct training data sets, corresponding to 10 samples from the resource:

Experiences for the PE resource

Training data set		Testing data set
#Labeled	#Unlabeled	#Test
3	42	155
6	39	155
9	36	155
12	33	155
15	30	155
18	27	155
21	24	155
24	21	155
27	18	155
30	15	155
33	12	155
36	9	155
39	6	155
42	3	155
45*	0*	155*

Experiences for the AI resource

Training data set		Testing data set
#Labeled	#Unlabeled	#Test
3	39	24
6	36	24
9	33	24
12	30	24
15	27	24
18	24	24
21	21	24
24	18	24
27	15	24
30	12	24
33	9	24
36	6	24
39	3	24
42*	0*	24*

Legend: * corresponds to the supervised setting

Table 5.10 – Experiences design for semi-supervised accuracy

Error rates were obtained through the mean, computed over the 10 samples for each experience, just like in the supervised setting. Training examples were obtained, for each experience, by two distinct methods: random sampling – which we will refer to by *random* – and stratified sampling – which we will refer to by *stratified*. Random samples are the same that have been used in the supervised setting. Stratified sampling is analyzed because we wish to understand the influence that an asymmetric set of pre-labeled documents might have on the performance of semi-supervised classification, which may suggest special care on this subject at the pre-classification stage of the methodology.

5.6.2 Experiences

Bootstrapping process

Before analyzing error rates we will pay some attention to the bootstrapping process itself; in particular we have observed the evolution of the number of iterations and the evolution of the classification gradient as the number of labeled documents increases. This data is summarized on Figure 5.14 and Figure 5.15 that show the mean number of iterations and the mean classification gradients observed at the bootstrapping cycles for both resources. The values that were observed at the experiences are registered at Table 5.11 and Table 5.12.

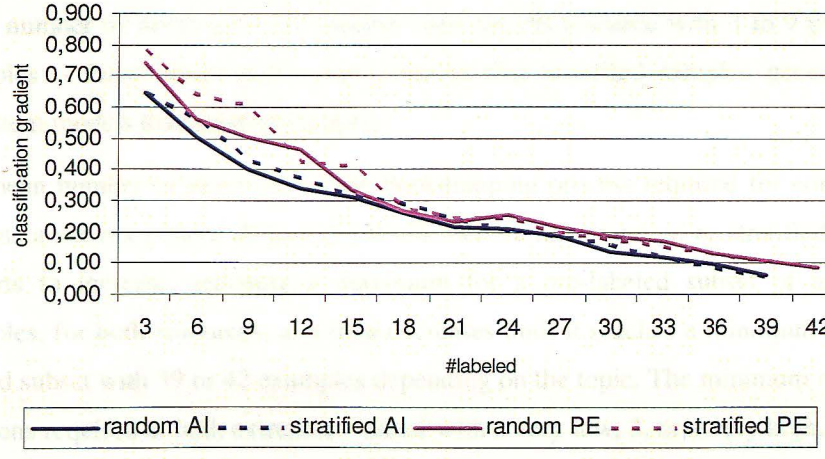


Figure 5.12 – Evolution of classification gradient

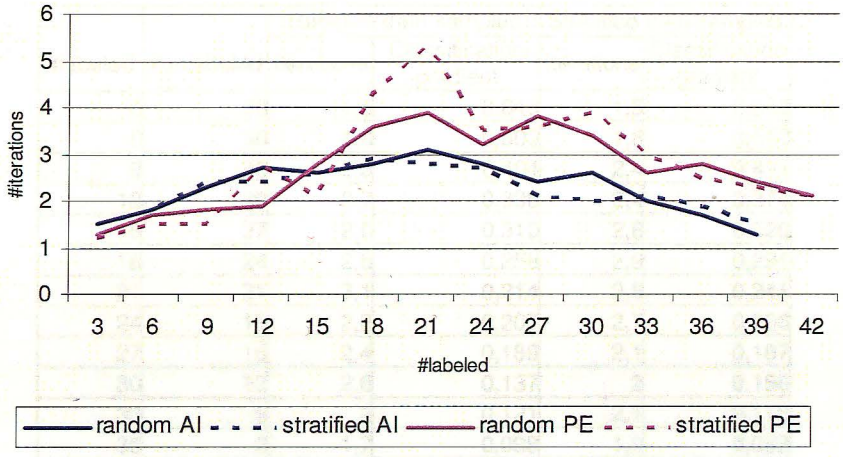


Figure 5.13 – Evolution of required number of iterations

We may observe that the classification gradient is monotonically decreasing as the number of pre-labeled documents increases. The class distribution of the pre-labeled subset does not significantly influence classification gradient. The evolution of mean classification gradient over bootstrapping iterations is nearly the same whether pre-labeled examples are obtained through random or stratified sampling.

The values of the classification gradient observed with few pre-labeled examples (from 3 to 21, on both resources) are consistently higher for the stratified training samples when compared to the random samples on the same resource. When combined with a lower number of iterations – as happens with the PE resource with 3 to 9 pre-labeled examples – these observations may indicate that stratified samples generate more consistent models than random samples.

The mean number of iterations of the bootstrapping process required for convergence presents a convex pattern (Figure 5.13) whether we have random or stratified samples. It starts to increase, reaching a maximum for a pre-labeled subset of around 21 examples, for both resources, and then decreases until it reaches a minimum for a pre-labeled subset with 39 or 42 examples depending on the topic. The minimum number of iterations required at both extremes – either with a very low, 3, or a very high, 39 or 42, number of pre-labeled examples – are approximately the same at both resources.

#labeled	#unlabeled	Random train samples		Stratified train samples	
		Iterations	Classification gradient	Iterations	Classification gradient
3	39	1,5	0,648	1,5	0,648
6	36	1,8	0,503	1,8	0,558
9	33	2,3	0,404	2,4	0,431
12	30	2,7	0,338	2,4	0,374
15	27	2,6	0,310	2,6	0,320
18	24	2,8	0,258	2,9	0,291
21	21	3,1	0,214	2,8	0,241
24	18	2,8	0,207	2,7	0,206
27	15	2,4	0,189	2,1	0,187
30	12	2,6	0,137	2	0,156
33	9	2	0,121	2,1	0,118
36	6	1,7	0,098	1,9	0,083
39	3	1,3	0,064	1,5	0,059

Table 5.11 – Evolution of bootstrapping for AI

#labeled	#unlabeled	Random train samples		Stratified train samples	
		Iterations	Classification gradient	Iterations	Classification gradient
3	42	1,3	0,740	1,2	0,783
6	39	1,7	0,558	1,5	0,641
9	36	1,8	0,503	1,5	0,608
12	33	1,9	0,466	2,7	0,424
15	30	2,8	0,336	2,1	0,411
18	27	3,6	0,268	4,3	0,278
21	24	3,9	0,230	5,3	0,242
24	21	3,2	0,254	3,5	0,243
27	18	3,8	0,213	3,6	0,199
30	15	3,4	0,186	3,9	0,177
33	12	2,6	0,170	3	0,152
36	9	2,8	0,129	2,5	0,129
39	6	2,4	0,107	2,3	0,107
42	3	2,1	0,083	2,1	0,083

Table 5.12 – Evolution of bootstrapping for PE

Accuracy

After these observations on the semi-supervised process, we will focus on the generalization error of semi-supervised classification. We have run the prescribed experiences and computed error rates estimates. The results are presented in Figure 5.14 and Figure 5.15. We intend to compare random and stratified semi-supervised error with our benchmark, the supervised error reference.

In the case of the AI resource the performance obtained with the stratified samples tends to be higher than with the random samples. For an error rate of 40% we may see that it takes 27 randomly selected labeled examples against 15 labeled examples from a stratified sample. As to our reference from the supervised setting – an error rate of 25% obtained with 42 labeled examples – we observe that the semi-supervised setting may improve it greatly, with an error rate of 25% with 30 labeled examples (stratified). With 36 labeled random examples we reach an error rate of 27%.

We may conclude that, at our AI resource, the semi-supervised learning algorithm is able of leveraging evidence from labeled and unlabeled examples, significantly reducing the workload required to obtain a certain accuracy level, especially if the distribution of labels in the set of pre-labeled examples reflects the distribution of labels in the resource.

The advantage of semi-supervised techniques becomes clear at this point. We had already discussed the need for learning techniques that could learn with just a few labeled examples. These experiences seem to confirm the validity of semi-supervised learning techniques and to prove that it allows the reduction of the workload required to obtain a certain accuracy level. With a workload of 45 labeled documents we have achieved a minimum supervised error of 8%. Applying semi-supervised learning, with the same classifier, we have an equivalent error rate, of 10%, for a workload of just 33 documents from stratified samples, a reduction of 27% on the workload, or 39 documents from random samples, a reduction of 13% on the workload.

These experiences confirm the advantages of semi-supervised learning for the web environment, especially on the workload reduction they allow without compromising accuracy.

Finally we would like to analyze if there is any relation between error rate and classification gradient. We have plotted them, one against the other, for the case of the stratified sample, to evaluate any correlation between these two variables (Figure 5.16).

It seems evident that there is a positive correlation between the error rate and the classification gradient. High classification gradients may mean that classifiers are not too confident and constantly change their evaluations, from iteration to iteration, which may contribute for the low accuracy that is observed under these circumstances. The

intuition is that classifiers that learn consistently slightly improve their capabilities at each iteration, always moving towards the same direction; in these circumstances classification gradient is low and learned models are accurate. On the contrary, if the classifiers that are learnt at each step are over-fitted, following irrelevant noise, they are constantly changing their opinions on previously known documents. As a consequence, the classifications they generate change frequently, originating high classification gradients and a final model that is inaccurate.

	Error	
#labeled	random	stratified
3	0,65	0,65
6	0,61	0,54
9	0,68	0,49
12	0,70	0,47
15	0,70	0,38
18	0,60	0,37
21	0,60	0,28
24	0,46	0,33
27	0,40	0,30
30	0,34	0,25
33	0,32	0,26
36	0,27	0,25
39	0,28	0,25

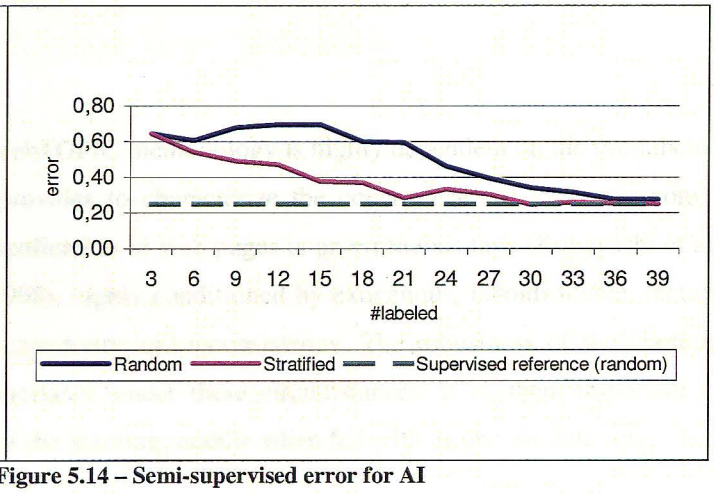


Figure 5.14 – Semi-supervised error for AI

	Error	
#labeled	random	stratified
3	0,54	0,55
6	0,64	0,54
9	0,55	0,35
12	0,57	0,40
15	0,57	0,33
18	0,54	0,44
21	0,60	0,49
24	0,32	0,25
27	0,32	0,18
30	0,27	0,17
33	0,17	0,10
36	0,14	0,10
39	0,11	0,08
42	0,09	0,08

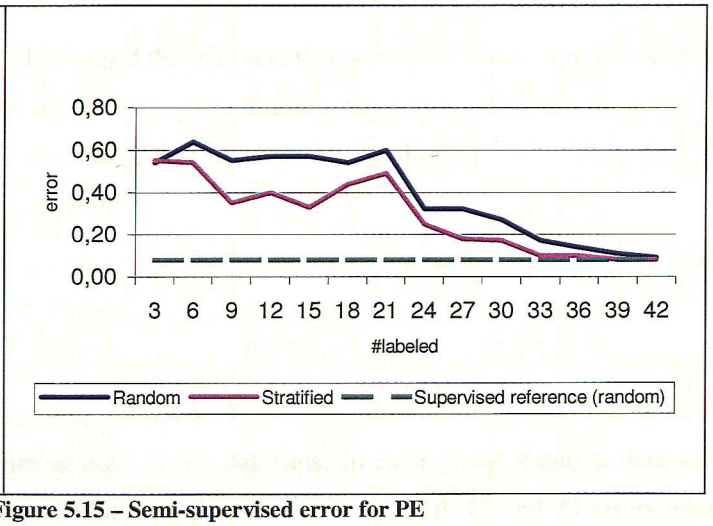


Figure 5.15 – Semi-supervised error for PE

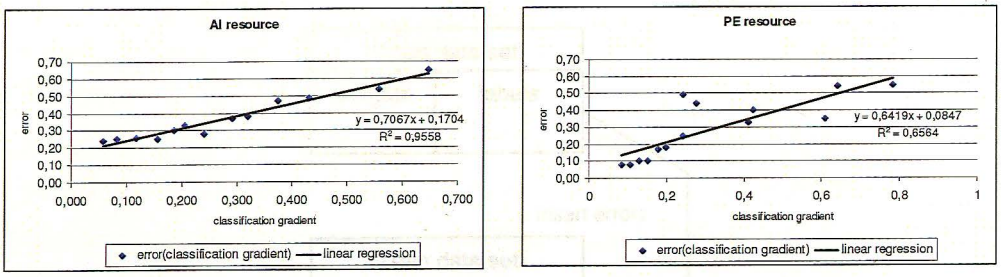


Figure 5.16 – Correlation between error and classification gradient

5.7 Robustness

The performance of the webTOPIC methodology is highly dependent on the pre-labeled examples that the user provides to characterize the concepts at the topic taxonomy. Unfortunately human classification of web pages is an erroneous task (Kobayashi et al, 2000; Macskassy et al, 1998), highly conditioned by exogenous, incontrollable, factors that contribute to high subjectivity and inconsistency. The robustness of the learning task assumes special importance under these circumstances; it is, then, important to understand the behavior of the learning module when fed with erroneous data sets.

5.7.1 Training and testing data sets

To evaluate robustness we have used the data sets that were previously used to estimate accuracy. Testing data sets are the same. The training data sets have the same examples but we have deliberately inserted errors at their labels (Figure 5.17). These errors emulate human classification errors.

For the supervised setting we have used 28 training data sets with 45 documents each, for the PE resource, and 22 training data sets with 42 documents each, for the AI resource. In reality there are ten samples of each of these data sets, the same that were previously used when computing accuracy. These data sets are grouped in four distinct groups, each of which with at most seven data sets. In each group, training data sets differ by the number of errors that were forced – 1, 3, 5, 7, 10, 15 and 20 errors were deliberately inserted on the training data sets that constitute each group. The four groups differ in the distribution of the errors that were inserted.

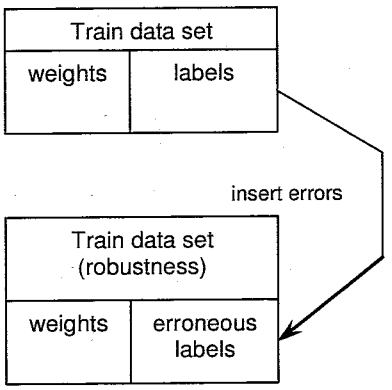


Figure 5.17 – Training data set preparation for robustness

Erroneous labels present two characteristics: the original label where the error was committed and the erroneous label that was assumed instead of the true one. Committed errors may be either random or systematic, concerning both these characteristics. Random errors are randomly committed at documents that belong to distinct categories while systematic errors are consistently committed on documents belonging to the same category. We will call *error profile* to any instance of these two characteristics: which label was the error committed on and to which erroneous label. The four training groups reflect distinct error profiles.

We will refer to error profiles as *ft.en*, where *f* – from "from" – stands for the distribution of the label where the error is committed, *t* – from "to" – stands for the distribution of the erroneous label and *en* refers to the number of inserted errors – *e* is a token referring to "error" and *n* states the number of errors that were inserted on the data set. Both *f* and *t* may be *r*, standing for random errors, or *sl*, standing for systematic error on label *l*. We will use this nomenclature to refer to each one of the training data sets used to evaluate the robustness on the supervised setting (Figure 5.18): for instance *s3r.e10* identifies the data sets where 10 errors were artificially inserted according to the profile *s3r* – systematic from label 3 to random (uniform) label – and *s3s5.e1* identifies the data set where one error was inserted according to profile *s3s5* – systematic from 3 to 5, meaning that one document originally labeled 3 has been assigned label 5.

This nomenclature was adapted to the semi-supervised setting. In this scenario we need to further specify the number of pre-labeled documents to consider within the training

data set. We have specified this characteristic of a given data set by including a new token, *np*, to the nomenclature – between the error profile and the number of forced errors – where p stands for the number of pre-labeled documents to be considered: for instance, *s3r.n9.e3* refers to a training data set with nine pre-labeled documents among which there are three errors generated according to profile *s3r*.

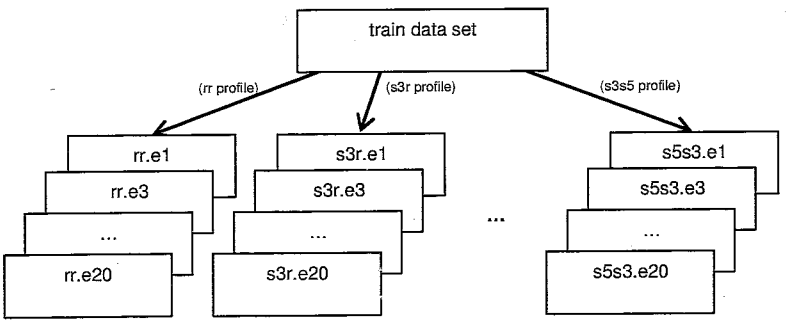


Figure 5.18 – Training data sets for supervised robustness

To evaluate semi-supervised robustness we have generated four groups of training data sets, corresponding to four error profiles: rr, s3r, s3s4 and s3s5. Each one of these groups has 55 and 61 distinct data sets, respectively for AI and PE resources, differing in the number of pre-labeled examples and in the number of inserted errors. The number of pre-labeled examples ranges from 6 to 39 and from 6 to 42, respectively for AI and PE resources, with a step of 3. The number of errors is 1, 3, 5, 7, 10, 15 or 20 inserted errors, just like at the supervised experiences. The number of errors inserted into a given data set does not ever exceed half the number of pre-labeled documents in the data set – for instance, there are only four data sets on the subgroup rr.n15, those with 1, 3, 5 and 7 inserted errors. Under these circumstances we have generated a total of 1720 data sets for AI and 2230 data sets for PE.

5.7.2 Experiences

Each of the training data sets previously generated for the supervised and semi-supervised settings were, in turn, used to learn classifiers for the respective topic taxonomies. Error rates were estimated as the mean of the error rates over the 10

samples for each particular configuration.

Supervised setting

The robustness of the SVM classifier may be analyzed from Figure 5.19 and Figure 5.20, which summarize the results obtained from the experiences conducted on the supervised setting; the graph has four series, each one representing an error profile.

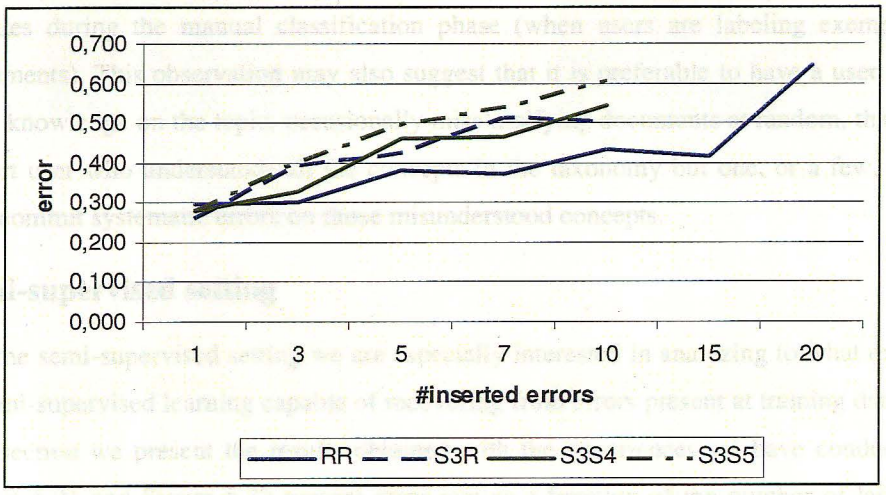


Figure 5.19 – Supervised error rate versus number of inserted errors for AI

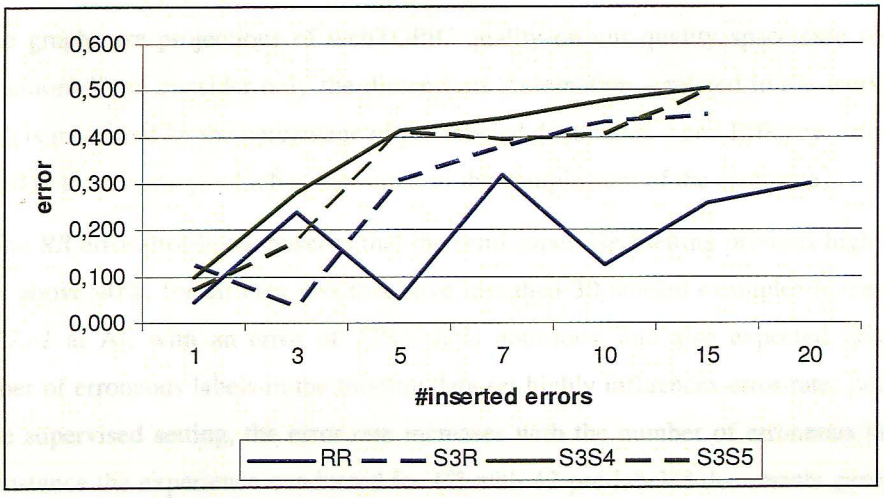


Figure 5.20 – Supervised error rate versus number of inserted errors for PE

We may observe that the error rate increases with the number of errors inserted at the training data set; this tendency is natural since erroneous labels misguide the classifier at the training stage, which will produce inadequate models. If we analyze the patterns evidenced at Random/Random error profile (*RR*), Systematic/Random profiles (*S3R*) and Systematic/Systematic profiles (*S3S4* and *S3S5*), we may observe that erroneous labels produce worth effects when they are systematic. This leads us to take special care on the detection and correction of Systematic/Systematic and Systematic/Random error profiles during the manual classification phase (when users are labeling exemplary documents). This observation may also suggest that it is preferable to have a user with little knowledge on the topic, occasionally misclassifying documents at random, than an expert user who understands all the concepts in the taxonomy but one, or a few, who will commit systematic errors on those misunderstood concepts.

Semi-supervised setting

For the semi-supervised setting we are especially interested in analyzing to what extent is semi-supervised learning capable of recovering from errors present at training data. In this section we present the results obtained with the experiences we have conducted. Figure 5.21 and Figure 5.22 present error rate as a function of the number of labeled documents and the number of erroneous labels that occur on the training data sets, for the different error profiles we have analyzed.

These graphs are projections of webTOPIC quality on our quality space (see section Evaluation) if we consider only the dimensions Automation – related to the workload, which is measured by the percentage of pre-labeled documents – and Efficacy – directly related to the accuracy, which is measured as the complement of the error rate.

For the *RR* error profile we observe that the semi-supervised setting presents high error rates, above 40%, for all data sets that have less than 30 labeled examples (except for *rr.n27.e1* at AI, with an error of 32%). It is notorious, and also expected, that the number of erroneous labels in the training data set highly influences error rate: just like in the supervised setting, the error rate increases with the number of erroneous labels. For instance the experience conducted for PE with 42 pre-labeled documents generates error rate estimates of 16% when one label is erroneous, 32% when there are seven errors and 53% if there are 20 errors.

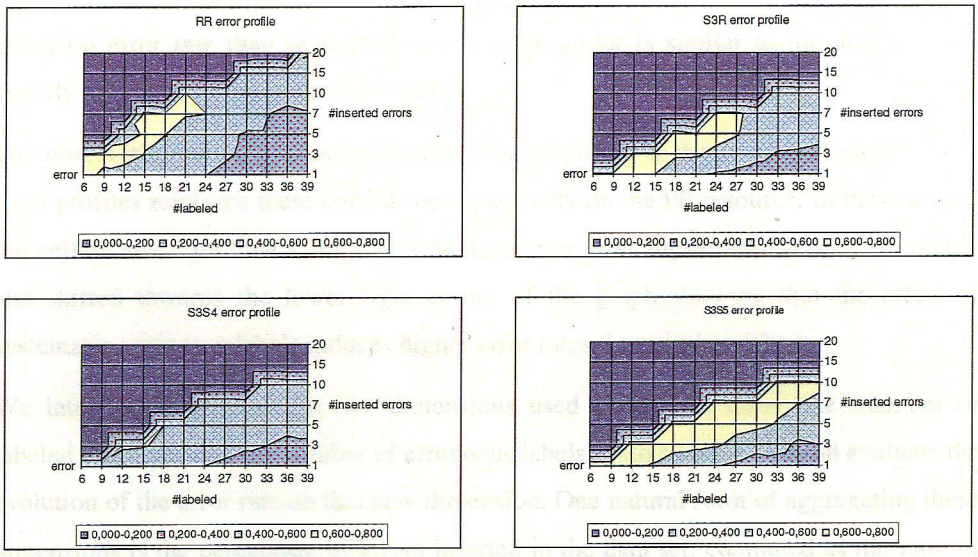


Figure 5.21 – Semi-supervised robustness for AI

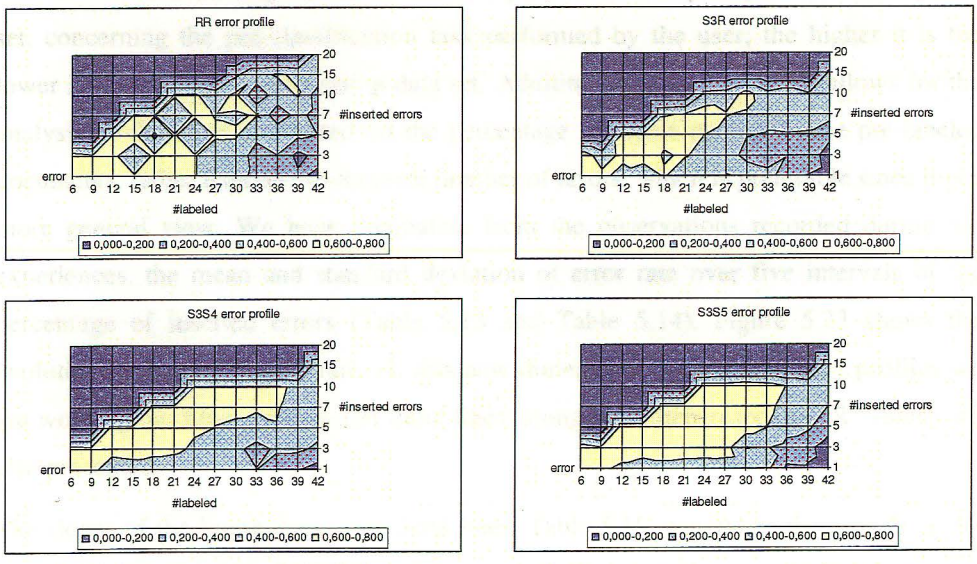


Figure 5.22 – Semi-supervised robustness for PE

More interesting evidence comes from the observation of the lower right corner of these graphs. *S3R* graphs show an area in dark and medium dark blue – corresponding to low error rates – which is much smaller than the one appearing in the *RR* graphs. This observation reflects that, on the semi-supervised setting, systematic errors have a worst

effect on error rate than random errors. This behavior is similar to the one we have already observed on the supervised setting.

The observation of the graphs representing the evolution of the error rates on *S3S4* and *S3S5* profiles reinforce these conclusions, especially on the PE resource. In these graphs the yellow and light blue spots – representing error rates higher than 40% – are bigger and shifted towards the lower right corner of the graph showing that the effect of systematic erroneous labels induces higher error rates than random errors.

We intend to aggregate the two dimensions used to analyze error rate (number of labeled examples and the number of erroneous labels) onto a single one and evaluate the evolution of the error rate on this new dimension. One natural form of aggregating these dimensions is the percentage of errors inserted in the data set, computed as the ratio of the number of inserted errors to the number of labeled examples.

This new dimension may be seen as the complement of the quality of the training data set, concerning the pre-classification task performed by the user; the higher it is the lower is the quality of the training data set. Additionally, this dimension allows for the analysis of the error rate based on the percentage of errors present in the pre-labeled documents, rather than on the absolute number of errors, which is preferable since it is a more general view. We have computed, from the observations recorded during the experiences, the mean and standard deviation of error rate over five intervals of the percentage of inserted errors (Table 5.13 and Table 5.14). Figure 5.23 shows the evolution of the mean error rate, on this new dimension, for the four error profiles we are working on. Linear regression trend lines, along with their respective R^2 values are also presented.

The slopes of the linear regression trend lines, Table 5.15, generalize the growth of the error rate committed by the semi-supervised classifier when the percentage of errors in the pre-labeled examples increases 1%.

These values also show that systematic errors produce worst effects on the error rate than random errors: while *RR* profiles have slopes of around 6% for AI and 3,5% for PE – meaning that an increment of 1% on the number of errors committed on the pre-labeled examples increases the error rate by 6% and 3,5%, respectively on AI and

PE resources – systematic error profiles have higher slopes. *S3S5* error profiles have the highest slopes of 7,7% and 7,2%, respectively for AI and PE resources.

		Inserted errors (% of labeled examples)				
Profile	**Error rate statistics**	**0-10%**	**10-20%**	**20-30%**	**30-40%**	**40-50%**
RR	**Mean**	0,358	0,437	0,538	0,568	0,596
	Standard deviation	0,095	0,095	0,079	0,058	0,038
S3R	**Mean**	0,395	0,538	0,608	0,579	---
	Standard deviation	0,110	0,074	0,052	---	---
S3S4	**Mean**	0,374	0,515	0,575	0,571	---
	Standard deviation	0,085	0,058	0,031	---	---
S3S5	**Mean**	0,410	0,615	0,663	0,650	---
	Standard deviation	0,115	0,065	0,027	---	---

Table 5.13 – Error rate by inserted errors statistics for AI

		Inserted errors (% of labeled examples)				
Profile	**Error rate statistics**	**0-10%**	**10-20%**	**20-30%**	**30-40%**	**40-50%**
RR	**Mean**	0,416	0,523	0,533	0,537	0,585
	Standard deviation	0,177	0,120	0,112	0,139	0,096
S3R	**Mean**	0,382	0,477	0,584	0,550	0,605
	Standard deviation	0,177	0,115	0,098	0,089	0,131
S3S4	**Mean**	0,418	0,569	0,626	0,634	0,646
	Standard deviation	0,137	0,066	0,030	0,025	0,034
S3S5	**Mean**	0,383	0,560	0,656	0,665	0,692
	Standard deviation	0,163	0,141	0,076	0,063	0,012

Table 5.14 – Error rate by inserted errors statistics for PE

It seems also interesting to compare the evolution of error rates between supervised and semi-supervised settings. Figure 5.24 present the evolution of the error rate for the experiences that are directly comparable. Experiences in the supervised setting were conducted over training data sets with 42 and 45 labeled examples so we compare them to the semi-supervised experiences that used training data sets with 42, 45 and 48 pre-labeled examples; these are the most approximate conditions between both settings.

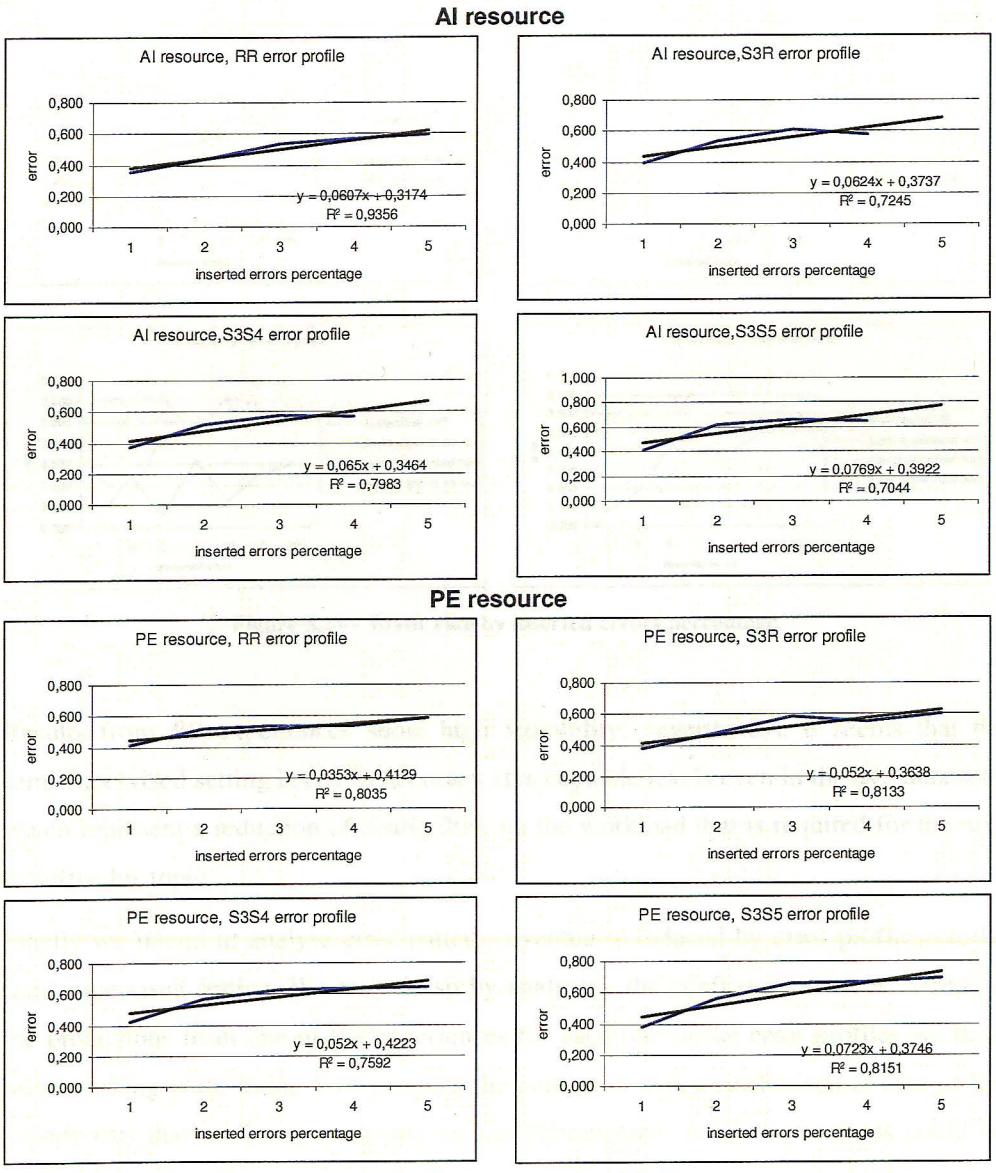


Figure 5.23 – Semi-supervised error rate by inserted errors percentage

	Trend line slope	
Error profile	AI resource	PE resource
Random/Random	6,1%	3,5%
Systematic3/Random	6,2%	5,2%
Systematic3/Systematic4	6,3%	5,2%
Systematic3/Systematic5	7,7%	7,2%

Table 5.15 – Slope of error rate by inserted errors percentage

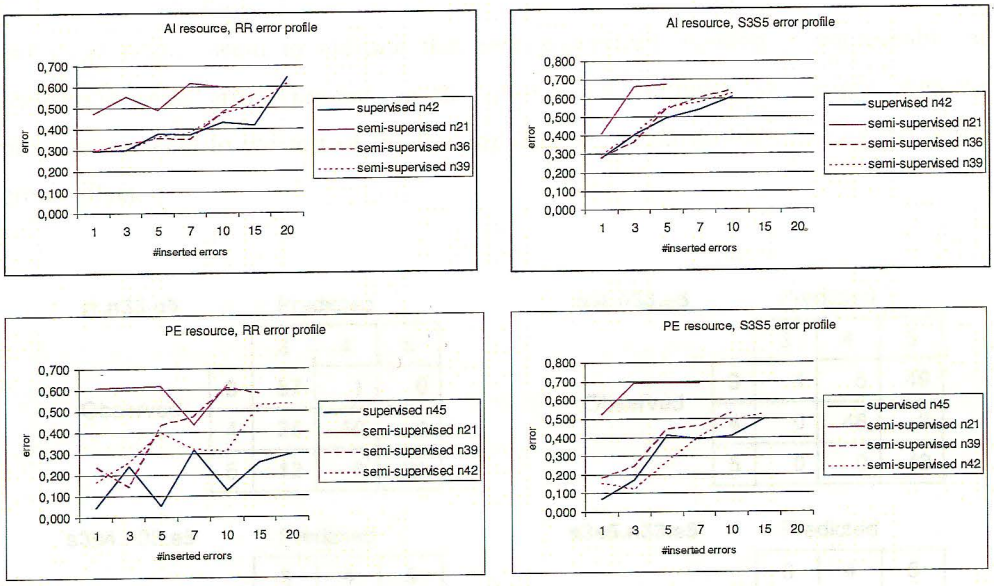


Figure 5.24 – Error rate by inserted errors percentage

Results from *RR* experiences show high variability; nevertheless, it seems that the semi-supervised setting keeps its accuracy at acceptable levels even in the *n42* data sets, which represent a reduction of nearly 20% on the workload that is required for the user to define his topic.

Finally we intend to analyse error patterns eventually induced by error profiles on the semi-supervised setting. We may do so by analysing the confusion matrices, based on the predictions from one of the experiences for each one of the error profiles we have been working with. Table 5.16 presents the confusion matrices obtained at four of the experiences that we have conducted on the PE resource. A similar analysis could be performed on the AI resource.

As expected, consistently committing errors on the same label, as occurs at systematic error profiles, produces biased classifiers towards favored labels. On the *RR* setting the classifier loses accuracy but does not favors any label against the others. Systematic error profiles present different behavior since classifiers learned on these profiles favor the most common label on the training data set: the classifier learned from this particular *S3S4* data set classifies almost all testing examples with label "4" while the *S3S5* profile generates a classifier that labels all testing examples with label "5". These

particular results seem to indicate that semi-supervised learning is not capable of recovering from errors present on the pre-labeled examples and might even be propagating them to the rest of the training data set, negatively reinforcing the initial errors effect.

rr.n33.e5

Observed \ Predicted	3	4	5
3	57	1	0
4	35	10	4
5	12	0	31

s3r.n33.e5

Observed \ Predicted	3	4	5
3	4	5	49
4	0	46	3
5	0	0	43

s3s4.n33.e5

Observed \ Predicted	3	4	5
3	0	58	0
4	0	49	0
5	0	36	7

s3s5.n33.e5

Observed \ Predicted	3	4	5
3	0	0	58
4	0	0	49
5	0	0	43

Table 5.16 – Confusion matrices for semi-supervised learning

Although this hypothesis requires additional experimental work to be confirmed, it already suggests the negative influence that errors on pre-labeled examples have on the classifier accuracy, thus reinforcing that special care must be taken on the manual pre-classification task. Since erroneous labels are probably associated to TF×IDF representations that are significantly distinct from the TF×IDF representations of the correctly assigned labels, it might be of great help to analyze the set of manually pre-classified documents and remove the labels assigned to outliers. The cross-classification mechanism, proposed on section Cross-classification, might also help improving the correctness on the pre-labeled exemplary documents.

5.8 Conclusive remarks

The learning task is critical since it largely influences overall system quality, as the user perceives it. The correct organization of the resource is directly and immediately perceived by the user and depends exclusively on the classifier accuracy. Moreover, this task is still more critical because training documents are very expensive to obtain and

high classification accuracy depends on large training data sets. Semi-supervised classification algorithms are particularly suitable under these circumstances.

The pre-processing phase transforms documents into an adequate representation for the learning task. Besides generating an appropriate model for each document, this phase is also responsible for reducing text feature space dimension: we have achieved significant reduction on both resources, whose final vocabularies have around 40% of corpus total terms. We have also observed that the length of resource lexicon grows proportionally to the number of documents in the resource. This may be explored to predict the computational effort that will be necessary to process a given resource and its periodic updates.

Semi-supervised learning presents error rates that are comparable to the ones we have obtained at the supervised learning but for lower workloads. This is especially the case if pre-labeled documents are stratified according to label's distribution in the resource.

Concerning robustness, experimental evidence seems to indicate that supervised learning is slightly more accurate than semi-supervised learning. Systematic errors produce worst effects on error rate than random errors. While random errors introduce white noise that equally affects all labels, systematic errors introduce erroneous patterns that explicitly misguide the classifier.

6 Conclusions

6.1 Thesis contributions

This thesis proposes a methodology to automatically retrieve a document collection on some specific topic, as defined by the user, and to organize and keep it up-to-date over time. The methodology keeps track of the user's present needs, following eventual drifts in his or hers interests without explicitly requiring any additional user effort. Drift in user's interests is inferred from the actions log that the system builds in the background, based on user interaction – through actions such as printing, saving, viewing or changing the label of a document – while exploring the resource.

The system keeps several views on the topic – each view is defined as a resource, which is a document collection – each one representing the topic at a specific period of time. The methodology operates on the core Topic/User binomial. The user defines a topic with a set of keywords – used to build queries to submit to public search engines – a taxonomy – representing the desired ontological organization for the resource – and a set of exemplary documents, previously classified by the user on his or hers specific topic taxonomy.

Document collections, or resources, are retrieved through a continuous meta-search process, where queries are dynamically scheduled by an agent that analyses the trade-off between resource freshness and percentage of duplicates in the result sets obtained in response to a query, in order to keep the resource up-to-date while reducing the number of submitted queries.

Exploring the resource is enhanced with the inclusion of a user interface that enables the exploration of large document collections. The main concern is to allow for the user to be able to analyze a single document while keeping a global view of the entire collection.

System performance is highly subject to errors committed while classifying exemplary documents, which will be the base for the learning task. To prevent errors in this phase, we propose two methods: the cross-classification process and the elimination of the

outliers from each cluster of pre-classified documents in each category within the training data set, before submitting these examples to the learning task. We have categorized error profiles based on their distribution as to the label where the error was committed and to the erroneously assumed label. Systematic errors on these pre-labeled examples have particularly negative effects on the performance of the system.

We also propose a framework to evaluate system performance and to keep it in line with user interests. This framework is supported on a quality space defined on the Automation, Efficacy and Efficiency dimensions. Each dimension in this quality space aggregates a set of metrics, relevant to the system's global quality. Automation is computed from user workload, essentially required in the initial phase, while labeling exemplary documents; Efficacy is computed from precision and accuracy and Efficiency, is computed from recall, freshness and novelty. The quality of the system is permanently monitored; the system periodically measures and stores the values of its quality parameters. Using this quality log the system permanently analyzes operating performance and may prevent quality degradation. With this adaptive quality control framework the system may automatically monitor its own performance, in a continuous manner, and trigger pre-defined corrective procedures when prescribed. Besides these corrective and reactive measures the system may also trigger preventive procedures when a quality forecast, based on measured quality records, falls below some threshold.

6.2 Directions for future research

The application of webTOPIC on some production environment, with real users, as we intend to do at the APPIA site, will certainly reveal many of the methodology strengths and weaknesses providing many opportunities to improve it. For the moment the methodology has only been applied at laboratory, in a controlled environment, but a few improvements seem appropriate:

- The feature space can be further reduced if we eliminate the terms that are least discriminatory regarding topic taxonomy; this may be accomplished, at the expense of some additional computational effort if we apply Latent Semantic Indexing (Deerwester et al, 1988).

- Identification of the documents language is somehow guaranteed by some of the information sources – search engines – used in the meta-search process. Unfortunately, we have observed that the public search engines that we have used in our prototype are not very reliable concerning the identification of web pages language. We believe this could be improved using some specific technique to identify document's language (Martins et al, 2002). Identification of document's language should occur on the initial phase of the pre-processing stage; once the language is identified we may use the appropriate stop-words lists and apply appropriate stemming algorithms, generating document representations fitted for each language. Each language requires its own feature space, its own lexicon. With these adaptations our prototype would be able to manage multi-lingual resources, as long as classifiers are independently trained for each language – which requires a minimum number of pre-labeled documents on each category for all the languages that are to be considered.
- Take advantage of the taxonomy's hierarchical structure, by applying hierarchical classification techniques (Dumais et al, 2000), might improve classification. Our prototype uses flat classifiers that ignore inheritance properties and hierarchical relationships between the classes that constitute the topic taxonomy.
- We rely exclusively on meta-search to retrieve documents from the Web; it will probably improve recall if we explore some kind of crawling. Simply augmenting the primary data set with each document's out-links has proven to increase the set of relevant documents. We believe that a more sophisticated technique based on focused crawling (Chakrabarti et al, 1999a) might significantly improve system performance.
- The application of information extraction techniques may also add very interesting capabilities to webTOPIC. These techniques may be explored to generate document summaries, or to summarize the content of sets of strongly related documents, which may be very valuable at the presentation layer. At a second phase, we may explore information extraction techniques to automatically build a report on the current state of the art of some topic, specifically structured according to the organization privileged by the user.

- When defining a topic the user may choose any arbitrary taxonomy; webTOPIC does not impose any kind of restriction or rule on this subject. Thus, there is no guarantee that web documents can be automatically labeled in this taxonomy with a text classifier – that just looks for the document content text – or any other classifier that takes into account anyone of the document aspects that are being considered in the webTOPIC document model. We may imagine a user who is interested in the topic Computers, for instance, and wishes to organize his or hers resource in two categories: documents that have pictures and those who do not. It is impossible, or at least much harder, to classify documents on these categories if the only features we have are text features, than it would be if we had one more attribute in the document model that counts the number of images present in the document. Depending on the kind of ontological organization that the user is interested in the classification task may be based on different sets of features. We intend to develop a web document model, consisting of several independent sets of features, covering most of the aspects that the user may use to organize his or hers resource. The classification task would then be partitioned in two steps: the first step is to learn the most adequate set of features for the topic at hand and the second step is to use that set of features to learn the taxonomy. This classification process may guarantee a great flexibility on the topic definition.

7 Bibliography

Aas, K., Eikvil, L. (1999), *Text Categorization: A Survey*, Norwegian Computing Center.

Aggarwal, C.C. (2004), "On Leveraging User access Patterns for Topic Specific Crawling", *Data mining and Knowledge Discovery*, 9, pp 123-145, Kluwer Academic Publishers.

Aggarwal, C.C., Al-Garawi, F., Yu, P. (2001), "Intelligent crawling on the World Wide Web with arbitrary predicates", *Proceedings of the 10th World Wide Web Conference.*

Agrawal, R., Gupta, A., Sarawagi, S. (1997), "Modeling multidimensional databases", *Proceedings of the 13th International Conference on Data Engineering.*

American Heritage Dictionary of English Language (2000).

Arasu, A., Cho, J., Garcia-Molina, H., Paepcke, A., Raghavan, S. (2001), "Searching the Web", *ACM Transactions on Internet Technology* 1(1), pp 2-43.

Aslam, J.A., Montague, M. (2001), "Models for Metasearch", *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval.*

Baeza-Yates, R., Neto, R. (1999), *Modern Information Retrieval*, Addison Wesley.

Baeza-Yates, R., Saint-Jean, F., Castillo, C. (2002), "Web Structure, Dynamics and Page Quality", *Proceedings of SPIRE 2002,* pp 117-130.

Baeza-Yates, R. (2003), "Information Retrieval in the Web: beyond current search engines", *Elsevier International Journal of Approximate Reasoning*, 34, pp 97-104.

Baldi, P., Frasconi, P., Smyth, P. (2003), *Modeling the Internet and the Web. Probabilistic Methods and Algorithms*, Wiley.

Bennet, K.P., Demiriz, A. (1998), "Semi-Supervised Support Vector Machines",

Proceeding of Neural Information Processing Systems.

Bharat, K., Broder, A. (1998), "A technique for measuring the relative size and overlap of public Web search engines", *Proceedings of the 7th World Wide Web Conference.*

Blum, A., Mitchell, T. (1998), "Combining labelled and unlabelled data with Co-training", *Proceedings of the 11th Annual Conference on Computational Learning Theory*, pp 92-100.

Borges, J.L.C.M. (2000), *A Data Mining Model to Capture User Web Navigation Patterns*, PhD dissertation submitted to the University of London.

Brin, S., Page, L. (1998), "The anatomy of a large-scale hypertextual web search engine", *Proceedings of the 7th World Wide Web Conference*, pp 107-117.

Broder, A., Kumar, R., Maghoul, F., Raghavan, P., Rajagopalan, S., Stata, R., Tomkins, A., Wiener, J. (2000), "Graph structure in the web: Experiments and Models", *Proceedings of the 9th World Wide Web Conference*, pp 309-320.

Bueno, D., David, A.A. (2001), "METIORE: A Personalized Information Retrieval System", *Proceedings of the 8th International Conference on User Modeling,* Springer-Verlag.

Bruza, P., McArthur, R., Dennis, S. (2000), *Interactive Internet search: keyword, directory and query reformulation mechanisms compared*, Research and Development in Information Retrieval.

Callison-Burch, C., (2003), *Active Learning for Statistical Machine Translation*, PhD proposal, School of Informatics, Institute for Communicating and Collaborative Systems, University of Edinburgh.

Carey, M., Kriwaczek, F., Ruger, S.M. (2000), "A Visualization Interface for Document Searching and Browsing", *Proceedings of the NPIVM 2000.*

Castillo, C., Marin, M., Rodriguez, A., Baeza-Yates, R. (2004), "Scheduling Algorithms for Web Crawling", *Proceedings of the Latin American Web Conference 2004.*

Chakrabarti, S., Dom, B., Indyk, P. (1998a), "Enhanced hypertext categorization using hyperlinks", *Proceedings of ACM SIGMOD International Conference on Management of data*, pp 307-318.

Chakrabarti, S., Dom, B., Agrawal, R., Raghavan, P. (1998b), "Scalable feature selection, classification and signature generation for organizing large text databases into hierarchical topic taxonomies", *The VLDB Journal*, 7, pp 163-178.

Chakrabarti, S., Dom, B., Raghavan, P., Rajagopalan, S., Gibson, D., Kleinberg, J. (1998c), "Automatic Resource Compilation by Analyzing Hyperlink Structure and Associated Text", *Proceedings of the 7th International World Wide Web Conference.*

Chakrabarti, S., Byron, E., Kumar, S., Raghavan, P., Rajagopalan, S., Tomkins, A., Gibson, D., Kleinberg, J. (1999), "Mining Web Link Structure", *IEEE Computer*, 32(8), pp 60-67.

Chakrabarti, S., Berg, M., Dom, B. (1999a), "Focused crawling: a new approach to topic-specific resource discovery", *Proceedings of the 8th World Wide Web Conference.*

Chakrabarti, S. (1999b), "Automatic Web Resource Discovery", *ACM Computing Surveys*, 31(4).

Chakrabarti, S. (2000), "Data mining for hypertext: A tutorial survey", *ACM SIGKDD Explorations.*

Chakrabarti, S. (2003), *Mining the web, Discovering Knowledge from Hypertext Data*, Morgan Kaufmann Publishers.

Chaves, M.S. (2003), "Um Estudo e Apreciação sobre Algoritmos de Stemming para a Língua Portuguesa", *IX Jornadas Iberoamericanas de Informática.*

Chen, C.C., Chen, M.C., Sun, Y. (2001), "PVA: A Self-Adaptive Personal View Agent System", *Proceedings of the ACM SIGKDD 2001 Conference.*

Chewar, C.M., Krowne, A., O'Laughlen, M. (2001), *User Object Collections: Visualization Concepts by collection-Insight Need*, CITIDEL project.

Cho, J., Garcia-Molina, H. (2000a), "Synchronizing a database to improve freshness", *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data.*

Cho, J., Garcia-Molina, H. (2000b), "The evolution of the Web and implications of an incremental crawler", *Proceedings of the 26th International Conference on Very Large Data Bases*, pp 200-209.

Cho, J., Garcia-Molina, H. (2000c), *Estimating Frequency of Change*, Technical report, Stanford University.

Cohn, D., Gharamani, Z., Jordan, M. (1996), "Active learning with statistical models", *Journal of Artificial Intelligence Research*, 4, pp 129-145.

Cooley, R., Mobasher, B., Srivastava, J.(1997), "Web Mining: Information and Pattern Discovery on the World Wide Web", *Proceedings of the 9th IEEE International conference on tools with Artificial Intelligence*, pp 558-567.

Cugini, J., Piatko, C., Laskowski, S. (1996), Interactive 3D Visualization for Document Retrieval, *Proceedings of the ACM Conference on Information and Knowledge Management.*

Deerwester, S., Dumais, S.T., Furnas, G.W., Landauer, T.K., Harshman, R. (1988), Indexing by Latent Semantic Analysis, *Proceedings of the 1988 ACM SIGIR Conference.*

De Bra, P., Houben, G., Kornatzky, Y., Post, R.D.J. (1994a), "Information retrieval in distributed hypertexts", *Proceedings of the 4th RIAO Computer Assisted Information Retrieval Conference*, pp 481-491.

De Bra, P., Post, R.D.J. (1994b), "Information retrieval in the World Wide Web: making client-based searching feasible", *Computer Networks*, 27(2), pp 183-192.

Donato, D., Laura, L., Millozi, S. (2000), "A beginner's guide to the Webgraph: Properties, Models and Algorithms", *Proceedings of the 41st FOCS*, pp.57-65.

Dumais, S., Chen, H. (2000), "Hierarchical Classification of Web Content",

Proceedings of the 23rd ACM SIGIR Conference, pp 256-263.

Etzioni, O. (1996), "The World-Wide-Web: quagmire or gold mine?", *Communications of the ACM*, Vol. 39, No. 11, pp 65-68.

Fang, Y. C., Parthasarathy, S., Schwartz, F. (2001), "Using Clustering to Boost Text Classification", *Proceedings of the IEEE ICDM-2001, The 2001 IEEE International Conference on Data Mining.*

Gammerman, A., Vovk, V., Vapnik, V. (1998), "Learning by Transduction", *Conference on Uncertainty in Artificial Intelligence*, pp 148-155.

Ghani, R., Jones, R., Mladenic, D., Nigam, K., Slattery, S. (2000), "Data Mining on Symbolic Knowledge Extracted from the Web", *Workshop on Text Mining at the 6th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining.*

Ghani, R. (2001), "Combining labelled and unlabelled data for text classification with a large number of categories", *Proceedings of the 2001 IEEE International Conference on Data Mining.*

Glover, E.J., Flake, G.W., Lawrence, S., Birmingham, P., Kruger, A., Giles, C.L., Pennock, D.M. (2001), "Improving Category Specific Web Search by Learning Query Modifications", *Symposium on Applications and the Internet, IEEE Computer Society*, pp 23-31.

Gravano, L, Chang, C.K., Garcia-Molina, H., Paepcke, A. (1997), "STARTS: Stanford Proposal for Internet Meta-Searching", *Proceedings of the 1997 ACM SIGMOD International Conference on Management of Data.*

Gravano, L., Papakonstantinou, Y. (1998), "Mediating and Metasearching on the Internet", *Bulletin of the IEEE Computer Society Technical Committee on Data Engineering.*

Halkidi, M., Nguyen, B., Varlamis, I., Vazirgiannis, M. (2003), "Thesus: Organizing Web document collections based on link semantics", *The VLDB Journal*, 12, pp 320-332.

Hersovici, M., Jacovi, M., Maarek, Y.S., Pelleg, D., Shtalaim, M., Ur, S. (1998), "The

Shark-search algorithm. An application: tailored web site mapping", *Computer Networks* 30(1-7), pp 317-326.

Howe, A.E., Dreilinger, D. (1997), "SavvySearch: A Meta-Search Engine that Learns which Search Engines to Query", *AI Magazine*, 18(2), pp 19-25.

Hu, W., (2002), "World Wide Web Search Technologies", *Architectural Issues of Web-Enables Electronis Business*, edited by Shi Nansi for Idea Group Publishing.

Jansen, B.J., Spink, a., Saracevic, T. (2000), "Real life, real users, and real needs: A study and analysis of user queries on the web", *Information Processing and Management*, 36(2), pp 207-227.

Joachims, T. (1997), "A probabilistic analysis of the rocchio algorithm with TFIDF for text categorization", *Proceedings of the 1997 International Conference on Machine Learning*.

Joachims, T. (1998), *Text Categorization with Support Vector Machines: Learning with Many Relevant Features*, Research Report of the unit no. VIII(AI), Computer Science Department of the University of Dortmund.

Joachims, T. (1999), "Transductive Inference for Text Classification using Support Vector Machines", *Proceedings of the 16th International Conference on Machine Learning*, pp 200-209.

Joachims, T. (1999), *Making large-Scale SVM Learning Practical. Advances in Kernel Methods - Support Vector Learning*, edited by Schölkopf, B., Burges, C., Smola, A., MIT-Press, *http://www-ai.cs.uni-dortmund.de/DOKUMENTE/joachims_99a.pdf.*

Jones, R., McCallum, A., Nigam, K., Riloff, E. (1999), "Bootstrapping for Text Learning Tasks", *IJCAI-99 Workshop on Text Mining: Foundation, Techniques and Applications*, pp. 52-63.

Kahle, B. (1997), "Preserving the internet", *Scientific American*, 276(3), pp 82-83.

Kleinberg, J.M., Kumar, R., Raghovan, P., Rajagopalan, S., Tomkins (1999), "The Web as a graph: measurements, models and methods", *Proceedings of the 5th International*

Conference on Computing and Combinatorics.

Kleinberg, J. (1998), "Authoritative sources in a hyperlinked environment", *Proceedings of the 9th ACM-SIAM Symposium on Discrete Algorithms*, pp 668-677.

Kobayashi, M., Takeda, K. (2000), "Information Retrieval on the Web", *ACM Computing Surveys*, 32(2), pp 144-173.

Koller, D., Sahami, M. (1996), "Toward Optimal Feature Selection", *Proceedings of the 13th International Conference on Machine Learning*, pp. 284-292, Morgan Kaufmann.

Kosala, R., Blockeel, H. (2000), "Web Mining Research: A Survey", *SIGKDD Explorations*, Vol. 2, No. 1, pp 1-13.

Kumar, R., Raghavan, P., Rajagopalan, S., Sivakumar, D., Tomkins, A., Upfal, E. (2000), "The Web as a graph", *Proceedings of the 19th ACM SIGACT-SIGMOD-AIGART Symp. Principles of Database Systems.*

Lawrence, S., Giles, C.L. (1998), "Searching the World Wide Web", *Science*, Vol. 280, pp 98-100.

Lawrence, S., Bollacker, K., Giles, C.L. (1999a), "Indexing and Retrieval of Scientific Literature", *Proceedings of the 8th International Conference on Information and Knowledge Management*, pp 139-146.

Lawrence, S., Giles, C.L. (1999b), "Accessibility of information on the web", *Nature*, 400, pp 107-109.

Lewis, D., Ringuette, M. (1994), "Comparison of two learning algorithms for text categorization", *Proceedings of the 3rd Annual Symposium on Document Analysis and Information Retrieval.*

Li, X., Liu, B. (2003), "Learning to classify text with positive and unlabelled data", *Proceeding of IJCAI – 2003.*

Lieberman, H. (1995), "Letizia: an Agent That Assists Web Browsing", *Proceedings of the International Joint Conference on AI.*

Lim, L., Wang, M., Padmanabhan, S., Vitter, J.S., Agarwal, R. (2001), "Characterizing Web Document Change", *Lecture notes in Computer Science.*

Liu, B., Chin, C.W., Ng, H. T. (2003a), "Mining Topic-Specific Concepts and Definitions on the Web", *Proceedings of the WWW2003 Conference.*

Liu, B., Dai, Y., Li, X., Lee, W., Yu, P. (2003b), "Building Text Classifiers Using Positive and Unlabelled Examples", *Proceedings of the 3rd IEEE International Conference on Data Mining.*

Liu, B., Lee, W., Yu, P., Li, X. (2003c), "Partially supervised classification of text documents", *Proceeding of ICML 2003.*

Lu, S., Dong, M., Fotouhi, F. (2002), "The semantic web: opportunities and challenges for next generation web applications", *Information Research*, 7 (4).

Macskassy, S.A., Banerjee, A., Dovison, B.D., Hirsh, H. (1998), "Human Performance on Clustering Web Pages: a Preliminary Study", *Proceedings of the 4th International Conference on Knowledge Discovery and Data Mining.*

Manmatha, R., Rath, T., Feng, F. (2001), "Modeling Scored Distributions for Combining the Outputs of Search Engines", *Proceedings of the 24th ACM SIGIR Conference.*

Martins, B., Silva, M.J. (2002), "Language Identification in Web Pages", *Document Engeneering Track of the 20th ACM Symposium on Applied Computing (Unpublished).*

Mitra, M., Singhal, A., Buckley, C. (1998), "Improving automatic query expansion", *Proceedings of the 21st ACM SIGIR Conference.*

Mitchell, S., Mooney, M., Mason, J., Paynter, G.W., Ruscheinski, J., Kedzierski, A., Humphreys, K. (2003), "iVia Open Source Virtual Library System", *D-Lib Magazine*, Vol. 9, No. 1.

Mladenic, D. (1999), *Personal WebWatcher: design and implementation*, Technical Report IJS-DP-7472, SI.

Nicola, C., Gaussier, E., Goutte, C., Renders, J. M. (2003), "Word-Sequence Kernels",

Journal of Machine Learning Research, Nº 3, pp 1053-1082.

Nigam, K., McCallum, A.K., Thrun, S., Mitchell, T.M. (2000), "Text classification from labelled and unlabelled documents using EM", *Machine Learning*, 39, pp 103-134.

Ng, K.B., Kantor, P.P. Kantor (1998), "An Investigation of the Preconditions for Effective Data Fusion in Information Retrieval: A Pilot Study", *Proceedings of the 61st Annual Meeting of the American Society for Information Science.*

O'Neill, E.T., Lavoie, B.F., Bennett, R. (2003), "Trends in the evolution of the public web", *D-Lib magazine*, Vol. 9, No. 4.

Olsen, K.A., Korfhage, R.R., Sochats, K.M., Spring, M.B., Williams, J.G. (1992), "Visualization of a Document Collection: The VIBE System.", *Information Processing & Management*, Vol. 29, No. 1, pp 69-81.

Orengo, V., Huyck, C. (2001), "A Stemming Algorithm for the Portuguese Language", *Proceedings of the 8th SPIRE.*

Pooley, R., Stevens, P. (1998), *Using UML. Software Engineering with Objects and Components*, Addison Wesley, Object Technology Series.

Porter, M.F. (1980), "An algorithm for suffix stripping", *Program*, 14, No. 3, pp 130-137.

Quek, C.Y. (1998), *Classification of World Wide Web Documents*, Senior Honors Thesis School of Computer Science Carnegie Mellon University.

Seeger, M. (2001), *Learning with labelled and unlabelled data*, PhD report, Institute for Adaptive and Neural Computation, University of Edinburgh.

Selberg, E., Etzioni, O. 1995, "The MetaCrawler Architecture for Resource Aggregation on the Web", *IEEE Expert*, 12(1), pp 8-14.

Sharma, S. (1996), *Applied Multivariate Techniques*, John Wiley & Sons, Inc.

Silva, M.J., Martins, B. (2002), "Web Information Retrieval with Result set Clustering", *Natural Language and Text Retrieval Workshop at EPIA'03.*

Spink, A., Wolfram, D., Jansen, B.J., Saravecic, T. (2001), "Searching of the web: the public and their queries", *Journal of American Society of Information Sciences and Technology* 52(3), pp 226-234.

Tibshirani, R., Hinton, G. (1995), *Coaching variables for regression and classification*, Technical report, University of Toronto.

Viji, S. (2002), *Term and Document Correlation and Visualization for a set of Documents*, Technical report, Stanford University.

Wang, J., Lochovsky, F. (2003), "Web Search Engines", *Journal of ACM Computing Survey* (accepted for revision).

Witten, I.A., Frank, E. (2000), *Data Mining. Practical Machine Learning Tools and Techniques with Java Implementations*, Morgan Kaufmann Publishers.

Yang, Y., Chute, C. G. (1994), "An example-based mapping method for text categorization and retrieval", *ACM Transaction on Information Systems,* pp. 253-277.

Yang, Y. (1999), "An Evaluation of Statistical Approaches to Text Categorization", *Journal of Information Retrieval*, vol. 1, nos. 1/2, pp 67-88.

Yang, Y., Pederson, J. (1997), "A Comparative Study of Feature Selection in Text Categorization", *International Conference on Machine Learning.*

Yang, Y., Slattery, S., Ghani, R. (2002), *A Study of Approaches to Hypertext Categorization*, Kluwer Academic Publishers, pp. 1-25.

Zamir, O., Etzioni, O. (1999), "Grouper: A Dynamic clustering Interface to Web Search Results", *Proceedings of the 1999 World Wide Web Conference.*

Zhang, D., Dong, Y. (2000), "An efficient algorithm to rank web resources", *Computer Networks*, 33(1-6), pp 449-455.

Appendix I – Estimating quality coordinates

Computing system coordinates on the quality space we have defined requires estimating a few parameters: recall, number of documents that are correctly classified and number of documents that are up-to-date.

I.1 Estimating recall

Estimating recall (U) may be rather complicated; yet, in our specific problem, it is not required to have a very precise estimate of U, since we are interested in relative, and not absolute, variations of q_S; it seems sufficient to have a rough estimate of U, as long as it is guaranteed that it lacks by excess.

In these circumstances it seems reasonable to estimate U (the number of documents in the Web that are relevant to query q) based on:

W = estimate of the total number of documents in the Web

S_i = estimate of the size of the document collection indexed by search engine i

D_{iq} = number of documents returned by search engine i in response to query q

T_{iq} = number of positive documents returned by search engine i in response to query q, in the top 100 of the rank (search engines usually limit the size of the returned result sets, it seems acceptable to expect that any search engine will return at least the 100 most relevant)

R_{iq} = number of relevant documents returned by search engine i in response to query q.

Since all these variables refer to query q, this index may be omitted.

We further admit that:

1. the document that are relevant to a given query q are uniformly distributed over the Web;
2. search engines used in the estimation of U are unbiased and have index databases that follow approximately the same distribution that is verified in the

Web as a whole;

3. document collections of all used search engines have intersections of about 2%, and the documents are also uniformly distributed in these intersections;
4. the probability of a document being relevant at the i^{th} set of 100 documents equals p^i, where p is the probability of a document being relevant in the first 100 returned documents. This presupposition seems to inflate the number of relevant documents returned by the search engine, which is not impeditive once it is acceptable, for our purposes, to estimate U by excess. (Manmatha et al, 2001) derive the distribution of the relevance of a given document as a function of the score attributed by the search engine; however, to use these results it is required that the search engine also returns the score of each document, which is not always true, or else to approximate the score by the rank of the document, which may also be a very noisy approach. The method we propose gives a rough estimate but is simple and seems adequate for the scenario at hand.

Now we can estimate U as:

$$U = \left[\frac{1}{n}(1-0{,}02)\sum_{i=1}^{n}\frac{R_i}{S_i}\right]\times W \qquad [I.1]$$

where

$$R_i = 100\sum_{k=1}^{\mathrm{int}\left(\frac{D_i}{100}\right)+1}\left(\frac{T_i}{100}\right)^k \qquad [I.2]$$

with *int(x)* representing the larger integer contained in x.

I.2 Estimating the number of correctly classified documents

The number of correctly classified documents (C) may be estimated from user interaction with the methodology. We may assume that any document that the user views or edits and keeps its original category is correctly classified. Documents that are explicitly re-labelled by the user, or automatic classifiers, and get a different label from

their previous one are assumed to be misclassified. From these figures – number of correctly classified documents N_c and misclassified documents N_m – we may compute the percentage of correctly classified documents and estimate C as:

$$C = \left(\frac{N_c}{(N_c + N_m)} \right) \times N \qquad [I.3]$$

This estimate is continuously updated as users interact with resources through the presentation layer.

I.3 Estimating the number of up-to-date documents

The number of documents that are up-to-date (Y) may be computed after each meta-search cycle. Similarly to the number of correctly classified documents, we may estimate the number of up-to-date documents from the total number of documents in the resource (N) and the percentage of up-to-date documents. This percentage may be computed from the result set of each meta-search cycle as the number of returned documents that have not changed since last search cycle – same URL returns same content – and the number of documents that have changed – same URL returns different content. This way Y is estimated as:

$$Y = \left(\frac{N_{\approx}}{(N_{\approx} + N_{\neq})} \right) \times N \qquad [I.4]$$

where $N_{\approx}$ stands for the number of URLs that have returned the same content and $N_{\neq}$ stands for the number of URLs that have returned different content.

FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO
BIBLIOTECA
0000093563