Machine Learning in Automated Text Categorisation

Fabrizio Sebastiani
Consiglio Nazionale delle Ricerche, Italy

The automated categorisation (or classification) of texts into topical categories has a long history, dating back at least to the early '60s. Until the late '80s, the most effective approach to the problem seemed to be that of manually building automatic classifiers by means of *knowledge-engineering* techniques, i.e. manually defining a set of rules encoding expert knowledge on how to classify documents under a given set of categories. In the '90s, with the booming production and availability of on-line documents, automated text categorisation has witnessed an increased and renewed interest, prompted by which the *machine learning* paradigm to automatic classifier construction has emerged and definitely superseded the knowledge-engineering approach. Within the machine learning paradigm, a general inductive process (called the *learner*) automatically builds a classifier (also called the *rule*, or the *hypothesis*) by "learning", from a set of previously classified documents, the characteristics of one or more categories. The advantages of this approach are a very good effectiveness, a considerable savings in terms of expert manpower, and domain independence. In this survey we look at the main approaches that have been taken towards automatic text categorisation within the general machine learning paradigm. Issues pertaining to document indexing, classifier construction, and classifier evaluation, will be discussed in detail. A final section will be devoted to the techniques that have specifically been devised for an emerging application such as the automatic classification of Web pages into "YAHOO!-like" hierarchically structured sets of categories.

Categories and Subject Descriptors: H.3.1 [**Information storage and retrieval**]: Content analysis and indexing—*Indexing methods*; H.3.3 [**Information storage and retrieval**]: Information search and retrieval—*Information filtering*; H.3.3 [**Information storage and retrieval**]: Systems and software—*Performance evaluation (efficiency and effectiveness)*; I.2.3 [**Artificial Intelligence**]: Learning—*Induction*

General Terms: Algorithms, Experimentation, Theory

Additional Key Words and Phrases: Machine learning, text categorisation, text classification

1. INTRODUCTION

In the last ten years automated content-based document management tasks have gained a prominent status in the information systems field, largely due to the widespread and continuously increasing availability of documents in digital form, and the consequential need on the part of the users to access them in flexible ways. *Text categorisation* (TC – also known as *text classification*, or *topic spotting*), the activity of labelling natural language texts with thematic categories from a predefined set, is one such task. TC has a long history, dating back at least to the early '60s, but it was not until the early '90s that, largely due to an increased applicative interest and to the availability of more powerful hardware, the discipline became a major field of investigation in the information systems discipline. Nowadays,

Address: Istituto di Elaborazione dell'Informazione, Consiglio Nazionale delle Ricerche, Via S. Maria, 46 - 56126 Pisa (Italy). E-mail: fabrizio@iei.pi.cnr.it

TC is used in many applicative contexts, ranging from automatic document indexing based on controlled vocabulary, to document filtering, automated metadata generation, word sense disambiguation, population of "YAHOO!-like" hierarchical catalogues of Web resources, and in general to any application requiring document organisation or selective dispatching. Although commercial TC systems are not yet as widespread as e.g. commercial information retrieval (IR) systems, experimental TC systems have long gone past the prototype stage (see e.g. [Lewis et al. 1999]), thanks to a by now consolidated body of literature rooted in a solid experimental methodology. It is this body of literature that the present paper attempts to review.

Until the late '80s, the most effective approach to TC seemed to be that of manually building automatic TC systems by means of knowledge-engineering techniques, i.e. manually defining a set of logical rules that encode expert knowledge on how to classify documents under the given set of categories. In the '90s this perspective has been overturn, and the *machine learning* paradigm to automated TC has emerged and definitely superseded the knowledge-engineering approach. Within the machine learning paradigm, a general inductive process automatically builds an automatic text classifier by "learning", from a set of previously classified documents, the characteristics of the categories of interest. The advantages of this approach are an accuracy comparable to human performance and a considerable savings in terms of expert manpower, since no intervention from either knowledge engineers or domain experts is needed. Current day TC may thus be seen as the meeting point of machine learning and *information retrieval* (IR), the "mother" of all disciplines concerned with automated content-based document management.

TC enjoys quite a rich literature now, but this is still fairly scattered. Two international journals have devoted special issues to this topic [Carbonell et al. 2000; Lewis and Hayes 1994]. There are no textbooks nor journals devoted to TC yet; only one textbook on machine learning [Mitchell 1996, Section 6.10] devotes a brief section to it. As a note, we should warn the reader that the term "automatic text classification" has sometimes been used in the literature to mean quite different things from the ones discussed in this paper. Aside from (i) the automatic assignment of documents to a predefined set of categories, which is the main topic of this paper, the term has also been used to mean (ii) the automatic definition of such a set of categories (nowadays universally referred to as *clustering*), or (iii) the automatic assignment of documents to a set of categories which is not predefined (a task nowadays universally referred to as *(free text) indexing*). The reader should keep this in mind especially when browsing through early literature, dating to a period in which terminology had not yet settled down.

This paper is organised as follows. In Section 2 we formally define the TC task and its various subcases. In Section 3 we review the most important applications to which TC has been put to, including automatic indexing for Boolean IR systems, and document filtering. In Section 4 we go on to describe the main notions of the machine learning approach to classification, including the notions of *learner*, *training set*, *test set* and *validation set*. Our discussion of *text* classification starts in Section 5 by introducing the issues related to *indexing*, i.e. the transformation of textual documents into a form amenable to interpretation by a classifier-building algorithm (and, successively, by the classifier built by it). These issues include *term weighting* and, perhaps more importantly, *dimensionality reduction*. Section 6 tack-

les what is probably the central issue of this paper, i.e. the inductive construction of a text classifier from a training set of manually classified documents. Various algorithms for the inductive construction of classifiers, from the tradition of either IR or machine learning, are reviewed here, including probabilistic algorithms, neural networks, regression models, decision tree and DNF rule induction methods, profile-construction methods, example-based classifiers, support vector machines, and classifier committees. Most of the classifiers built by these algorithms output a value of confidence in the correctness of a given classification decision, a value usually comprised in the [0,1] interval. In order to arrive at a yes/no decision, policies have to be devised for the individuation of *thresholds* that map the [0,1] interval into the {0,1} set of Boolean values; this is the topic of Section 7. Section 8 tackles a central issue in TC, namely that of evaluating (and comparing) the classifiers induced by means of the methods of Sections 6 and 7. This is a fairly sensitive issue, and it is widely believed that only a standardisation in the evaluation protocols may yield reliable comparisons. We discuss the various measures that have been proposed and used in the literature, the most important benchmarks on which experimentation has been carried out, and some of the conclusions these experiments have lead to. Section 9 takes a slight detour from the main path and discusses a specific application of TC, namely that of Web page categorisation into hierarchical catalogues, an application that has given rise to specific solutions to the problems of indexing, classifier construction, and classifier evaluation. Finally, Section 10 concludes, discussing some of the open issues and possible avenues of further research for TC.

2. A DEFINITION OF THE TEXT CATEGORISATION TASK

Text categorisation may be defined as the task of determining an assignment of a value from {0,1} to each entry a_{ij} of the *decision matrix*

	d_1	...	...	d_j	...	...	d_n
c_1	a_{11}	...	...	a_{1j}	...	...	a_{1n}
...	...	...	...	...	...	...	...
c_i	a_{i1}	...	...	a_{ij}	...	...	a_{in}
...	...	...	...	...	...	...	...
c_m	a_{m1}	...	...	a_{mj}	...	...	a_{mn}

where $C = \{c_1, \ldots, c_m\}$ is a set of pre-defined *categories*, and $D = \{d_1, \ldots, d_n\}$ is a set of documents to be classified. A value of 1 for a_{ij} indicates a decision to file d_j under c_i, while a value of 0 indicates a decision not to file d_j under c_i.

More formally, the task is to approximate the unknown total function $f : D \times C \rightarrow \{0, 1\}$ (that describes how documents ought to be classified) by means of a total function $f' : D \times C \rightarrow \{0, 1\}$ (called the *classifier*, or *model*, or *hypothesis*) such that f and f' coincide as much as possible; how to define precisely and measure this degree of coincidence (that we will call *effectiveness*) will be discussed in detail in Section 8.1.

Fundamental to the understanding of this task are two observations:

—the categories are just symbolic labels. No additional knowledge (either of a procedural or of a declarative nature) of their "meaning" is assumed available to

help in the process of building the classifier. In particular, this means that the "text" constituting the category label (e.g. `Sports` in a news categorisation task) is not to be used.

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

Given that the semantics of a document is an inherently *subjective* notion, it follows that the fundamental notion of TC, that of *relevance* of a document to a category, cannot be decided deterministically. This is exemplified by the well-known phenomenon of *inter-indexer inconsistency* [Cleverdon 1984; Hamill and Zamora 1980]: when two different humans must take a decision on whether to classify document d_j under category c_i, they may disagree, and this in fact happens with relatively high frequency. A news article on the Clinton-Lewinsky case could be filed under `Politics`, or under `Gossip`, or under both, or even under neither, depending on the subjective judgment of the classifier. The above-mentioned notion of relevance of a document to a category basically coincides with the notion of relevance of a document to an information need, as from IR [Saracevic 1975].

2.1 Single-label and multi-label categorisation

Different constraints may be enforced on the categorisation task, depending on the application.

For instance, we might want that for a given integer k, each element of C must be assigned to exactly k (or $\leq k$, or $\geq k$) elements of D. For instance, this happens when we want categories to be evenly populated, or when we want them to be populated each to a certain degree (see also Section 7).

More importantly, we might want that for a given integer k, exactly k (or $\leq k$, or $\geq k$) elements of C must be assigned to each element of D. The case $k = 1$ (as e.g. in [Baker and McCallum 1998; Cohen and Hirsch 1998; Guthrie et al. 1994; Koller and Sahami 1997; Joachims 1997; Larkey 1999; Li and Jain 1998; Moulinier and Ganascia 1996; Schütze et al. 1995]) is often called the *single-label* case (or the *non-overlapping categories* case), whereas the general case in which any number of categories from 0 to m may be assigned to the same document is dubbed the *multi-label* case.

From a theoretical point of view, the single-label case is more general than the multi-label case, in the sense that an algorithm for single-label classification can also be used for multi-label classification by simply transforming a problem of multi-label classification with categories $\{c_1, \ldots, c_m\}$ into m independent problems of single-label classification with categories $\{c_i, \overline{c}_i\}$, for $i = 1, \ldots, m$. This requires, however, that categories are stochastically independent of each other, i.e. $f(d_j, c')$ does not depend on $f(d_j, c'')$ and viceversa, which is usually assumed to be the case (exceptions to this rule will be dealt with in Section 9). The converse is not true in general: if we have an algorithm for performing multi-label classification, it is not

always the case that we can use it for single-label classification too. In fact (i) it might not be obvious how to choose a single "best" category among the k categories that the classifier has attached to the document, or (ii) for some documents k might be equal to 0.

In general, the techniques we will consider in this paper are applicable irrespectively of whether any of the constraints discussed in this section are enforced or not; therefore, from now on we will assume that no constraint of this sort is enforced. In particular, we will mostly be concerned with the single-label case, given its greater generality. This means that we we will view the classification problem for the $D \times C$ decision matrix as consisting of m independent problems of classifying the documents in D under a given category c_i, for $i = 1, \ldots, m$. A *classifier for* c_i is then a function $f'_i : D \rightarrow \{0, 1\}$ that approximates an unknown function $f_i : D \rightarrow \{0, 1\}$.

2.2 Category-pivoted and document-pivoted categorisation

An important distinction is whether we want to fill the decision matrix one row at a time (*category-pivoted categorisation* – CPC), or fill it one column at a time (*document-pivoted categorisation* – DPC). Quite obviously this distinction is more pragmatic than conceptual, but is important in the sense that the sets C of categories and D of documents might not always be available in their entirety right from the start. It is also of some relevance to the choice of the method for building the classifier, as some of these methods (e.g. the k-NN method of Section 6.8) allow the construction of classifiers with a definite slant towards one or the other classification style.

DPC is thus suitable when documents might become available one at a time over a long span of time, e.g. in the case a user submits one document at a time for categorisation rather than submitting a whole batch of them all at once. In this case, sometimes the categorisation task takes the form of ranking the categories in decreasing order of their estimated appropriateness for document d_j; because of this, DPC is sometimes called *category-ranking classification* or *on-line classification* [Yang 1999].

CPC is instead suitable if we consider the possibility that a new category c_{m+1} is inserted into a previously existing set of categories $C = \{c_1, \ldots, c_m\}$ after a number of documents have already been evaluated for categorisation under C, which means that these documents need to be evaluated for c_{m+1} too (e.g. [Larkey 1999]). In this case, the categorisation task might take the form of ranking the documents in decreasing order of their estimated appropriateness for category c_{m+1}; symmetrically to the previous case, CPC may also be called *document-ranking classification*.

DPC is more commonly used than CPC, as the case in which documents are submitted one at a time is somehow more common than the case in which newer categories dynamically crop up. However, although some specific techniques apply to one and not to the other (e.g. the proportional thresholding method discussed in Section 7, which applies only to CPC), this is more the exception than the rule: most of the techniques we will discuss in this paper allow the construction of classifiers capable of working in either mode.

3. APPLICATIONS OF DOCUMENT CATEGORISATION

Automatic TC goes back at least to the early '60s and to Maron's [1961] seminal work. Since then, it has been used in a number of different applications. In the following, we briefly review the most important ones. Other applications we do not explicitly discuss for reasons of space are speech categorisation by means of a combination of speech recognition and TC [Schapire and Singer 2000], multimedia document categorisation through caption analysis [Sable and Hatzivassiloglou 1999], author identification for literary texts of unknown or disputed authorship [Forsyth 1999], and (gasp!) automatic essay grading [Larkey 1998].

3.1 Automatic indexing for Boolean information retrieval systems

The first use to which automatic text classifiers were put at, and the application that spawned most of the early research in the field [Borko and Bernick 1963; Fangmeyer and Lustig 1968; Field 1975; Gray and Harley 1971; Hamill and Zamora 1978; Hamill and Zamora 1980; Heaps 1973; Hoyle 1973; Klingbiel 1973a; Klingbiel 1973b; Maron 1961], is that of automatic document indexing for use in information retrieval (IR) systems relying on a controlled dictionary. The most prominent example of such IR systems is, of course, that of Boolean systems. In these systems, each document is assigned one or more keywords or keyphrases describing its content, where these keywords and keyphrases belong to a finite set of words called *controlled dictionary* and often consisting of a hierarchical thesaurus (e.g. the NASA thesaurus for the aerospace discipline, or the MESH thesaurus covering the medical field). Usually, this assignment is performed by trained human indexers, and is thus an extremely costly activity.

If the entries in the thesaurus are viewed as categories, document indexing becomes an instance of the document categorisation task, and may thus be addressed by the automatic techniques described in this paper. Recalling Section 2.1, note that in this case a typical constraint may be that $k_1 \leq x \leq k_2$ keywords are assigned to each document, for given k_1, k_2. Document-pivoted categorisation might typically be the best option, so that new documents may be classified as they become available. Various automatic document classifiers explicitly addressed at document indexing applications have been described in the literature; see e.g. [Fuhr and Knorz 1984; Fuhr 1985; Biebricher et al. 1988; Fuhr et al. 1991; Robertson and Harding 1984; Tzeras and Hartmann 1993].

The issue of automatic indexing with controlled dictionaries is closely related to the topic of *automated metadata generation*. In digital libraries we are usually interested to tag documents by metadata that describe them under a variety of aspects (e.g. creation date, document type or format, availability, etc.). Usually, some of these metadata are *thematic*, i.e. their role is to describe the semantics of the document by means of bibliographic codes, keywords or keyphrases. The generation of these metadata may thus be viewed as a problem of document indexing with controlled dictionary, and thus tackled by means of automatic TC techniques. An example system for automated metadata generation by TC techniques is the KLARITY system (`http://www.topic.com.au/products/klarity.html`).

3.2 Document organisation

In general, all issues pertaining to document organisation and filing, be it for purposes of personal organisation or document repository structuring, may be addressed by automatic categorisation techniques. For instance, at the offices of a newspaper, incoming "classified" ads must be, prior to publication, categorised under the categories used in the categorisation scheme adopted by the newspaper; typical categories might be e.g. `Personals`, `Cars for Sale`, `Real Estate`, ... While most newspapers would handle this application manually, those dealing with a high daily number of classified ads might prefer an automatic categorisation system to choose the most suitable category for a given ad. In this case a typical constraint might be that exactly one category is assigned to each document. A first-come, first-served policy might look the aptest here, which would make one lean for a document-pivoted categorisation style. Similar applications might be the automatic filing of newspaper articles under the appropriate sections (e.g. `Politics`, `Home News`, `Lifestyles`, etc.), or the automatic grouping of conference papers into sessions.

Document organisation, both in the cases of paper documents and electronic documents, often has the purpose of making document search easier. An interesting example of this approach is the system for classifying and searching patents of the U.S. Patent and Trademark Office, described by Larkey [1999]. In this system documents describing patents are classified according to a hierarchical set of categories. Patent office personnel may thus search for existing patents related to a claimed new invention with greater ease.

3.3 Document filtering

Document filtering refers to the activity of classifying a *dynamic* collection of documents, typically in the form of a stream of incoming documents dispatched in an asynchronous way by an information producer to an information consumer [Belkin and Croft 1992]. A typical case of this is a newsfeed, whereby the information producer is a news agency (e.g. Reuters or Associated Press) and the information consumer is a newspaper. In this case, the filtering system should block the delivery to the consumer of the documents the consumer is not likely to be interested in (e.g. all news not concerning sports, in the case of a sports newspaper). Filtering can be seen as a case of single-label categorisation, i.e. the categorisation of incoming documents in two disjoint categories, the relevant and the irrelevant. Additionally, a filtering system may also perform a further categorisation into topical categories of the documents deemed relevant to the consumer; in the example above, all articles about sports are deemed relevant, and should be further subcategorised according e.g. to which sport they deal with, so as to allow individual journalists specialised in individual sports to access only documents of high prospective interest for them. Similarly, an e-mail filter might be trained to further classify previously filtered e-mail into topical categories of interest to the user [Cohen 1996].

A document filtering system may be installed at the producer end, in which case its role is to route the information to the interested consumers only, or at the consumer end, in which case its role is to block the delivery of information deemed uninteresting to the user. In the former case the system has to build and update a

"profile" for each consumer it serves [Liddy et al. 1994], whereas in the latter case (which is the more common, and to which we will refer in the rest of this section) a single profile is needed.

A profile may be initially specified by the user, thereby resembling a standing IR query, and is usually updated by the system by using feedback information provided by the user on the relevance or non-relevance of the delivered messages. In the TREC community [Lewis 1995c; Hull 1998] this is called *adaptive filtering*, while the case in which no user-specified profile is available is called either *routing* or *batch filtering*, depending on whether documents have to be ranked in decreasing order of estimated relevance or just accepted/rejected.

In information science document filtering has a tradition dating back to the '60s, when, addressed by systems of varying degrees of automation and dealing with the multi-consumer case discussed above, it was variously called *selective dissemination of information* or *current awareness* (see e.g. [Korfhage 1997, Chapter 6]). The explosion in the availability of digital information, particularly on the Internet, has boosted the importance of such systems. These are nowadays being used in many different contexts, including the creation of personalised Web newspapers, "junk e-mail" blocking, and the selection of Usenet news.

The construction of information filtering systems by means of machine learning techniques is widely discussed in the literature: see e.g. [Amati and Crestani 1999; Hull 1994; Hull et al. 1996; Lang 1995; Lewis et al. 1996; Schapire et al. 1998; Schütze et al. 1995; Singhal et al. 1997; Weiss et al. 1999].

3.4 Word sense disambiguation

Word sense disambiguation (WSD) refers to the activity of finding, given the occurrence in a text of an ambiguous (i.e. polysemous or homonymous) word, the sense this particular word occurrence has. For instance, the English word `bank` may have (at least) two different senses, as in `the Bank of England` (a financial institution) or `the bank of river Thames` (a hydraulic engineering artifact). It is thus a WSD task to decide to which of the above senses the occurrence of `bank` in `Last week I borrowed some money from the bank` refers to. WSD is very important for a number of applications, including indexing documents by word senses rather than by words for IR or other content-based document management applications.

WSD may be seen as a categorisation task once we view word occurrence contexts as documents and word senses as categories. Quite obviously, this is a case in which exactly one category needs to be assigned to each document, and one in which document-pivoted categorisation is most likely to be the right choice. WSD is viewed as a TC task in a number of different works in the literature; see e.g [Gale et al. 1993; Hearst 1991].

WSD is just an example of the more general issue of resolving natural language ambiguities, one of the most important problems in computational linguistics. Other instances of this problem, which may all be tackled by means of TC techniques along the lines discussed for WSD, are *context-sensitive spelling correction*, *prepositional phrase attachment*, *part of speech tagging*, and *word choice selection* in machine translation. See the excellent [Roth 1998] for an introduction to this field.

wheat & *farm*	→	WHEAT
wheat & *commodity*	→	WHEAT
bushels & *export*	→	WHEAT
wheat & *agriculture*	→	WHEAT
wheat & tonnes	→	WHEAT
wheat & *winter* & ¬ *soft*	→	WHEAT

Fig. 1. Classifier for the WHEAT category in the CONSTRUE system; keywords occurring in documents are indicated in *italic*, categories are indicated in SMALL CAPS (from [Apté et al. 1994]).

3.5 Yahoo!-style search space categorisation

Automatic document categorisation has recently aroused a lot of interest also for its possible Internet applications. One of these is automatically classifying Web pages, or sites, into one or several of the categories that make up commercial hierarchical catalogues such as those embodied in YAHOO!, INFOSEEK, etc. When Web documents are catalogued in this way, rather than addressing a generic query to a general-purpose Web search engine, a searcher may find it easier to first navigate in the hierarchy of categories and then issue her search from (i.e. restrict her search to) a particular category of interest.

Automatically classifying Web pages has obvious advantages, since the manual categorisation of a large enough subset of the Web is problematic to say the least. Unlike in the previous applications, this is a case in which one might typically want each category to be populated by a set of $k_1 \leq x \leq k_2$ documents, and one in which category-centered categorisation may be aptest so as to allow new categories to be added and obsolete ones to be deleted.

The automatic categorisation of Web pages or sites into YAHOO!-like hierarchical catalogues is discussed in several recent papers (see e.g. [Attardi et al. 1999; Baker and McCallum 1998; Chakrabarti et al. 1998; McCallum et al. 1998; Mladenić 1998b]), and will be more extensively discussed in Section 9.

4. THE MACHINE LEARNING APPROACH TO TEXT CATEGORISATION

In the '80s the main approach used to the realisation of automatic document classifiers consisted in their manual construction through *knowledge-engineering* techniques, i.e. in manually building an expert system capable of taking categorisation decisions. Such an expert system typically consisted of a set of manually defined rules (one per category) of type **if** ⟨*DNF Boolean formula*⟩ **then** ⟨*category*⟩, to the effect that if the document satisfied ⟨*DNF Boolean formula*⟩ (*DNF* standing for "disjunctive normal form"), then it was classified under ⟨*category*⟩. The typical example of this approach is the CONSTRUE system [Hayes et al. 1990], built by Carnegie Group for use at the Reuters news agency. A sample rule from CONSTRUE is illustrated in Figure 1, and its effectiveness as measured on a benchmark selected by the authors is reported in Figure 2. Other examples of this "manual" approach to the construction of text classifiers are [Goodman 1990; Rau and Jacobs 1991].

The drawback of this "manual" approach to the construction of automatic classifiers is the existence of a *knowledge acquisition bottleneck*, similarly to what happens in expert systems. That is, rules must be manually defined by a knowledge engineer with the aid of a domain expert (in this case, an expert in document relevance to the

		expert judgments	
		WHEAT	¬ WHEAT
classifier judgments	WHEAT	73	8
	¬ WHEAT	14	3577

Fig. 2. Effectiveness of the classifier of Figure 1 as measured on a subset of the Reuters collection (from [Apté et al. 1994]).

chosen set of categories). If the set of categories is updated, then these two trained professionals must intervene again, and if the classifier is ported to a completely different domain (i.e. set of categories) the work has to be repeated anew.

On the other hand, it was suggested that this approach can give very good effectiveness results: Hayes et al. [1990] report a .90 "breakeven" result (see Section 8) on a subset of the Reuters test collection, a figure that outperforms even the best classifiers built in the late '90s by machine learning techniques. However, no other classifier has been tested on the same dataset as CONSTRUE (see also Table 6), and it is not clear how this dataset was selected from the Reuters collection (i.e. whether it was a random or a favourable subset of the whole collection). All in all, as convincingly argued in [Yang 1999], the results above do not allow us to confidently say that these effectiveness results may be obtained in the general case.

Since the early '90s, a new approach to the construction of automatic document classifiers (the *machine learning approach*) has gained prominence and eventually become the dominant one (see [Mitchell 1996] for a comprehensive introduction to machine learning). In this approach a general inductive process (also called the *learner*) automatically builds a classifier for a category c_i by "observing" the characteristics of a set of documents that have previously been classified manually under c_i by a domain expert; from these characteristics, the inductive process gleans the characteristics that a novel document should have in order to be classified under c_i. In machine learning terminology, the classification problem is an activity of *supervised* learning, since the learning process is driven, or "supervised", by the knowledge of the categories to which the training instances belong[1].

The advantages of this approach over the previous one are evident: the engineering effort goes towards the construction not of a classifier, but of an automatic builder of classifiers. This means that if the original set of categories is updated, or if the system is ported to a completely different domain, all that is needed is the inductive, *automatic* construction of a new classifier from a different set of manually classified documents, with no required intervention of either the domain expert or the knowledge engineer.

In terms of effectiveness, classifiers built by means of machine learning techniques nowadays achieve impressive levels of performance (see Section 8), making automatic classification a *qualitatively* (and not only economically) viable alternative to manual classification.

[1]Within the area of content-based document management tasks, an example of an *unsupervised* learning activity is *document clustering* (see e.g. [Willett 1988b]).

4.1 Training set and test set

As previously mentioned, the machine learning approach relies on the existence of a an *initial corpus* $Co = \{\overline{d}_1, \ldots, \overline{d}_s\}$ of documents previously classified under the same set of categories $C = \{c_1, \ldots, c_m\}$ with which the system will need to operate. This means that the initial corpus comes with a *correct decision matrix*

	Training set				Test set			
	$\overline{d}_1$	...	...	$\overline{d}_g$	$\overline{d}_{g+1}$	...	...	$\overline{d}_s$
c_1	ca_{11}	...	...	ca_{1g}	$ca_{1(g+1)}$	...	...	ca_{1s}
...	...	...	...	...	...	...	...	...
c_i	ca_{i1}	...	...	ca_{ig}	$ca_{i(g+1)}$	...	...	ca_{is}
...	...	...	...	...	...	...	...	...
c_m	ca_{m1}	...	...	ca_{mg}	$ca_{m(g+1)}$	...	...	ca_{ms}

A value of 1 for ca_{ij} is interpreted as an indication from the expert to file d_j under c_i, while a value of 0 is interpreted as an indication from the expert not to file d_j under c_i. A document d_j is called a *positive example* of c_i if $ca_{ij} = 1$, a *negative example* of c_i if $ca_{ij} = 0$. TC may then be reformulated as the task of approximating the function $f : (D \cup Co) \times C \rightarrow \{0, 1\}$, unknown for all $d \in D$ and known for all $\overline{d} \in Co$, by means of a function $f' : (D \cup Co) \times C \rightarrow \{0, 1\}$.

For evaluation purposes, in the first stage of classifier construction the initial corpus is typically divided into two sets, not necessarily of equal size:

—a *training set* $Tr = \{\overline{d}_1, \ldots, \overline{d}_g\}$. This is the set of example documents observing the characteristics of which the classifiers for the various categories are induced;

—a *test set* $Te = \{\overline{d}_{g+1}, \ldots, \overline{d}_s\}$. This set will be used for the purpose of testing the effectiveness of the induced classifiers. Each document in Te will be fed to the classifiers, and the classifier decisions compared with the expert decisions; a measure of classification effectiveness will be based on how often the values for the a_{ij}'s obtained by the classifiers match the values for the ca_{ij}'s provided by the experts.

Note that in order to give a scientific character to the experiment the documents in Te cannot participate in any way in the inductive construction of the classifiers; if this condition were not satisfied, the experimental results obtained would probably be unrealistically good [Mitchell 1996, page 129].

This approach is called the *train-and-test* approach. An alternative approach is the *k-fold cross-validation* approach (see e.g. [Mitchell 1996, page 146]), whereby t different classifiers are induced by partitioning the initial corpus into k disjoint sets $Te_1, \ldots, Te_k$, and then iteratively applying the train-and-test approach on pairs $\langle Tr_i = Co - Te_i, Te_i \rangle$. The resulting t classifiers $\Phi_1, \ldots, \Phi_t$, different among each other because they have been generated from t different training sets, are then averaged in some way to yield the final classifier. This approach (that is clearly reminiscent of the "classifier committee" approach that will be discussed in Section 6.10) is usually adopted when the initial corpus is small and the training process cannot afford to lose the information present in the test documents. This is hardly the case in TC applications, where initial corpora are usually large at will.

In the train-and-test approach, it is often the case that in order to optimise a classifier, its internal parameters should be tuned by testing which values of the parameters yield the best effectiveness. In order to make this optimisation possible while at the same time safeguarding scientific standards, the set $\{\overline{d}_1, \ldots, \overline{d}_g\}$ may be further split into a "true" training set $Tr = \{\overline{d}_1, \ldots, \overline{d}_f\}$, from which the classifier is inductively constructed, and a *validation set* $Va = \{\overline{d}_{f+1}, \ldots, \overline{d}_g\}$ (sometimes called a *hold-out set*), on which the repeated tests of the induced classifier aimed at parameter optimisation are performed[2].

Given a corpus Co, one may define the *generality* $g_{Co}(c_i)$ of a category c_i as the percentage of documents that belong to c_i, i.e.:

$$g_{Co}(c_i) = \frac{|\ \{\overline{d}_j \in Co \mid ca_{ij} = 1\}\ |}{|\ \{\overline{d}_j \in Co\}\ |} \tag{1}$$

The *training set generality* $g_{Tr}(c_i)$, *validation set generality* $g_{Va}(c_i)$, and *test set generality* $g_{Te}(c_i)$ of a category c_i may be defined in the obvious way by substituting Tr, Va, or Te, respectively, to Co in Equation 1.

4.2 Information retrieval techniques and text categorisation

The machine learning approach to classifier construction heavily relies on the basic machinery of information retrieval. The reason is that both IR and TC are *content-based document management tasks*, and therefore share many characteristics.

IR techniques are used in three phases of the text classifier life cycle:

(1) IR-style *indexing* is always performed on the documents of the initial corpus and on those to be categorised during the operating phase of the classifier;
(2) IR-style techniques (such as document-request matching, query reformulation, ...) are often used in the *inductive construction* of the classifiers;
(3) IR-style *evaluation* of the effectiveness of the classifiers is performed.

The various approaches to classification differ mostly for how they tackle Step 2, although in a few cases non-standard approaches to Steps 1 and 3 are also used. Steps 1, 2 and 3 will be the main themes of Sections 5, 6-7 and 8, respectively.

5. INDEXING AND DIMENSIONALITY REDUCTION

Text documents, as they are, are not amenable to being interpreted by a classifier or by a classifier-building algorithm. Because of this, an *indexing* procedure that maps a text d into a succinct representation of its content needs to be invoked. Although numerous indexing methods exist, it goes without saying that *the same* indexing procedure should uniformly be applied to training, validation and test documents alike.

The choice of a representation for text depends on what one regards as the meaningful textual units (the problem of *lexical semantics*) and the meaningful natural language rules for the combination of these units (the problem of *compositional semantics*). In true IR style, each document is usually represented by a vector

[2]From now on, we will take the freedom to use the expression "test document" to denote any document not in the training set and validation set. This includes thus any document submitted to the classifier in its operating phase.

of n weighted *index terms* (hereafter simply *terms*) that occur in the document; differences among the various approaches are accounted for by

(1) different ways to understand what a term is;

(2) different ways to weight terms.

A typical choice for Issue 1 is to identify terms with all the words occurring in the document. This is often referred to as the *bag of words* approach to document representation. In a number of experiments [Apté et al. 1994; Dumais et al. 1998; Lewis 1992] it has been found that representations more sophisticated than this yield worse categorisation effectiveness, thereby confirming similar results from IR [Salton and Buckley 1988]. In particular, a number of authors have tried to use *noun phrases*, rather than individual words, as indexing terms, but the experimental results found to date have not been encouraging, irrespectively of whether the notion of "phrase" is motivated

—*syntactically*, i.e. the phrase is such according to a grammar of the language (see e.g. [Fuhr et al. 1991; Lewis 1992; Tzeras and Hartmann 1993]);

—*statistically*, i.e. the phrase is not grammatically such, but is composed of a set/sequence of words that occur contiguously with high frequency in the collection (see e.g. [Schütze et al. 1995]).

Quite convincingly, Lewis [1992] argues that the likely reason for the discouraging results is that, although indexing languages based on phrases have superior semantic qualities, they have inferior statistical qualities with respect to indexing languages based on single words. Notwithstanding these discouraging results, investigations on the effectiveness of phrase indexing are still being actively pursued. This is true especially of statistically motivated phrases [Cohen and Singer 1999; Mladenić 1998b; Schapire et al. 1998], since in this case Lewis' argument above applies to a smaller degree.

As for Issue 2, weights usually range between 0 and 1, and with no loss of generality we will assume they always do. As a particular case, a few authors (e.g. [Apté et al. 1994; Koller and Sahami 1997; Lewis and Ringuette 1994; Li and Jain 1998; Moulinier et al. 1996; Moulinier and Ganascia 1996; Schapire and Singer 2000; Schütze et al. 1995]) use binary weights, due to the symbolic, non-numeric nature of the learning systems they employ; in this case, 1 denotes presence and 0 absence of the term in the document. In the more frequent case of non-binary indexing, for determining the weight w_{kj} of term t_k in document d_j any IR-style indexing technique that represents a document as a vector of weighted terms may be used. Most of the times, the standard $tfidf$ weighting function is used (see e.g. [Salton and Buckley 1988]), defined as

$$tfidf(t_k, d_j) = \#(t_k, d_j) \cdot \log \frac{|Tr|}{\#_{Tr}(t_k)} \tag{2}$$

where $\#(t_k, d_j)$ denotes the number of times t_k occurs in d_j, and $\#_{Tr}(t_k)$ denotes the number of documents in Tr in which t_k occurs at least once (also known as the *document frequency* of term t_k). This function encodes the intuitions that (i) the more often a term occurs in a document, the more it is representative of the

content of the document, and (ii) the more documents the term occurs in, the less discriminating it is[3].

Note that this formula (as most other indexing formulae) weights the importance of a term to a document in terms of occurrence considerations only, thereby deeming of null importance the order in which the terms themselves occur in the document and the syntactic role they play; in other words, the semantics of a document is reduced to the lexical semantics of the terms that occur in it, thereby disregarding the issue of compositional semantics (an exception to this are the representation techniques used for the FOIL system [Cohen 1995a] and for the SLEEPING EXPERTS system [Cohen and Singer 1999]).

In order to make weights fall in the [0,1] interval and documents be represented by vectors of equal length, the weights resulting from $tfidf$ are often normalised by *cosine normalisation*, given by:

$$w_{kj} = \frac{tfidf(t_k, d_j)}{\sqrt{\sum_{s=1}^{r}(tfidf(t_s, d_j))^2}} \tag{3}$$

where r is the set of all terms that occur at least once in Tr.

Although $tfidf$ is by far the most popular one, other indexing functions have also been used, including probabilistic indexing methods [Fuhr et al. 1998] or techniques for indexing structured documents [Larkey and Croft 1996]. Functions different from $tfidf$ are especially needed when the training set is not available in its entirety from the start and document frequency data are thus unavailable, as e.g. in adaptive filtering; in this case, more empirical substitutes of $tfidf$ are usually employed [Dagan et al. 1997, Section 4.3].

Before indexing, the removal of *function words* (i.e. topic-neutral words such as articles, prepositions, conjunctions, etc.) is always performed. Concerning *stemming* (i.e. collapsing words that share the same morphological root), it is controversial whether it is a beneficial step for TC. Although, similarly to unsupervised term clustering (see Section 5.3.1) of which it is an instance, stemming has sometimes been reported to hurt effectiveness (e.g. [Baker and McCallum 1998]), the recent tendency is to adopt it (e.g. [Larkey and Croft 1996; Ng et al. 1997; Schütze et al. 1995; Wiener et al. 1995; Yang 1999]), as it considerably reduces both the dimensionality of the term space (see Section 5.1) and the level of stochastic dependence between terms (see Section 6.1).

Depending on the application, either the full text of the document or selected parts of it may be indexed. While the former option is the rule, exceptions do exist. For instance, in a patent categorisation application Larkey [1999] considers only the title, the abstract, the first twenty lines of the background summary, and the section containing the claims of novelty of the described invention. This approach is made possible by the fact that documents describing patents are structured. Similarly, when a document title is available, it is possible to pay extra importance to the words appearing therein [Apté et al. 1994; Cohen and Singer 1999; Weiss et al.

[3]Actually, $tfidf$ is, rather than a function, a whole class of functions, which differ from each other in terms of normalisation or other correction factors being applied or not. Formula 2 is then just one of the possible instances of this class; see [Salton and Buckley 1988] for variations on this theme.

1999]. Instead, in those applications in which documents are flat the identification of the most relevant part of a document is a non-obvious task.

5.1 Dimensionality reduction

Unlike in IR, in TC the high dimensionality of the term space (i.e. the fact that the number r of terms that occur at least once in the corpus Co is high) may be problematic. In fact, while the typical matching algorithms used in IR (such as cosine matching) scale well to high values of r, the same cannot be said of many among the sophisticated learning algorithms used for classifier induction (e.g. the LLSF algorithm of [Yang and Chute 1994]). Because of this, techniques for *dimensionality reduction* (DR) are often employed whose effect is to reduce the dimensionality of the vector space from r to $r' \ll r$.

Dimensionality reduction is also beneficial since it tends to reduce the problem of *overfitting*, i.e. the phenomenon by which a classifier is tuned also to the *contingent*, rather than just the *necessary* (or constitutive) characteristics of the training data[4]. Classifiers which overfit the training data tend to be extremely good at classifying the data they have been trained on, but are remarkably worse at classifying other data. For example, if a classifier for category `Cars for sale` were trained on just three positive examples among which two concerned the sale of a yellow car, the resulting classifier would deem "yellowness", clearly a contingent property of these particular training data, as a constitutive property of the category. Experimentation has shown that in order to avoid overfitting a number of training examples roughly proportional to the number of terms used is needed; Fuhr and Buckley [1991, page 235] have suggested that 50-100 training examples per term may be needed in TC tasks. This means that, if DR is performed, overfitting may be avoided even if a smaller amount of training examples is used.

Various DR functions, either from the information theory or from the linear algebra literature, have been proposed, and their relative merits have been tested by experimentally evaluating the variation in categorisation effectiveness that a given classifier undergoes after application of the function to the term space it operates on.

There are two quite distinct ways of viewing DR, depending on whether the task is approached locally (i.e. for each individual category, in isolation of the others) or globally[5]:

—*local dimensionality reduction*: for each category c_i, $r'_i \ll r$ terms are chosen to support the classification under category c_i (see e.g. [Apté et al. 1994; Lewis and Ringuette 1994; Li and Jain 1998; Ng et al. 1997; Sable and Hatzivassiloglou 1999; Schütze et al. 1995; Wiener et al. 1995]). Conceptually, this would mean that each document d_j has a different representation for each category c_i; in practice, though, this means that different subsets of d_j's original representation are used when classifying under the different categories. Authors who have adopted this approach tend to adopt values of $10 \leq r' \leq 50$.

[4]The overfitting problem is often referred to as "the curse of dimensionality".

[5]A third way of viewing DR, and exemplified in [Koller and Sahami 1997], is briefly discussed in Section 9.

—*global dimensionality reduction*: $r' \ll r$ terms are chosen to support the classification under all categories $C = \{c_1, \ldots, c_m\}$ (see e.g. [Mladenić 1998a; Yang 1999; Yang and Pedersen 1997]).

This distinction usually does not impact on the kind of technique chosen for DR, since most DR techniques can be used (and have been used) for local and for global DR alike.

A second, orthogonal distinction may be drawn in terms of the nature of the resulting terms:

—*dimensionality reduction by term selection*: the r' chosen terms are a subset of the r original terms;

—*dimensionality reduction by term extraction*: the r' terms are not a subset of the original r terms. Usually, the former are not homogeneous with the latter (e.g. if the original r terms are words, the r' chosen terms may not be words at all), but are obtained by combinations or transformations of the original ones.

Quite obviously, and unlike in the previous distinction, the two different ways of doing DR are tackled by quite distinct techniques; we will tackle them separately in the next two sections.

5.2 Dimensionality reduction by term selection

Given a fixed $r' \ll r$, techniques for term selection (also called *term space reduction* – TSR) purport to select, from the original set of r terms, the r' terms that, when used for document indexing, yield the smallest reduction in effectiveness with respect to the effectiveness that would be obtained by using full-blown representations. Results published in the literature [Yang and Pedersen 1997] have even shown a moderate ($\leq 5\%$) increase in effectiveness after term space reduction has been performed, depending on the classifier, on the *aggressivity* $\frac{r}{r'}$ of the reduction, and on the TSR technique used.

Moulinier et al. [1996] have experimented with a so-called *wrapper* term selection method, i.e. one in which the term set is identified by means of the same learning method which will be used for inducing the classifier [John et al. 1994]. Starting from an initial term set, a new term set is generated by either adding or removing a term. When a new term set is generated, a classifier based on it is induced and then tested on a validation set. The term set that results in the best effectiveness is chosen. This approach has the advantage that it is by definition tuned to the learning algorithm being used, and that, if used for local dimensionality reduction, different numbers of terms for different categories may be chosen, depending on whether a category is or is not easily separable from the others. However, this approach surely qualifies as a brute force method, and the number of possible different term sets renders its cost prohibitive for standard TC applications.

A computationally easier alternative to the wrapper approach is the *filtering* approach [John et al. 1994], i.e. keeping the $r' \ll r$ terms that score highest according to a predetermined numerical function that measures the "importance" of the term for the categorisation task. We will explore this solution in the rest of this section.

5.2.1 *Document frequency.* A simple and surprisingly effective global TSR function is the *document frequency* $\#_{Tr}(t_k)$ of a term t_k, first used in [Apté et al. 1994]

and then systematically studied in [Yang and Pedersen 1997]. Note that in this section we will interpret the event space as the set of all documents in the training set; in probabilistic terms, document frequency may thus also be written as $P(t_k)$. Yang and Pedersen [1997] have shown that, irrespectively of the adopted classifier and of the initial corpus used, by using $\#_{Tr}(t_k)$ as a term selection technique it is possible to reduce the dimensionality of the term space by a factor of 10 with no loss in effectiveness (a reduction by a factor of 100 brings about just a small loss).

This result seems to state, basically, that the most valuable terms for categorisation are those that occur more frequently in the collection. As such, it would seem at first to contradict a truism of IR, according to which the most informative terms are those with low-to-medium document frequency [Salton and Buckley 1988]. But these two results do not contradict each other, since it is well-known (see e.g. [Salton et al. 1975]) that the overwhelming majority of the words that occur at least once in a given corpus have an extremely low document frequency; this means that by performing a TSR by a factor of 10 using document frequency, only such words are removed, while the words from low-to-medium to high document frequency are preserved. Of course, stop word removal needs to be performed before this form of dimensionality reduction is attempted, lest only topic-neutral words remain after the reduction [Mladenić 1998a].

Finally, note that a slightly more empirical form of term selection by document frequency is adopted by many authors, who remove from consideration all terms that occur in at most x training documents (popular values for x range from 1 to 3), either as the only form of dimensionality reduction [Ittner et al. 1995] or before applying another more sophisticated form [Dumais et al. 1998; Li and Jain 1998; Wiener et al. 1995]. A variant of this policy is removing from considerations all terms that occur at most x times in the collection (e.g. [Dagan et al. 1997; Joachims 1997; Joachims 1998]), with popular values for x ranging from 1 (e.g. [Baker and McCallum 1998]) to 5 (e.g. [Apté et al. 1994; Cohen 1995a]).

5.2.2 *Other information-theoretic term selection functions.* Other more sophisticated information-theoretic functions have been used in the literature, among which *chi-square* [Schütze et al. 1995; Yang and Pedersen 1997; Yang and Liu 1999], *correlation coefficient* [Ng et al. 1997; Ruiz and Srinivasan 1999], *information gain* [Larkey 1998; Lewis 1992; Lewis and Ringuette 1994; Mladenić 1998a; Moulinier and Ganascia 1996; Yang and Pedersen 1997; Yang and Liu 1999], *mutual information* [Dumais et al. 1998; Lam et al. 1997; Larkey and Croft 1996; Lewis and Ringuette 1994; Li and Jain 1998; Moulinier et al. 1996; Ruiz and Srinivasan 1999; Taira and Haruno 1999; Yang and Pedersen 1997], *odds ratio* [Mladenić 1998a; Ruiz and Srinivasan 1999], *relevancy score* [Wiener et al. 1995], and *simplified chi-square* [Fuhr et al. 1998]. The mathematical definitions of these measures are summarised for convenience in Table 1. Probabilities are interpreted as usual on an event space of documents (e.g. $P(\overline{t_k}, c_i)$ thus means the probability that, for a random document x, term t_k does not occur in x and x belongs to category c_i), and are estimated by counting occurrences in the training set. Most of these functions try to capture the intuition according to which the most valuable terms for categorisation under c_i are those that are distributed most differently in the sets of positive and negative examples of the category.

Function	Denoted by	Mathematical form
Document frequency	$\#(t_k, c_i)$	$P(t_k\|c_i)$
Information gain	$IG(t_k, c_i)$	$P(t_k, c_i) \cdot \log \frac{P(t_k, c_i)}{P(c_i) \cdot P(t_k)} + P(\overline{t}_k, c_i) \cdot \log \frac{P(\overline{t}_k, c_i)}{P(c_i) \cdot P(\overline{t}_k)}$
Mutual information	$MI(t_k, c_i)$	$\log \frac{P(t_k, c_i)}{P(t_k) \cdot P(c_i)}$
Chi-square	$\chi^2(t_k, c_i)$	$\frac{g \cdot [P(t_k, c_i) \cdot P(\overline{t}_k, \overline{c_i}) - P(t_k, \overline{c_i}) \cdot P(\overline{t}_k, c_i)]^2}{P(t_k) \cdot P(\overline{t}_k) \cdot P(c_i) \cdot P(\overline{c_i})}$
Correlation coefficient	$CC(t_k, c_i)$	$\frac{\sqrt{g} \cdot [P(t_k, c_i) \cdot P(\overline{t}_k, \overline{c_i}) - P(t_k, \overline{c_i}) \cdot P(\overline{t}_k, c_i)]}{\sqrt{P(t_k) \cdot P(\overline{t}_k) \cdot P(c_i) \cdot P(\overline{c_i})}}$
Relevancy score	$RS(t_k, c_i)$	$\log \frac{P(t_k\|c_i) + d}{P(\overline{t}_k\|\overline{c_i}) + d}$
Odds Ratio	$OR(t_k, c_i)$	$\frac{P(t_k\|c_i) \cdot (1 - P(t_k\|\overline{c_i}))}{(1 - P(t_k\|c_i)) \cdot P(t_k\|\overline{c_i})}$
Simplified Chi-square	$s\chi^2(t_k, c_i)$	$P(t_k, c_i) \cdot P(\overline{t}_k, \overline{c}_i) - P(t_k, \overline{c}_i) \cdot P(\overline{t}_k, c_i)$

Table 1. Main functions proposed for term space reduction purposes. Information gain is also known as *expected mutual information*; it is used under this name by Lewis [1992, page 44] and Larkey [1998]. In the χ^2 and CC formulae, g is (as usual) the cardinality of the training set. In the $RS(t_k, c_i)$ formula d is a constant damping factor.

These functions have given even better results than document frequency: Yang and Pedersen [1997] have shown that, with different classifiers and different initial corpora, sophisticated techniques such as IG or χ^2 can reduce the dimensionality of the term space by a factor of 100 with no loss (or even with a small increase) of categorisation effectiveness.

Unfortunately, the complexity of some of these information-theoretic measures does not always allow one to readily interpret why their results are so good; in other words, the rationale of the use of these measures as TSR functions is not always clear.

In this respect, Ng et al. [1997] have observed that the use of $\chi^2(t)$ for TSR purposes is counterintuitive, as squaring the numerator has the effect of equating those factors that indicate a positive correlation between the term and the category (i.e. $P(t, c_i)$ and $P(\overline{t}, \overline{c_i})$) with those that indicate a negative correlation (i.e. $P(t, \overline{c_i})$ and $P(\overline{t}, c_i)$). The "correlation coefficient" $CC(t)$ they propose, being the square root of $\chi^2(t)$, emphasises thus the former and de-emphasises the latter, thus respecting intuitions. The experimental results by Ng et al. [1997] show a superiority of $CC(t)$ over $\chi^2(t)$, but it has to be remarked that these results refer to a local, rather than global, term space reduction application.

Fuhr et al. [1998] go a further step in this direction, by observing that in $CC(t_k, c_i)$ (and *a fortiori* in $\chi^2(t_k, c_i)$)

—the $\sqrt{g}$ factor at the numerator is ininfluent, since it is equal for all pairs (t_k, c_i);

—the presence of $\sqrt{P(t_k) \cdot P(\overline{t}_k)}$ at the denominator emphasises extremely rare terms, which Yang and Pedersen [1997] have clearly shown to be the least effective

in TC.

—the presence of $\sqrt{P(c_i) \cdot P(\overline{c}_i)}$) at the denominator emphasises extremely rare categories, which is extremely counterintuitive.

Eliminating these factors from $CC(t_k, c_i)$ yields $s\chi^2(t_k, c_i)$. The use of this function for TSR purposes has been tested by Galavotti [1999] on the Reuters collection (see Section 8) and on a variety of different classifiers, and its experimental results have outperformed those obtained by means of both $\chi^2(t)$ and $CC(t)$.

As a final comment, it should be noted that the reported improvements in performance that some TSR functions achieve over others cannot be taken as general statements of the properties of these functions unless the experiments involved have been carried out in thoroughly controlled conditions and on a variety of different situations (e.g. different classifiers, different initial corpora, ...). So far, only the comparative evaluations reported in [Galavotti 1999; Yang and Pedersen 1997] seem conclusive in this respect.

5.3 Dimensionality reduction by term extraction

Given a fixed $r' \ll r$, term extraction (also known as *reparameterisation*) purports to synthesize, from the original set of r terms, a set of r' new terms that maximises the obtained effectiveness. The rationale for using synthetic (rather than naturally occurring) terms is that, due to the pervasive problems of polysemy, homonymy and synonymy, terms may not be optimal dimensions for document content representation. Methods for term extraction aim at solving these problems by creating artificial terms that do not suffer from any of the above-mentioned problems. Two approaches of this kind have been experimented in the TC literature, namely term clustering and latent semantic indexing.

5.3.1 *Term clustering.* *Term clustering* aims at grouping words with a high degree of pairwise semantic relatedness into clusters, so that the clusters (or their centroids) may be used instead of the terms as dimensions of the vector space. Term clustering is different from term selection, since the former tends to address the problem of terms that are redundant because they are *synonymous* (or near-synonymous) with other terms, while the latter targets *non-informative* terms. Any term clustering method must specify (i) a method for grouping words into clusters, and (ii) a method for converting the original representation of a document d_j into a new representation for it based on the newly synthesized dimensions.

Lewis [1992] has been the first to investigate the impact of term clustering on TC. The method he employs, called *reciprocal nearest neighbour clustering*, consists of creating clusters of two terms that are one the most similar to the other according to some measure of similarity. The results were inferior to those obtained by single-word indexing, possibly due to a disappointing performance by the clustering method: as Lewis [1992, page 48] says, "The relationships captured in the clusters are mostly accidental, rather than the systematic relationships that were hoped for."

Another example of this approach is the work of Li and Jain [1998], who view semantic relatedness between words in terms of their co-occurrence and co-absence within training documents. By using this technique in the context of a hierarchical clustering algorithm they witnessed only a marginal effectiveness improvement;

however, the small size of their experiment (see Section 6.10) hardly allows any hard conclusion to be reached.

Both [Lewis 1992; Li and Jain 1998] provide examples of *unsupervised* clustering, since the clustering activity is not guided by the category labels attached to the documents. Baker and McCallum [1998] provide instead an example of *supervised* clustering, as the *distributional clustering* method they employ clusters together those terms that tend to indicate the presence of the same category, or group of categories. Their experiments, carried out in the context of a Naïve Bayes classifier, showed only a 2% effectiveness loss with aggressivity $\frac{r}{r'} = 1000$, and even showed some effectiveness improvement with less aggressive levels of reduction.

5.3.2 *Latent semantic indexing.* *Latent semantic indexing* (LSI – [Deerwester et al. 1990]) is a technique for dimensionality reduction originally developed in the context of IR in order to address the problems deriving from the use of synonymous, near-synonymous and polysemous words as dimensions of document and query representations. This technique compresses vectors representing either documents or queries into vectors of a lower-dimensional space whose dimensions are obtained as combinations of the original dimensions by looking at their patterns of co-occurrence. In practice, LSI infers the dependence among the original terms from a corpus and "wires" this dependence into the newly obtained, independent dimensions. The function mapping original vectors into new vectors is obtained by applying a singular value decomposition to the incidence matrix formed by the original document vectors. In the context of TC, this technique is applied by deriving the mapping function from the training set and applying it to each test document so as to produce a representation for it in the lower-dimensional space.

One characteristic of LSI as a dimensionality reduction function is that the newly obtained dimensions are not, unlike the cases of term selection and term clustering, intuitively interpretable. However, they tend to work well in bringing out the "latent" semantic structure of the vocabulary used in the corpus. For instance, Schütze et al. [1995, page 235] discuss the case of classification under category `Demographic shifts in the U.S. with economic impact` by a neural network classifier. In the experiment they report, they discuss the case of a document that was indeed a positive test instance for the category, and that contained, among others, the quite revealing sentence "`The nation grew to 249.6 million people in the 1980s as more Americans left the industrial and agricultural heartlands for the South and West`". The classifier decision was incorrect when local dimensionality reduction had been performed by χ^2-based term selection retaining the top original 200 terms, but was correct when the same task was tackled by means of LSI. This well exemplifies how LSI works: the above sentence does not contain any of the 200 terms most relevant to the category selected by χ^2, but quite possibly the words contained in it have concurred to produce one or more of the LSI higher-order terms that generate the document space of the category. As Schütze et al. [1995, page 230] put it, "if there is a great number of terms which all contribute a small amount of critical information, then the combination of evidence is a major problem for a term-based classifier". A drawback of LSI, though, is that if some original term is particularly good in itself at discriminating a category, that discriminatory power may be lost in the new vector space.

Wiener et al. [1995] use LSI in two alternative ways: (i) for local dimensionality reduction, thus creating several LSI representations specific to individual categories, and (ii) for global dimensionality reduction, by creating a single LSI representation for the entire category set. Their experimental results show the former approach to perform better than the latter. Anyway, both LSI-based approaches are shown to perform better than a simple term selection technique based on the Relevancy Score measure (see Table 1).

Schütze et al. [1995] have experimentally compared LSI-based term extraction with χ^2-based term selection using three different classifier induction techniques (namely, linear discriminant analysis, logistic regression and neural networks) in a routing application. Their results showed LSI to be far more effective than χ^2 for the first two techniques, while both methods performed equally well in the case of the neural network classifier.

Other works in TC that have made use of LSI or similar term extraction techniques are [Hull 1994; Li and Jain 1998; Schütze 1998; Weigend et al. 1999; Yang 1995].

6. METHODS FOR THE INDUCTIVE CONSTRUCTION OF A CLASSIFIER

The problem of the inductive construction of a text classifier has been tackled in a variety of different ways. Here we will describe in some detail only the methods that have proven the most popular in the literature, but at the same time we will also try to mention the existence of alternative, less standard approaches.

The inductive construction of a classifier for a category $c_i \in C$ usually consists of two different phases:

(1) the definition of a function $CSV_i : D \rightarrow [0,1]$ that, given a document d_j, returns a *categorisation status value* for it, i.e. a number between 0 and 1 that, roughly speaking, represents the evidence for the fact that d_j should be classified under c_i. The CSV_i function takes up different meanings according to the different classifiers: for instance, in the "Naïve Bayes" approach discussed in Section 6.1 $CSV_i(d_j)$ is defined in terms of a probability, whereas in the "Rocchio" approach discussed in Section 6.6 $CSV_i(d_j)$ is a measure of vector closeness in r-dimensional space;

(2) the definition of a *threshold* τ_i such that $CSV_i(d_j) \geq \tau_i$ is interpreted as a decision to classify d_j under c_i, while $CSV_i(d_j) < \tau_i$ is interpreted as a decision *not* to classify d_j under c_i. A particular case occurs when the classifier already provides a binary judgement (e.g. [Apté et al. 1994; Moulinier et al. 1996]), i.e. is such that $CSV_i : D \rightarrow \{0,1\}$. In this case, the threshold is trivially any value in the (0,1) open interval.

Issue 2 will be the subject of Section 7. In this section we will instead concentrate on Issue 1, discussing a number of approaches that have been proposed in the TC literature. In this section the presentation of the algorithms will be mostly qualitative rather than quantitative, i.e. will focus on the methods for classifier induction rather than on the performance of the classifiers that can be induced by means of them. This latter will instead be tackled in Section 8.

6.1 Probabilistic classifiers

We start our presentation of the methods for classifier induction with a discussion of the *probabilistic* approach. This is a very apt start, since the very first text classifier reported in the literature [Maron 1961] was a probabilistic one.

Probabilistic classifiers view $CSV_i(d_j)$ in terms of the probability $P(c_i|d_j)$ that a document represented by a vector $d_j = \langle w_{1j}, \ldots, w_{rj} \rangle$ of (binary or weighted) terms falls within a category c_i, and attempt to compute this probability through an application of Bayes' theorem, given by

$$P(c_i|d_j) = \frac{P(c_i)P(d_j|c_i)}{P(d_j)} \tag{4}$$

In this equation, probabilities are to be interpreted on the space of documents; accordingly, $P(d_j)$ represents the probability that a randomly picked document has vector d_j as its representation, and $P(c_i)$ the probability that a randomly picked document falls within category c_i.

The estimation of $P(d_j|c_i)$ in Equation 4 is problematic, since the number of possible vectors d_j is too high (the same holds for $P(d_j)$, but for reasons that will be clear shortly this will not concern us). In order to alleviate this problem, it is common to make the assumption that any two coordinates of the document vector are, when viewed as random variables, statistically independent of each other; this *independence assumption* is encoded by the equation

$$P(d_j|c_i) = \prod_{k=1}^{r} P(w_{kj}|c_i) \tag{5}$$

Probabilistic classifiers that make use of this assumption are usually called *Naïve Bayes* classifiers, and account for most of the probabilistic approaches to TC reported in the literature[6] (see e.g. [Joachims 1998; Koller and Sahami 1997; Larkey and Croft 1996; Lewis 1992; Lewis and Gale 1994; Li and Jain 1998; Robertson and Harding 1984]). The "naïve" character of the classifier is due to the fact that usually this assumption is, quite obviously, not verified in naturally occurring document corpora.

One of the best-known Naïve Bayes approaches is the *binary independence* classifier [Robertson and Sparck Jones 1976], which results from using binary-valued vector representations for documents. In this case, if we write p_{ki} as short for $P(w_{kx} = 1|c_i)$, the $P(w_{kj}|c_i)$ factors of Equation 5 may be written as

$$P(w_{kj}|c_i) = p_{ki}^{w_{kj}}(1 - p_{ki})^{1-w_{kj}} = (\frac{p_{ki}}{1 - p_{ki}})^{w_{kj}}(1 - p_{ki}) \tag{6}$$

We may further observe that in the TC application we are considering the document space is partitioned into two categories, namely c_i and its complement $\overline{c_i}$, which are such that $P(\overline{c_i}|d_j) = 1 - P(c_i|d_j)$. If we plug in Equations 5 and 6 into Equation 4 and take logs we obtain

$$\log P(c_i|d_j) = \log P(c_i) + \tag{7}$$

[6] Cooper [1995] has pointed out that the full independence assumption of Equation 5 is not actually made in the Naïve Bayes classifier; the assumption wired into Naïve Bayes is instead the weaker *linked dependence assumption*, which may be written as $\frac{P(d_j|c_i)}{P(d_j|\overline{c_i})} = \prod_{k=1}^{r} \frac{P(w_{kj}|c_i)}{P(w_{kj}|\overline{c_i})}$.

$$\log(1-P(c_i|d_j)) = \log(1-P(c_i)) + \sum_{k=1}^{r} w_{kj}\log\frac{p_{ki}}{1-p_{ki}} + \sum_{k=1}^{r}\log(1-p_{ki}) - \log P(d_j) \sum_{k=1}^{r} w_{kj}\log\frac{p_{k\bar{i}}}{1-p_{k\bar{i}}} + \sum_{k=1}^{r}\log(1-p_{k\bar{i}}) - \log P(d_j) \quad (8)$$

where we write $p_{k\bar{i}}$ as short for $P(w_{kx}=1|\overline{c}_i)$. We may convert Equations 7 and 8 into a single equation by subtracting componentwise Equation 8 from Equation 7, thus obtaining

$$\log\frac{P(c_i|d_j)}{1-P(c_i|d_j)} = \log\frac{P(c_i)}{1-P(c_i)} + \sum_{k=1}^{r} w_{kj}\log\frac{p_{ki}(1-p_{k\bar{i}})}{p_{k\bar{i}}(1-p_{ki})} + \sum_{k=1}^{r}\log\frac{1-p_{ki}}{1-p_{k\bar{i}}} \quad (9)$$

Note that $\frac{P(c_i|d_j)}{1-P(c_i|d_j)}$ is an increasing monotonic function of $P(c_i|d_j)$, and may thus be used directly as $CSV_i(d_j)$. Note also that the factor $\log\frac{P(c_i)}{1-P(c_i)}$ and the factor $\sum_{k=1}^{r}\log\frac{1-p_{ki}}{1-p_{k\bar{i}}}$ are constant for all documents, and may therefore be disregarded[7]. Defining a classifier for category c_i thus basically requires estimating the $2r$ parameters $\{p_{1i}, p_{1\bar{i}}, \ldots, p_{ri}, p_{r\bar{i}}\}$ from the training data, which may be done in the obvious way. As a matter of fact, probabilistic classifiers are also called *parametric* classifiers, exactly because their inductive construction consists in estimating probabilistic parameters from the training data. Note that in general the classification of a given document does not require to compute a sum of r factors, as the presence of $\sum_{k=1}^{r} w_{kj}\log\frac{p_{ki}(1-p_{k\bar{i}})}{p_{k\bar{i}}(1-p_{ki})}$ would imply; in fact, all those factors for which $w_{kj}=0$ may be disregarded, and this usually accounts for the vast majority of them, since document vectors are usually very sparse.

The binary independence classifier we have just illustrated is just one of the many variants of the Naïve Bayes approach, the common denominator of which may be taken to be Equation 5. A recent paper by Lewis [1998] is an excellent roadmap on the various directions that research on Naïve Bayes classifiers has taken. Important directions that Lewis highlights are the ones aiming

—*to relax the constraint that term vectors representing documents should be binary-valued.* This looks quite natural, given that weighted indexing techniques (see e.g. [Fuhr 1989; Fuhr and Buckley 1991; Salton and Buckley 1988]) that account for the "importance" that a term t_k has to a document d_j play a key role in IR.

—*to introduce document length normalisation in the model.* In fact, it is clear from Equation 9 that the value of $\log\frac{P(c_i|d_j)}{1-P(c_i|d_j)}$ tends to be higher for long documents (i.e. documents such that $w_{kj}=1$ for many values of k), irrespectively of their semantic relatedness to c_i. Taking document length into account is easy in non-probabilistic approaches to classification (see e.g. Section 6.6), but is problematic in probabilistic ones (see [Lewis 1998, Section 5]). One possible answer is the one

[7]This is not true, however, if the "fixed thresholding" method of Section 7 is adopted. In fact, in this case the k aptest categories for document d_j are chosen. This means that for a fixed document d_j the first and third factor in the formula above are different for different categories, and may therefore influence the choice of the categories under which to file d_j.

adopted by Baker and McCallum [1998], who switch from an interpretation of Naïve Bayes in which documents are events, to one in which terms are events. While in the former case we have multiple binomial random variables (standing for terms), in the latter case we have a single multinomial random variable (standing for documents). This accounts for document length naturally but, as noted in [Lewis 1998], has the drawback that different occurrences of the same word within the same document are viewed as independent, an assumption even more strikingly implausible that the standard word independence assumption. A similar solution, and suffering from the same problem, had already been proposed by Guthrie et al. [1994].

—*to relax the independence assumption.* This may be the hardest route to follow, since this inevitably produces classifiers of higher computational cost and characterised by harder parameter estimation problems. Earlier efforts in this direction within the field of probabilistic IR (e.g. [van Rijsbergen 1977]) have not shown the performance improvements that were hoped for. Recently, the fact that the binary independence assumption seldom harms categorisation effectiveness has also been given some theoretical justification [Domingos and Pazzani 1997].

The quotation of IR research in the last paragraph is not casual. Unlike other types of classifiers, the literature on probabilistic classifiers is inextricably intertwined with that on probabilistic IR (see [Crestani et al. 1998] for a review of this), since this latter may be seen as the attempt to determine the probability that a document falls in the category denoted by the query, and since it is the only approach to IR that takes *relevance feedback*, a notion essentially involving supervised learning, as central to its models.

A still controversial point in the application of Naïve Bayes to TC is whether term selection can be applied without inducing a degradation in the performance of the classifier. Li and Jain [1998] have reported a disastrous decrease in performance when term space reduction based on mutual information was applied to a Naïve Bayes classifier; in the same study, an application of the same reduction technique did not degrade effectiveness when applied to k-NN or decision tree classifiers. In another study, Joachims [1998] has shown that a Naïve Bayes classifier working with a term space consisting of the terms ranked *lowest* by an information gain measure still performed way better than a random classifier. Rather than showing that even the most uninformative words are useful *in text categorisation*, as Joachims concluded, this may well indicate that even the least informative terms are useful *to a Naïve Bayes classifier*. The work of Baker and McCallum [1998] discussed in Section 5.3.1 seems however to contradict this hypothesis, and the same may be said of the work of Lewis and Ringuette [1994]. It seems safe to conclude that more systematic experiments are needed to say something conclusive on this issue.

For a more thorough discussion on Naïve Bayes classifiers and their variations, see the excellent [Lewis 1998].

6.2 Decision tree classifiers

Probabilistic induction methods are essentially quantitative (i.e. numeric) in nature, and as such have sometimes been criticised since, effective as they may be, are not readily interpretable by humans. A class of algorithms that do not suffer from this

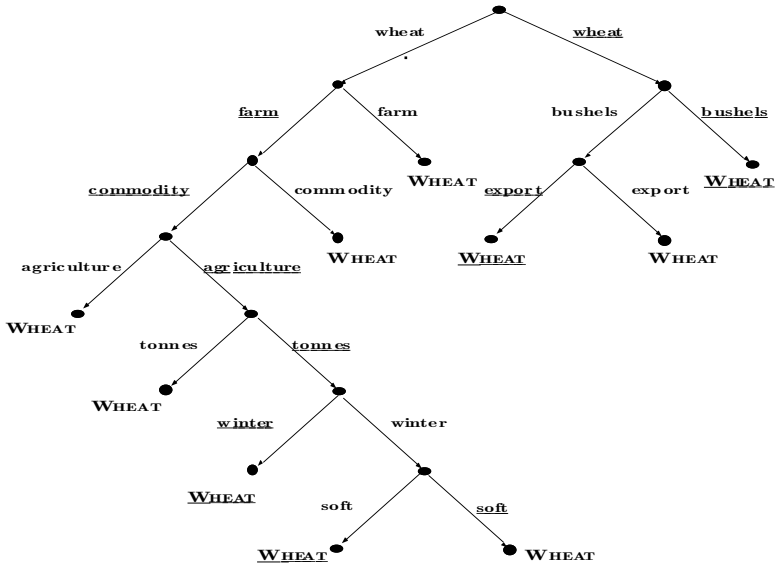

Fig. 3. A decision tree equivalent to the DNF rule of Figure 1. Edges are labelled by terms and leaves are labelled by categories (underlining denotes negation).

problem are *symbolic* (i.e. non-numeric) algorithms, among which inductive rule learners (which we will discuss in Section 6.3) and decision tree inducers are the most important examples.

A *decision tree* text classifier (see e.g. [Mitchell 1996, Section 3]) consists of a tree in which internal nodes are labelled by terms, branches departing from them are labelled by tests on the weight that the term has in the representation of the test document, and leaf nodes are labelled by (not necessarily different) categories. Such a classifier categorises a test document d_j by recursively testing for the weights that the terms labeling the internal nodes have in the representation of d_j, until a leaf node is reached; the label of this leaf node is then assigned to d_j. Most such text classifiers assume a binary document representation, and thus consist of binary trees. An example of such a tree is illustrated in Figure 3.

There are a number of standard packages around for the induction of a decision tree from a training set, and most decision tree approaches to TC have made use of one such package. Among the most popular packages are ID3 (used in [Fuhr et al. 1991]), C4.5 (used in [Cohen and Hirsch 1998; Cohen and Singer 1999; Joachims 1998; Lewis and Catlett 1994]) and C5 (used in [Li and Jain 1998]). Other TC endeavours based on experimental decision tree packages include [Dumais et al. 1998; Lewis and Ringuette 1994; Weiss et al. 1999].

A possible procedure for the induction of a decision tree for category c_i from a set of training examples consists in a "divide and conquer" strategy of recursively (i) checking whether all the training examples have the same label (either c_i or $\overline{c}_i$); (ii) if not, selecting a term t_k, partitioning the training examples into classes of documents that have the same value for t_k, and placing each such class in a separate subtree. The process is recursively repeated on the subtrees until each leaf node of

the tree so generated contain training examples assigned to the same category c_i, which is then chosen as the label for the leaf node. The key step of this process is the choice of the term t_k on which to operate the partition, a choice which is generally made according to an information gain or entropy criterion. However, such a "fully grown" tree may be prone to overfitting, as some branches may be excessively specific to the training data. Any decision tree induction method thus includes a method for growing the tree and one for pruning it, i.e. for removing the overly specific branches so as to minimise the probability of misclassifying test documents. Variations on this basic schema for tree induction abound; the interested reader is referred to [Mitchell 1996, Section 3].

Decision tree text classifiers have been used either as the main classification tool [Fuhr et al. 1991; Lewis and Catlett 1994; Lewis and Ringuette 1994], or as baseline classifiers [Cohen and Singer 1999; Joachims 1998], or as members of classifier committees [Li and Jain 1998; Schapire et al. 1998; Schapire and Singer 2000; Weiss et al. 1999].

6.2.1 *The AIR/X project.* Within the decision trees TC literature, a special place is occupied by the AIR/X system [Fuhr et al. 1991]. This system is important since it constitutes the final result of the AIR project, one of the most important endeavours in the history of TC. The AIR project, spanning a duration of more than ten years [Fuhr and Knorz 1984; Fuhr 1985; Biebricher et al. 1988; Fuhr et al. 1991], has produced a system operatively employed in the classification of corpora of scientific literature of more than a million documents, and has had important theoretical spin-offs in the field of probabilistic indexing [Fuhr 1989; Fuhr and Buckley 1991][8].

The approach to TC taken in AIR/X is known as the *Darmstadt Indexing Approach* (DIA). Here, "indexing' is meant in the sense of Section 3.1, i.e. as using terms from a controlled vocabulary, and is thus a synonym of TC (the DIA was later extended to indexing with a non-controlled vocabulary [Fuhr and Buckley 1991]). The DIA is based on the preliminary computation of *association factors* $z(t_k, c_i)$ between free text terms t_k and categories c_i, where $z(t_k, c_i) = \frac{P(t_k, c_i)}{P(t_k)}$ is the proportion of training documents containing t_k that are classified under c_i (association factors are called *adhesion coefficients* in many early papers on TC – see e.g. [Field 1975; Robertson and Harding 1984]). Once computed these association factors, the DIA trains a decision tree in two steps. In the *description step*, for every occurrence t^x_{kj} of term t_k in training document $\overline{d}_j$, a *relevance description* $rd(c_i, \overline{d}_j)$ is updated by using $z(t_k, c_i)$ and the characteristics of occurrence t^x_{kj} (e.g. the section of $\overline{d}_j$ to which t^x_{kj} belongs, etc.). In the *decision step*, the relevance description $rd(c_i, \overline{d}_j)$ is transformed into a discrete-valued vector $\vec{rd}(c_i, \overline{d}_j)$. At this point, the ID3 decision tree algorithm is invoked, which, by selecting (based on a χ^2 criterion) one attribute of the vector representation at a time, partitions training vectors into equivalence classes of identical vectors.

[8]The AIR/X system, its applications (including the AIR/PHYS system [Biebricher et al. 1988], an application of AIR/X to indexing physics literature), and its experiments, have also been richly documented in a series of papers and doctoral theses written in German. The interested reader may consult [Fuhr et al. 1991] for a detailed bibliography.

For classifying a test document d_l, the description and decision steps are applied to d_l. The probability of the correctness of attributing c_i to d_l is identified with the percentage of training vectors $\vec{rd}(c_i, \overline{d}_j)$ assigned to the equivalence class of $\vec{rd}(c_i, d_l)$ that correspond to correct categorisation decisions.

6.3 Decision rule classifiers

A classifier for category c_i built by an *inductive rule learning* method consists of a disjunctive normal form (DNF) rule, i.e. of a conjunction of conditional formulae ("clauses") of the type of those manually built into the CONSTRUE system and illustrated in Figure 1. Clause premises denote the presence or absence of terms in the test document, while the clause head denotes the decision whether to classify it or not under c_i. DNF induction methods are known to be of equal power to decision tree methods from machine learning theory. However, one of their advantages is that they tend to generate more compact classifiers than decision tree inducers.

Rule induction methods usually attempt to select from all the possible covering rules (i.e. those rules that correctly classify all the training examples) the "best" one according to some minimality criterion. While decision trees are typically induced through a top-down, "divide-and-conquer" strategy, DNF rules are often induced in a bottom-up fashion. At the start of the induction of the classifier for c_i, every training example is viewed as a clause $\tau_1, \ldots, \tau_n \rightarrow \gamma_i$, where $\tau_1, \ldots, \tau_n$ are the terms contained in the document and γ_i equals c_i or $\overline{c}_i$ according to whether the document is a positive or negative example of c_i. This set of clauses is already a DNF classifier for c_i, but obviously scores tremendously high in terms of overfitting. The induction algorithm employs then a process of generalisation whereby the rule is simplified through a series of modifications (e.g. removing premises from clauses, or merging clauses) that maximise its compactness while at the same time not affecting the "covering" property of the classifier. At the end of this process, a "pruning" phase similar in spirit to that employed in decision trees is applied, where the ability to correctly classify *all* the training examples is traded for more generality.

Individual DNF rule learners vary widely in terms of the methods, heuristics and criteria employed for generalisation and for pruning. Among the inductive DNF rule learners that have been applied to TC are CHARADE [Moulinier and Ganascia 1996], DL-ESC [Li and Yamanishi 1999], RIPPER [Cohen 1995a; Cohen and Hirsch 1998; Cohen and Singer 1999], SCAR [Moulinier et al. 1996], and SWAP-1 [Apté et al. 1994].

It should be mentioned that, while the DNF induction methods mentioned above deal with rules at the propositional logic level, research has also been carried out on using rules of first order logic, obtainable through the use of *inductive logic programming* methods. Cohen [1995a] has extensively compared propositional and first order induction in a TC application (for instance, comparing the propositional learner RIPPER with its first order version FLIPPER), and has concluded that the additional representational power of first order logic brings about only modest benefits.

6.4 Regression models

Various TC endeavours have made use of regression models (see e.g. [Ittner et al. 1995; Lewis and Gale 1994; Schütze et al. 1995]). In the statistical learning community, *regression* refers to the problem of approximating a real-valued function f by means of a function $\hat{f}$ that fits the training data [Mitchell 1996, page 236]. Here we will describe one such model, the *Linear Least Squares Fit* (LLSF) proposed by Yang and Chute [1994]. In LLSF, each document d_j has two vectors associated to it: an *input vector* $I(d_j)$, i.e. a standard vector of r weighted terms, and an *output vector* $O(d_j)$, consisting of a vector of m weights, representing the categories (the weights for this latter vector are binary in the case of training documents, are standard non-binary CSVs in the case of test documents). Classification may thus be seen as the task of determining an output vector $O(d_j)$ for test document d_j, given its input vector $I(d_j)$; hence, building a classifier boils down to computing an $m \times r$ matrix $\hat{M}$ such that $I(d_j)\hat{M} = O(d_j)$. LLSF computes the matrix from the training data by computing a linear least-squares fit that minimizes the error on the training set according to the formula

$$\hat{M} \;=\; \arg\min_{M} \|MI - O\|_F$$

where $\arg\min_{M}(x)$ stands as usual for the M for which the argument x is minimum, $\|V\|_F \stackrel{def}{=} \sum_{i=1}^{m}\sum_{j=1}^{n} v_{ij}$ represents the so-called *Frobenius norm* of an $m \times n$ matrix, I is the $r \times g$ matrix whose columns are the input vectors of the training documents, and O is the $m \times g$ matrix whose columns are the output vectors of the training documents. The $\hat{M}$ matrix is usually computed by performing a singular value decomposition on the training set. In the resulting matrix $\hat{M}$ the generic entry $\hat{m}_{ik}$ represents the degree of association between category c_i and term t_k.

Experimentation has shown that LLSF is one of the most effective text classifiers known to date. One of its disadvantages, though, is that the computational cost of computing the $\hat{M}$ matrix is much higher than that of many other competitors in the TC arena.

6.5 On-line linear classifiers

A *linear classifier* is a representation of the category of interest in terms of a vector $c_i = \langle w_{1i}, \ldots, w_{ri}\rangle$ belonging to the same r-dimensional space in which documents are also represented, and such that $CSV_i(d_j)$ corresponds to the inner product $\sum_{k=1}^{r} w_{ki} \cdot w_{kj}$ of the document vector and the category vector. It is interesting to notice that when both classifier and document weights are cosine-normalised (see Equation 3), the inner product between the two vectors corresponds to their *cosine similarity*, i.e.

$$S(c_i, d_j) \;=\; cos(\alpha) = \frac{\sum_{k=1}^{r} w_{ki} \cdot w_{jk}}{\sqrt{\sum_{k=1}^{r} w_{ki}^2} \cdot \sqrt{\sum_{k=1}^{r} w_{kj}^2}}$$

which represents the cosine of the angle α that separates the two vectors. The interest lies in the fact that this is the similarity measure between query and document computed by standard vector-space IR engines, which means in turn that once a linear classifier has been induced, classification can be performed by invoking such

a standard IR engine.

Linear classifiers are also called *profile-based classifiers*, as they rely on the extraction of an explicit *profile* (or prototypical document) of the category from the training set. This has obvious advantages in terms of interpretability, as such a profile is a representation more readily interpretable by a human than, say, a neural network classifier. Linear classifiers are often partitioned in two broad classes, batch classifiers and on-line classifiers.

Batch induction methods build a classifier by analysing the training set all at once. Within the TC literature, one example of a batch induction method is *linear discriminant analysis*, a model of the stochastic dependence between terms that relies on the covariance matrices of the various categories [Blosseville et al. 1992; Hull 1994; Schütze et al. 1995]. However, the foremost example of a batch linear classifier is the Rocchio classifier; because of its importance in the TC literature this will be discussed separately in Section 6.6. In this section we will instead concentrate on on-line classifiers.

On-line (aka *incremental*) *induction methods* build a classifier soon after examining the first training document, and incrementally refine it as they examine new ones. This may be an advantage in those TC applications in which the training set is not available in its entirety right from the start, or in which the "meaning" of the category may change in time, as e.g. in adaptive filtering. This is also extremely apt to those applications in which we may expect the user of a classifier to provide feedback on how test documents have been classified, as in this case further training may be performed during the operating phase by exploiting user feedback. Applications of this kind may be interactive classification ([Larkey and Croft 1996] – see also Section 7) or, again, adaptive filtering.

A simple example of an on-line method is the *perceptron* algorithm, first proposed for TC applications in [Schütze et al. 1995; Wiener et al. 1995] and subsequently experimented with in [Dagan et al. 1997; Ng et al. 1997]. In this algorithm, the classifier for c_i is first initialised by setting all weights w_{ki} to the same positive value. When a training example $\overline{d}_j$ (represented by a vector of binary weights) is examined, the classifier built so far attempts to classify it, and the result of the classification is examined. If this is correct nothing is done, while if this is wrong the weights of the classifier are modified: if $\overline{d}_j$ was a positive example of c_i then the weights w_{ki} of "active terms" (i.e. those terms t_k such that $w_{kj} = 1$) are promoted, by increasing them by a fixed quantity $\alpha > 0$ (called *learning rate*), while if $\overline{d}_j$ was a negative example of c_i then the same weights are demoted, by decreasing them by α. Note that when at a certain stage of the training phase the classifier has reached a reasonable level of effectiveness, the fact that a weight w_{ki} is very low means that t_k has negatively contributed to the classification behaviour so far, and may thus be discarded from the representation. We may then see the perceptron algorithm (as all other incremental induction methods, for that matter) as allowing for a sort of "on-the-fly term space reduction" [Dagan et al. 1997, Section 4.4]. The perceptron classifier has shown a good effectiveness in all the experiments quoted above.

The perceptron algorithm is an *additive weight-updating* algorithm. A *multiplicative* variant of the perceptron is Positive Winnow [Dagan et al. 1997], which differs from perceptron because two different constants $\alpha_1 > 1$ and $0 < \alpha_2 < 1$ are

used for promoting and demoting weights, respectively, and because promotion and demotion are achieved by multiplying, instead of adding, by α_1 and α_2. BALANCED WINNOW [Dagan et al. 1997; Ragas and Koster 1998] is a further variant of POSITIVE WINNOW, in which the classifier consists of *two* weights w_{ki}^+ and w_{ki}^- for each term t_k; the final weight w_{ki} used in computing the inner product is the difference $w_{ki}^+ - w_{ki}^-$. Following the misclassification of a positive instance, active terms have their w_{ki}^+ weight promoted and their w_{ki}^- weight demoted, whereas in the case of a negative instance it is w_{ki}^+ that gets promoted while w_{ki}^- gets demoted (for the rest, promotions and demotions are as in POSITIVE WINNOW). BALANCED WINNOW allows negative w_{ki} weights, while in the perceptron algorithm and in POSITIVE WINNOW the w_{ki} weights are always positive.

In experiments conducted by Dagan et al. [1997], POSITIVE WINNOW showed a better effectiveness than the perceptron algorithm but was in turn outperformed by (Dagan et al.'s own version of) BALANCED WINNOW. These experimental results confirm various analytical intuitions concerning these algorithms that are discussed in the theoretical learning literature.

Other examples of on-line classifiers are the WIDROW-HOFF classifier, a refinement of it called the EXPONENTIATED GRADIENT classifier (both applied for the first time to TC in [Lewis et al. 1996]) and the SLEEPING EXPERTS algorithm [Cohen and Singer 1999; Ragas and Koster 1998], a version of BALANCED WINNOW. While the first is an additive weight-updating algorithm, the second and third are multiplicative. Key differences with the previously described algorithms are that these three algorithms (i) update the classifier not only after misclassifying a training example, but also after classifying it correctly, and (ii) update the weights corresponding to all terms (instead of just active ones).

Linear classifiers lend themselves to both category-pivoted and document-pivoted TC. In the former case, the classifier c_i is used as a query against the set of test documents $\{d_1, \ldots, d_n\}$, while in the latter case the test document d_j is used as a query against the set of classifiers $\{c_1, \ldots, c_m\}$.

Finally, it is worth noticing that the induction of a linear classifier may be typically preceded by local term space reduction, i.e. the r' most important terms for category c_i are selected according to some measure. For this, the c_i-variants of the functions illustrated in Table 1 are usually employed, where by "c_i-variant" of a TSR measure we mean a measure computed on the positive training examples of category c_i rather than on the entire training set.

6.6 The Rocchio classifier

The *Rocchio classifier* relies on an adaptation to the TC case of Rocchio's formula for relevance feedback in the vector-space model, and it is perhaps the only TC method whose roots lie exclusively in the IR tradition rather than in the machine learning one. This adaptation was first proposed by Hull [1994]; since then, the Rocchio classifier has been used by many authors, either as an object of research in its own right [Ittner et al. 1995; Joachims 1997; Ragas and Koster 1998; Sable and Hatzivassiloglou 1999; Schapire et al. 1998; Singhal et al. 1997], or as a baseline classifier [Cohen and Singer 1999; Fuhr et al. 1998; Galavotti 1999; Joachims 1998; Lewis et al. 1996; Schapire and Singer 2000; Schütze et al. 1995], or as a member of a classifier committee [Larkey and Croft 1996] (see Section 6.10).

Rocchio's method computes a classifier $\langle w_{1i}, \ldots, w_{ri} \rangle$ for category c_i by means of the formula

$$w_{ki} = \left(\frac{\beta}{| \{\overline{d}_j \mid ca_{ij} = 1\} |} \cdot \sum_{\{\overline{d}_j \mid ca_{ij}=1\}} w_{kj} \right) - \left(\frac{\gamma}{| \{\overline{d}_j \mid ca_{ij} = 0\} |} \cdot \sum_{\{\overline{d}_j \mid ca_{ij}=0\}} w_{kj} \right)$$

where w_{kj} is the weight that term t_k has in document $\overline{d}_j$. In this formula, β and γ are control parameters that allow setting the relative importance of positive and negative examples. For instance, if β is set to 1 and γ to 0 (as e.g. in [Dumais et al. 1998; Hull 1994; Joachims 1998; Schütze et al. 1995]), this corresponds to viewing the profile of c_i as the *centroid* of its positive training examples. In general, the Rocchio classifier rewards the closeness of a test document to the centroid of the positive training examples, and its distance from the centroid of the negative training examples. Most of the times the role of negative examples is de-emphasised, by setting β to a high value and γ to a low one (e.g. Cohen and Singer [1999], Ittner et al. [1995], and Joachims [1997] use $\beta = 16$ and $\gamma = 4$).

This method is quite easy to implement, and the resulting classifiers tend to be quite efficient [Schapire et al. 1998]. In terms of effectiveness, instead, one of its drawbacks is that if the documents in the category tend to occur in disjoint clusters (e.g. a set of newspaper articles falling under the `Sports` category and dealing with either boxing or rock-climbing), the Rocchio classifier may miss most of them, as the centroid of these documents may fall well outside all of these clusters (see Figure 4a). More generally, the Rocchio classifier, as all linear classifiers, has the disadvantage that it basically divides the space of documents in two subspaces; any document falling within the former (in the case of Rocchio, an n-sphere) will be classified under c_i, while all documents falling within the latter will not. This situation is graphically depicted in Figure 4a, where documents are classified within c_i if and only if they fall within the circle. Note that even most of the positive training examples would not be classified correctly by the classifier. This is due to the fact that what Rocchio basically does is taking the average (centroid) of all positive examples, and as all averages this is only partly representative of the whole set.

6.6.1 *Enhancements to the basic Rocchio framework.* One issue in the application of the Rocchio formula to profile extraction is whether the set of negative training instances $\{\overline{d}_j \in Tr \mid ca_{ij} = 0\}$ should be considered in its entirety, or whether a well-chosen sample of it, such as the set of *near-positives* (defined as "the most positive amongst the negative training examples"), should be selected. In this latter case, the contribution of the $\frac{\gamma}{|\{\overline{d}_j \mid ca_{ij}=0\}|} \cdot \sum_{\{\overline{d}_j \mid ca_{ij}=0\}} w_{kj}$ factor tends to be more significant, since near-positives are the most difficult documents to tell apart from the relevant documents. Using near-positives corresponds to the *query zoning* method proposed for IR by Singhal et al. [1997]. This method originates from the observation that when the original Rocchio formula is used for relevance feedback in IR, near-positives tend to be used rather than generic negatives, as the documents on which user judgments are available tend to be the ones that had scored highest in the previous ranking. Early applications of the Rocchio formula to TC (e.g. [Ittner et al. 1995]) generally did not make a distinction between near-positives and generic negatives. Schapire et al. [1998] individuate the near-positives by issuing a

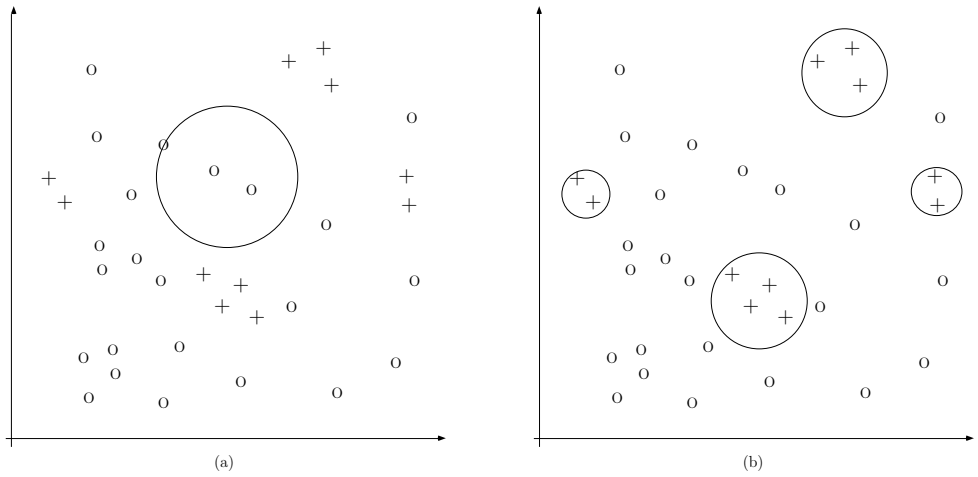

Fig. 4. A comparison between the categorisation behaviour of (a) the Rocchio classifier, and (b) the k-NN classifier. Small crosses and circles denote positive and negative training instances, respectively. The big circles denote the "influence area" of the classifier.

Rocchio query consisting of the centroid of the positive training examples against a document base consisting of the negative training examples; the top-ranked ones are the most similar to this centroid, and can then be used as near-positives. Fuhr et al. [1998], instead, identify the near-positives of category c_i with the positive examples of the *sibling* categories of c_i, as in the application they work on (Web page categorisation into hierarchical catalogues – see also Section 9) the notion of a "sibling category of c_i" is well-defined. Similar policies are also adopted in [Ng et al. 1997; Ruiz and Srinivasan 1999].

By using the query zoning method plus other enhancements (term selection, statistical phrases, and a method called *dynamic feedback optimisation*), Schapire et al. [1998] have experimentally shown that a Rocchio classifier can achieve levels of effectiveness comparable to those of a state-of-the-art machine learning method such as "boosting" (see Section 6.10.1) while being 60 times quicker to train. These recent results will no doubt bring about a renewed interest for the Rocchio classifier, previously a favourite punching bag of more sophisticated learning methods [Cohen and Singer 1999; Joachims 1998; Lewis et al. 1996; Schütze et al. 1995; Yang 1999].

6.7 Neural networks

A *neural network* classifier is a network of units, where the input units usually represent terms, the output unit(s) represent the category or categories of interest, and the weights on the edges that connect units represent conditional dependence relations. For classifying a test document d_j, its term weights w_{kj} are assigned to the input units; the activation of these units is propagated forward through the network, and the value that the output unit(s) take up as a consequence determines the categorisation decision(s). A typical way of training neural networks is backpropagation, whereby the term weights of a training document are loaded into the input units, and if a misclassification occurs the error is "backpropagated" so

as to change the parameters of the network and eliminate or minimise the error.

The simplest type of neural network classifier is the perceptron [Dagan et al. 1997; Ng et al. 1997], which is a linear classifier and as such has been extensively discussed in Section 6.5. Other types of linear neural network classifiers implementing a form of logistic regression have also been proposed and experimented with by Schütze et al. [1995] and Wiener et al. [1995], and they have usually given very good effectiveness results.

A non-linear neural network [Ruiz and Srinivasan 1999; Schütze et al. 1995; Weigend et al. 1999; Wiener et al. 1995; Yang and Liu 1999] is instead a network with one or more additional "layers" of units, which in TC usually represent higher-order interactions between terms that the network is able to learn. When comparative experiments relating non-linear neural networks to their linear counterparts have been performed, the former have yielded either no improvement [Schütze et al. 1995] or very small improvements [Wiener et al. 1995] over the latter.

6.8 Example-based classifiers

Example-based classifiers do not build an explicit, declarative representation of the category of interest, but "parasite" on the categorisation judgments that the experts have given on the training documents similar to the test document. These methods have thus been called *lazy* learning systems, since "they defer the decision on how to generalize beyond the training data until each new query instance is encountered" [Mitchell 1996, pag 244].

The first introduction of example-based methods (sometimes also called *memory-based reasoning methods*) in the TC literature is due to Creecy, Masand and their colleagues [Creecy et al. 1992; Masand et al. 1992]; other TC endeavours in which these methods have been employed include [Joachims 1998; Lam et al. 1999; Larkey 1998; Larkey 1999; Li and Jain 1998; Yang and Pedersen 1997; Yang and Liu 1999]. Our presentation of the example-based approach will be based on the *k-NN* (for "*k* nearest neighbours") algorithm implemented by Yang [1994] in the ExpNet system. For deciding whether d_j should be classified under c_i, k-NN looks at whether the k training documents most similar to d_j have also been classified under c_i; if the answer is positive for a large enough proportion of them, a positive categorisation decision is taken, and a negative decision is taken otherwise.

Actually, Yang's is a *distance-weighted* version of k-NN (see e.g. [Mitchell 1996, Section 8.2.1]), since the fact that a most similar document has been classified under c_i is weighted by its similarity with the test document. Mathematically, classifying a document by means of k-NN comes down to computing

$$CSV_i(d_j) = \sum_{\overline{d}_z \in Tr_k(d_j)} RSV(d_j, \overline{d}_z) \cdot ca_{iz} \tag{10}$$

where $Tr_k(d_j)$ is the set of the k documents $\overline{d}_z$ for which $RSV(d_j, \overline{d}_z)$ is maximum and the ca_{iz} values are from the correct decision matrix of Section 4.1. In turn, $RSV(d_j, \overline{d}_z)$ represents some measure or semantic relatedness between a test document d_j and a training document $\overline{d}_z$; any matching function, be it probabilistic (as used in [Larkey and Croft 1996]) or vector-based (as used in [Yang 1994]), from a ranked IR system may be used for this purpose.

The k-NN method may also be represented graphically as in Figure 5. From this

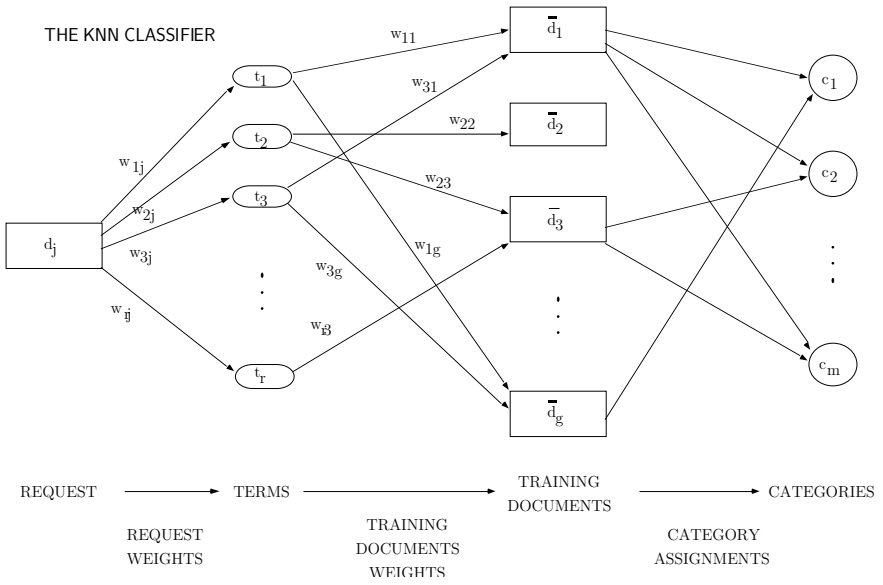

Fig. 5. A graphical representation of the k-NN method. Node d_j has weight equal to 1. Weights flow from left to right and get multiplied by the weights of the edges through which they flow; weights incoming into the same node are summed together. The weight that node c_i receives as a result of the process is the value of $CSV_i(d_j)$.

figure, it is quite evident that this approach is naturally geared towards document-pivoted categorisation. In theory, for category-pivoted-categorisation one would only need to "inhibit" the arcs from all training documents to categories c_1, ..., c_{i-1}, c_{i+1}, ..., c_m. The resulting network is a classifier for category c_i only (the original network can instead be seen as running the classifiers for all categories in parallel). One would need to run all documents through this network, one at a time. In practice, though, this would be unacceptably inefficient, since the entire training set would have to be re-ranked m times, one for each category; DPC is thus *de facto* the only reasonable way to use k-NN.

The construction of a k-NN classifier also involves determining a threshold k, indicating how many top-ranked training documents have to be considered for computing $CSV_i(d_j)$. This threshold is usually determined experimentally on a validation set. For instance, Larkey and Croft [1996] use $k = 20$, while Yang [1994, 1999] has found $30 \leq k \leq 45$ to yield the best effectiveness. Anyhow, various experiments have shown that increasing the value of k does not significantly degrade the performance.

Note that k-NN, unlike linear classifiers, does not subdivide the document space in just two subspaces, hence it does not suffer from the problem discussed at the end of Section 6.6. This is graphically depicted in Figure 4b, where the more "local" character of k-NN with respect to Rocchio can be appreciated.

Besides its remarkable effectiveness, which has been proven through a number of different experiments (see Section 8.3), one of the advantages of k-NN is its efficiency, as the classification of a document in the m categories of interest can

be performed in time linear in the cardinality g of the training set [Yang 1999]. Nevertheless, it has to be remarked that "lazy" learning methods like k-NN are less efficient than "eager" methods at classification time, since they do not have a training phase and thus perform all the computation at classification time.

6.8.1 *Other example-based techniques.* Various nearest neighbour techniques have been used in the TC literature.

Cohen and Hirsch [1998] implement an example-based classifier by extending standard relational DBMS technology with "similarity-based soft joins". In their WHIRL system they use the scoring function

$$CSV_i(d_j) = 1 - \prod_{\overline{d}_z \in Tr_k(d_j)} (1 - RSV(d_j, \overline{d}_z)) \cdot ca_{iz}$$

as an alternative to Equation 10, obtaining a small but statistically significant improvement over a version using Equation 10. Their experiments show that this technique outperforms a number of other classifiers, such as the C4.5 decision tree classifier and the RIPPER DNF rule-based classifier.

An interesting variant of the basic k-NN approach is proposed by Galavotti [1999], who reinterprets Equation 10 by redefining ca_{iz} as

$$ca_{iz} = \begin{cases} 1 & \text{if } \overline{d}_z \text{ is a positive example of } c_i \\ -1 & \text{if } \overline{d}_z \text{ is a negative example of } c_i \end{cases}$$

The difference with the original k-NN approach is that if a training document $\overline{d}_z$ similar to the test document d_j does not belong to c_i, this information is not discarded but considered as negative evidence, i.e. weights *negatively* in the decision to classify d_j under c_i.

A combination of profile- and example-based methods is presented in [Lam and Ho 1998]. In this work a k-NN system is fed, in place of training documents, what the authors call *generalised instances* (GIs). This approach may be seen as the result of

—clustering the training set, thus obtaining a set of clusters $CL_i = \{cl_{i1}, \ldots, cl_{ik_i}\}$;
—inducing a linear classifier $lc(cl_{iz})$ ("generalised instance") from the documents belonging to cluster cl_{iz} with any of the algorithms discussed in Section 6.5 and 6.6;
—applying k-NN with linear classifiers in place of training documents, i.e. computing

$$\begin{aligned} CSV_i(d_j) &\stackrel{def}{=} \sum_{cl_{iz} \in CL_i} RSV(d_j, lc(cl_{iz})) \cdot \frac{|\{\overline{d}_j \in cl_{iz} |\ ca_{ij} = 1\}|}{|\{\overline{d}_j \in cl_{iz}\}|} \cdot \frac{|\{\overline{d}_j \in cl_{iz}\}|}{|Tr|} \\ &= \sum_{cl_{iz} \in CL_i} RSV(d_j, lc(cl_{iz})) \cdot \frac{|\{\overline{d}_j \in cl_{iz} |\ ca_{ij} = 1\}|}{|Tr|} \end{aligned}$$

where $\frac{|\{\overline{d}_j \in cl_{iz} |\ ca_{ij}=1\}|}{|\{\overline{d}_j \in cl_{iz}\}|}$ represents the "degree" to which the generalised instance $lc(cl_{iz})$ is a positive instance of c_i, and $\frac{|\{\overline{d}_j \in cl_{iz}\}|}{|Tr|}$ represents its weight within the whole process.

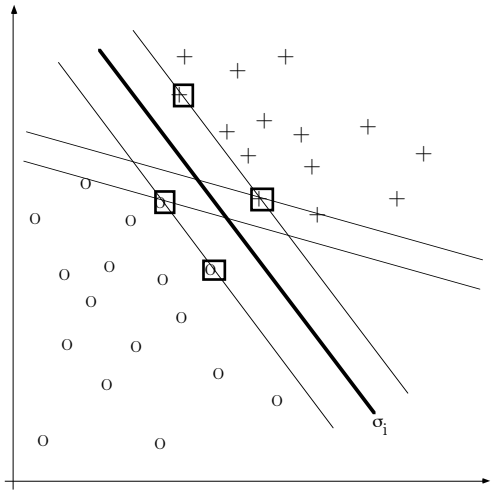

Fig. 6. The induction of support vector classifiers. The small crosses and circles represent positive and negative training examples, respectively, whereas lines represent decision surfaces. Decision surface σ_i (indicated by the thicker line) is, among those shown, the best possible one, as it is the middle element of the widest set of parallel decision surfaces (i.e. its minimum distance to any training example is maximum). Small boxes indicate the support vectors.

This exploits the superior effectiveness (graphically illustrated in Figure 4) of k-NN over linear classifiers while at the same time avoiding the sensitivity of k-NN to the presence of "outliers" (i.e. positive instances of c_i that "lie out" of the region of document space where most other positive instances of c_i are located) in the training documents.

6.9 Building classifiers by support vector machines

The application of the *support vector machine* method to TC has been recently proposed by Joachims [1998, 1999], and subsequently used in [Dumais et al. 1998; Taira and Haruno 1999; Yang and Liu 1999]. In geometrical terms, this method may be seen as the attempt to find, among all the surfaces $\sigma_1, \sigma_2, \ldots$ in r-dimensional space that separate the positive from the negative training examples (*decision surfaces*), the surface σ_i that does it in the best possible way. "Best" here means that σ_i separates the positives from the negatives by the widest possible margin, i.e. the separation property is invariant with respect to the widest possible shift of σ_i yielding a parallel surface σ_i'. This method is a practical application of the so-called *structural risk minimization principle*, according to which the decision surface should minimise *true error*, i.e. the probability of misclassification of a randomly selected, yet unseen test example.

This concept is best understood in the case in which the positives and the negatives are linearly separable, in which case the decision surfaces are $(r-1)$-hyperplanes. In the 2-dimensional case of Figure 6, various sets of parallel lines may be chosen as decision surfaces. The support vector method chooses the "widest" set of parallel lines (and from this its middle line), i.e. the one in which the max-

imum distance between two elements in the set is highest[9]. It is noteworthy that this "best" decision surface is determined by only a small set of training examples, called the *support vectors*.

The method described is applicable also to the case in which the positive and the negative examples are not linearly separable. Yang and Liu [1999] experimentally compared the linear case (namely, when the assumption is made that the categories are linearly separable) with the non-linear case on a standard benchmark, and obtained slightly better results in the former case.

According to Joachims [1998], support vector machines offer two important advantages for TC:

—no term selection is needed, as support vector machines do not suffer from overfitting and can scale up to considerable dimensionalities;

—no human and machine effort in parameter tuning on a validation set is needed, as there is a theoretically motivated, "default" choice of parameter settings, which has also been shown to provide the best effectiveness.

Dumais et al. [1998] have recently tested a novel algorithm for training SVM text classifiers and shown that this brings about training speeds comparable to computationally easy methods such as Rocchio.

6.10 Classifier committees

The method of classifier *committees* (or *ensembles*) is based on the idea that, given a task that requires expert knowledge to be performed, k experts may be better than one if their individual judgments are appropriately combined. In TC, the idea is to apply k different classifiers $\Phi_1, \ldots, \Phi_k$ to the same task of deciding whether document d_j should be classified under category c_i, and then combine their outcome appropriately. Such a classifier committee is then characterised by (i) a choice of k classifiers, and (ii) a choice of a combination function[10].

Concerning the former issue, it is well-known from the machine learning literature that, in order to guarantee good effectiveness, the classifiers forming the committee should be as independent as possible, i.e. should be possibly based on radically different intuitions on how classification is to be performed. The classifiers may be different in terms of the indexing approach followed, or in terms of the inductive method applied in order to induce them, or both. Within TC, the only avenue which has been explored is, to our knowledge, the latter.

Different combination rules have been experimented with in the literature. The simplest possible rule is *majority voting* (MV), whereby the binary classification judgments obtained by the k classifiers are pooled together, and the classification decision that reaches the majority of $\frac{k+1}{2}$ votes is taken (k obviously needs to be an odd number) [Li and Jain 1998; Liere and Tadepalli 1997]. This method is particularly suited to the case in which the committee includes classifiers characterised

[9]An empirical attempt at implementing this principle is also present in the version of the Balanced Winnow algorithm proposed in [Dagan et al. 1997].

[10]Some of the classifiers that we have already touched upon in previous sections actually fit into this picture; among them, Sleeping Experts [Cohen and Singer 1999] and Widrow-Hoff [Lewis et al. 1996], both discussed in Section 6.5.

by a binary decision function $CSV_i : D \rightarrow \{0, 1\}$. A second rule is *weighted linear combination* (WLC), whereby a weighted sum of the CSV_i's individually produced by the k classifiers yields the final CSV_i. The weights w_j are meant to reflect the expected relative effectiveness of classifier Φ_j, and are such that $\sum_{j=1}^{k} w_j = 1$. Typically, these weights are optimised on a validation set [Larkey and Croft 1996]. Another possible policy is *dynamic classifier selection* (DCS), whereby among committee $\{\Phi_1, \ldots, \Phi_k\}$ the classifier Φ_t that yields the best effectiveness on the l validation examples most similar to d_j is selected, and its judgment adopted by the committee [Li and Jain 1998]. A still different policy, somehow intermediate between WLC and DCS, is *adaptive classifier combination* (ACC), whereby the judgments of *all* the classifiers in the committee are summed together, but their individual contribution is weighted by the effectiveness that they have shown on the l validation examples most similar to d_j [Li and Jain 1998].

Classifier committees have had mixed results in TC so far. Larkey and Croft [1996] have used combinations of Rocchio, Naïve Bayes and k-NN, all together or in pairwise combinations, using a WLC rule. In their experiments the combination of any two classifiers has outperformed the best individual classifier (k-NN), and the combination of the three classifiers has improved an all three pairwise combinations. These results would seem to give strong support to the idea that classifier committees can somehow profit from the complementary strengths of their individual members. However, the small size of the test set used (187 documents) suggests that more experimentation is needed before conclusions can be reached.

Li and Jain [1998] have experimented with a committee formed of (various combinations of) a Naïve Bayes classifier, a nearest neighbour classifier, a decision tree classifier, and a classifier induced by means of their own "subspace method"; the combination rules they have worked with are MV, DCS and ACC. Only in the case of a committee formed by Naïve Bayes and the subspace classifier combined by means of ACC the committee has outperformed, and by a narrow margin, the best individual classifier (for every attempted classifier combination, anyway, ACC gave better results than MV and DCS). This seems discouraging, especially in the light of the fact that the committee approach is computationally expensive (its cost trivially amounts to the sum of the computational costs of the individual classifiers plus the cost incurred for the computation of the combination rule). Again, it has to be remarked that the small size of their experiment (two test sets of less than 700 documents each were used) does not allow to draw definitive conclusions on the approaches adopted.

6.10.1 *Boosting.* The *boosting* method [Schapire et al. 1998; Schapire and Singer 2000] occupies a special place in the classifier committees literature, since the k classifiers $\Phi_1, \ldots, \Phi_t$ forming the committee are obtained not by means of k different learning methods, but by the *same* learning method (here called the *weak learner*). For instance, if a decision tree classifier is used as the weak learner, the resulting committee will be formed by k decision tree classifiers. The key intuition of boosting is that the k classifiers should be trained not in a conceptually parallel and independent way, as in the classifier committees described above, but sequentially, one after the other. In this way, the training of classifier Φ_i may take into account how classifiers $\Phi_1, \ldots, \Phi_{i-1}$ perform on the training examples, and concentrate on

getting right those examples on which $\Phi_1, \ldots, \Phi_{i-1}$ have performed worst.

Specifically, for the induction of classifier Φ_t each $\langle \overline{d}_j, c_i \rangle$ pair is attributed an "importance weight" h_{ij}^t (where h_{ij}^1 is set to be equal for all $\langle \overline{d}_j, c_i \rangle$ pairs[11]), meant to represent how hard to get a correct decision for this pair was for classifiers $\Phi_1, \ldots, \Phi_{t-1}$. These weights are exploited in the induction of classifier Φ_t, which will be specially tuned to solve correctly the pairs with higher importance weight. The induced classifier Φ_t is then applied to the training documents, and as a result weights h_{ij}^t are updated to h_{ij}^{t+1}; in this update operation, pairs correctly classified by Φ_t will have their importance weight decreased, while pairs misclassified by Φ_t will have their weight increased. After all the k classifiers have been constructed, a weighted linear combination rule is applied to yield the final committee, where the weight attributed to the decision contributed by classifier Φ_t is a function of the effectiveness that Φ_t has shown on the training set.

In the BOOSTEXTER system [Schapire and Singer 2000], two different boosting algorithms are provided and experimented with, using a one-level decision tree weak learner. The former algorithm (ADABOOST.MH, simply called ADABOOST in [Schapire et al. 1998]) is explicitly geared towards the maximization of microaveraged effectiveness, whereas the latter (ADABOOST.MR) is aimed at minimizing *ranking loss* (i.e. at getting a correct category ranking for each individual document). In experiments conducted over three different test collections, Schapire et al. [1998] have shown ADABOOST to outperform SLEEPING EXPERTS, a classifier that had proved quite effective in the experiments of [Cohen and Singer 1999]. Further experiments by Schapire and Singer [2000] showed ADABOOST to outperform, aside from SLEEPING EXPERTS, a Naïve Bayes classifier, a standard (non-enhanced) Rocchio classifier, and Joachims' [1997] PRTFIDF classifier. Boosting has also been used in [Li and Jain 1998], with a decision tree classifier as the weak learner; the authors reported a significant (13%) improvement in effectiveness over the pure weak learner.

An approach similar to boosting is employed by Weiss et al. [1999]. In this work the authors experiment with committees of decision trees each having an average of 16 leaves (hence much more complex than the simple 2-leaves trees used in the [Schapire and Singer 2000] experiment), eventually combined by using the simple MV rule as a combination rule. Similarly to boosting, a mechanism for emphasising documents that have been misclassified by previous decision trees is enforced. The authors have experimentally determined that this approach yields excellent effectiveness gains over the individual decision tree case (and excellent effectiveness *tout court*), with gains rising briskly until the 10 trees case and reaching a plateau at about 100 trees.

6.11 Other methods

Although in the previous sections we have tried to give an overview as complete as possible of the approaches that have been proposed in the automated TC literature, it would be hardly possible to be exhaustive in this respect. The recent explosion of this discipline has brought about a fragmentation in terms of the learning ap-

[11] Schapire et al. [1998] also show that a simple modification of this policy allows an evaluation of the classifier based on "utility" (see Section 8.1.3) rather than effectiveness.

proaches adopted, some of which either do not fall squarely under one or the other class of algorithms, or have remained somehow isolated attempts. Although for reasons of space we will not discuss them in detail, we at least want to mention the existence of approaches based on *Bayesian inference networks* [Dumais et al. 1998; Lam et al. 1997; Tzeras and Hartmann 1993] and *genetic algorithms* [Clack et al. 1997].

7. DETERMINING THRESHOLDS

There are various possible policies for determining the threshold τ_i discussed at the beginning of Section 6, also depending on the constraints imposed by the application.

One possible policy is *CSV thresholding* [Cohen and Singer 1999; Schapire et al. 1998; Wiener et al. 1995] (also called *probability thresholding* in the case of probabilistic classifiers [Lewis 1992], or *Scut* [Yang 1999]). In this case the threshold τ_i is a value of the CSV_i function. Lewis [1992] considered using a fixed threshold τ equal for all c_i's, but noted that this might result in assigning *all* the test documents to a category c_i while not even assigning a single test document to another category c_j. He then considered using different thresholds τ_i for different categories c_i established by normalising probability estimates (following a suggestion from [Maron 1961]). His experimental results did not show, however, a considerable difference in effectiveness between the two variants. Yang [1999] uses different thresholds τ_i for the different categories c_i. Each threshold is optimised by testing different values for it on the validation set and choosing the value which yields the best value of the chosen effectiveness function.

A second, popular policy is *proportional thresholding* [Iwayama and Tokunaga 1995; Larkey 1998; Lewis 1992; Lewis and Ringuette 1994; Wiener et al. 1995] (also called *Pcut* in [Yang 1999]). The aim of this policy is to set the threshold τ_i so that the test set generality $g_{Te}(c_i)$ of a category c_i is as close as possible to its training set generality $g_{Tr}(c_i)$. This idea encodes the quite sensible principle according to which the same percentage of documents of both training and test set should be classified under c_i. One drawback of this thresholding policy is that, for obvious reasons, it does not lend itself to document-pivoted categorisation. Yang [1999] proposes a still more refined version of this policy, i.e. one in which a factor x (equal for all c_i's) is multiplied to $g_{Tr}(c_i)$ to actually obtain $g_{Te}(c_i)$. Yang claims that this factor, whose value is to be empirically determined by experimentation on a validation set, allows a smoother trade-off between recall and precision to be obtained (see Section 8.1.1). For both k-NN and LLSF she found that optimal values lie in the [1.2, 1.3] range.

Sometimes, depending on the application, a *fixed thresholding* policy (also known as "k-per-doc" thresholding [Lewis 1992] or *Rcut* [Yang 1999]) is applied, whereby it is stipulated that a fixed number k of categories, equal for all d_j's, are to be assigned to each document d_j. This is often used, for instance, in applications of TC to automated document indexing [Field 1975; Lam et al. 1999]. Strictly speaking, however, this is not a thresholding policy in the sense defined at the beginning of Section 6, as it might happen that d' is classified under c_i, d'' is not, and $CSV_i(d') < CSV_i(d'')$. Quite clearly, this policy is mostly at home with document-pivoted categorisation. It suffers, however, from a certain coarseness, as

the fact that k is equal for all documents (nor could this be otherwise) does not allow system fine-tuning.

In terms of experimental results, Lewis [1992] found the proportional policy to be definitely superior to CSV thresholding when microaveraged effectiveness was tested but slightly inferior when using macroaveraging (see Section 8.1.1). Yang [1999] found instead CSV thresholding to be superior to proportional thresholding (possibly due to her category-specific optimisation on a validation set), and found fixed thresholding to be consistently inferior to the other two policies. Of course, the fact that these results have been obtained across different classifiers no doubt reinforce them.

In general, aside from the considerations above, the choice of the thresholding policy may also be influenced by the application; for instance, in applying a text classifier to document indexing for Boolean systems a fixed thresholding policy might be chosen, while a proportional or CSV thresholding method might be chosen for Web page classification under Yahoo!-like catalogues.

As a final note we recall that in some applications thresholding is not needed, as it may be better to return the original non-binary value produced by the CSV_i function rather than a binary value obtained from it by means of thresholding. This is typically the case of *interactive classification systems* [Larkey and Croft 1996]. Given a document d_j to classify, such a system is meant to suggest a ranked list of categories apt for classifying d_j to an expert, who then takes the final categorisation decision. In this case, the list of categories c_i ranked by their $CSV_i(d_j)$ value is much more useful to the expert than the flat set of relevant categories produced by thresholding. Interactive classification systems are useful especially when the quality of the training data is low, or when the training data cannot be trusted as being a representative sample of the unseen data that are to come, so that the results of a completely automatic classifier could not be trusted completely.

8. EVALUATION ISSUES FOR TEXT CATEGORISATION

As in the case of IR systems, the evaluation of document classifiers is typically conducted *experimentally*, rather than analytically. The reason for this tendency is that, in order to evaluate a system analytically (e.g. proving that the system is correct and complete) we would need a formal specification of the problem that the system is trying to solve (e.g. *with respect to what* correctness and completeness are defined), and the central notion of TC (namely, that of relevance of a document to a category) is, due to its subjective character, inherently non-formalisable.

The experimental evaluation of classifiers, rather than concentrating on issues of efficiency, usually tries to evaluate the *effectiveness* of a classifier, i.e. its capability of taking the *right* categorisation decisions.

8.1 Measures of categorisation effectiveness

8.1.1 *Precision and recall.* Classification effectiveness is most often measured in terms of the classic IR notions of precision (Pr) and recall (Re), adapted to the case of document categorisation. *Precision wrt* c_i (Pr_i) is defined as the conditional probability $P(ca_{ix} = 1 \mid a_{ix} = 1)$, i.e. as the probability that if a random document d_x is classified under c_i, this decision is correct. Analogously, *recall wrt* c_i (Re_i) is defined as the conditional probability $P(a_{ix} = 1 \mid ca_{ix} = 1)$, i.e. as the probability

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

Table 2. The contingency table for category c_i.

that, if a random document d_x ought to be classified under c_i, this decision is taken. These category-relative values may be averaged, in a way to be discussed shortly, to obtain Pr and Re, i.e. values global to the whole category set. Borrowing terminology from logic, Pr may be viewed as the "degree of soundness" of the classifier wrt the given category set C, while Re may be viewed as its "degree of completeness" wrt C.

As they are defined here, Pr_i and Re_i (and consequently Pr and Re) are to be understood, in the line of [Wong and Yao 1995], as *subjective* probabilities, i.e. values measuring the expectation of the user that the system will behave correctly when classifying a random document under c_i. These probabilities may be estimated in terms of the *contingency table* for category c_i on a given test set (see Table 2). Here, FP_i (*false positives wrt* c_i, also known as *errors of commission*) is the number of documents of the test set that have been incorrectly classified under c_i; TN_i (*true negatives wrt* c_i), TP_i (*true positives wrt* c_i) and FN_i (*false negatives wrt* c_i, also known as *errors of omission*) are defined accordingly. Estimates (indicated by carets) of precision wrt c_i and recall wrt c_i may thus be obtained as

$$\hat{Pr}_i = \frac{TP_i}{TP_i + FP_i}$$
$$\hat{Re}_i = \frac{TP_i}{TP_i + FN_i}$$

For obtaining estimates of precision and recall relative to the whole category set, two different methods may be adopted:

—*microaveraging*: precision and recall are obtained by globally summing over all individual decisions, i.e.:

$$\hat{Pr}^{\mu} = \frac{TP}{TP + FP} = \frac{\sum_{i=1}^{m} TP_i}{\sum_{i=1}^{m}(TP_i + FP_i)}$$
$$\hat{Re}^{\mu} = \frac{TP}{TP + FN} = \frac{\sum_{i=1}^{m} TP_i}{\sum_{i=1}^{m}(TP_i + FN_i)}$$

where the "μ" superscript stands for microaveraging. The "global" contingency table shown in Table 3 is thus obtained by summing over all category-specific contingency tables.

—*macroaveraging* : precision and recall are first evaluated "locally" for each category, and then "globally" by averaging over the results of the different categories, i.e.:

$$\hat{Pr}^{M} = \frac{\sum_{i=1}^{m} \hat{Pr}_i}{m}$$

Category set $C = \{c_1, \ldots, c_m\}$		expert judgments	
		YES	**NO**
classifier	**YES**	$TP = \sum_{i=1}^{m} TP_i$	$FP = \sum_{i=1}^{m} FP_i$
judgments	**NO**	$FN = \sum_{i=1}^{m} FN_i$	$TN = \sum_{i=1}^{m} TN_i$

Table 3. The global contingency table.

$$\hat{Re}^{M} = \frac{\sum_{i=1}^{m} \hat{Re}_i}{m}$$

where the "M" superscript stands for macroaveraging.

It is important to recognise that these two methods may give quite different results, especially if the different categories are unevenly populated: for instance, if the classifier performs well just on categories with a large number of positive test instances, its effectiveness will probably be better according to microaveraging than to macroaveraging. There is no complete agreement among authors on which is better. Some believe that "microaveraged performance is somewhat misleading (...) because more frequent topics are weighted heavier in the average" [Wiener et al. 1995, page 327] and thus favour macroaveraging, which indeed rewards those classifiers that perform robustly also in the presence of skewed category distributions. Others (actually, the majority of researchers) believe that topics should indeed count proportionally to their frequence, and thus lean towards microaveraging. From now on, we will assume that microaveraging is used, and will thus drop the "i" subscript from Pr, Re and other symbols; it should be clear, however, that everything we will say in the rest of Section 8 may be adapted to the case of macroaveraging in the obvious way.

8.1.2 *Other measures of categorisation effectiveness.* Other measures alternative to Pr and Re and commonly used in the machine learning literature, such as *accuracy* (which can be estimated as $\hat{Ac} = \frac{TP+TN}{TP+TN+FP+FN}$) and *error* (which can be estimated as $\hat{Er} = \frac{FP+FN}{TP+TN+FP+FN} = 1 - \hat{Ac}$) are not widely used in TC. The reason for this is that, as Yang [1999] noted, the typically large value that their denominator takes up in TC makes them much more insensitive to a variation in the number of correct decisions ($TP + TN$) than Pr and Re are. Besides, [Yang 1999] shows how taking Ac as the evaluation measure, in the (frequent) case of a low value for $apc \stackrel{def}{=} \frac{TP+FN}{TP+TN+FP+FN} = \sum_{i=1}^{m} g_{Te}(c_i)$ (which represents the average percentage of categories per test document), the *trivial rejector* (i.e. the classifier that sets $a_{ij} = 0$ for all $1 \leq i \leq m, 1 \leq j \leq n$, thereby not classifying any document under any category) tends to outperform all non-trivial classifiers (see also [Cohen 1995a, Section 2.3]). A consequence of adopting Ac is also that parameter tuning for a non-trivial classifier on a validation set would result in parameter choices that make the classifier behave very much like the trivial rejector.

A non-standard effectiveness measure is proposed by Sable and Hatzivassiloglou [1999, Section 7], who suggest to base precision and recall not on "absolute" values of suc-

cess ($a_{ij} = ca_{ij}$) and failure ($a_{ij} \neq ca_{ij}$), but on values of *relative success*, given by

$$\begin{aligned} CSV_i(d_j) \text{ if } ca_{ij} = 1 \\ 1 - CSV_i(d_j) \text{ if } ca_{ij} = 0 \end{aligned}$$

In other words, in the case of a positive example d_j for category c_i, the classifier score will not be the binary decision taken, but the degree of confidence $CSV_i(d_j)$ that the classifier has in the fact that d_j should indeed be classified under c_i:

> "In this way, the system receives partial credit for each answer, more if the system leans in the correct direction and directly increasing as the system's confidence in a correct decision increases. (...) we consider this modified method as more revealing, as it offers a way to evaluate the system's confidence in its decisions.". [Sable and Hatzivassiloglou 1999, page 33]

This proposed measure does not reward the choice of a good thresholding policy, and is thus unfit for autonomous (fully automatic) classification systems. However, it might be appropriate for interactive classifiers of the type used in [Larkey 1999], where the confidence that the classifier has in its own decision influences category ranking and, as a consequence, the overall usefulness of the system.

8.1.3 *Measures alternative to effectiveness.* In general, criteria different from effectiveness are seldom used in classifier evaluation. For instance, the *efficiency* criterion, although important for applicative purposes, is seldom used as the sole yardstick on which to evaluate or compare classifiers, due to the volatility of the parameters on which the evaluation rests. However, efficiency may be a useful criterion for choosing among classifiers with similar effectiveness. On this respect, an interesting evaluation has been carried out by Dumais et al. [1998], who have compared five different classifier induction methods along three different dimensions, namely effectiveness, *training efficiency* (i.e. the average time it takes to build a classifier for a category c_i from a training set Tr with a given induction method), and *classification efficiency* (i.e. the average time it takes to classify a new document d_j under category c_i with a given classifier). We refer the interested reader to [Dumais et al. 1998] for details.

One exception to the dominance of the effectiveness criterion in TC evaluation is perhaps *utility*, a class of measures well-known in decision theory that extend effectiveness by economic criteria such as *gain* or *loss*. Utility is based on defining a *utility matrix* such as that of Table 4, where the numeric values u_{TP}, u_{FP}, u_{FN} and u_{TN} represent the economic gain brought about by a true positive, false positive, false negative and true negative, respectively; both u_{TP} and u_{TN} are greater than both u_{FP} and u_{FN}. "Standard" effectiveness is a particular case of utility, i.e. the one in which $u_{TP} = u_{TN} > u_{FP} = u_{FN}$. Less trivial cases of utility are those in which $u_{TP} \neq u_{TN}$ and/or $u_{FP} \neq u_{FN}$; for instance, Schapire et al. [1998] experiment with the three different utility matrices

Category set $C = \{c_1, \ldots, c_m\}$		expert judgments	
		YES	**NO**
classifier judgments	**YES**	u_{TP}	u_{FP}
	NO	u_{FN}	u_{TN}

Table 4. The utility matrix.

	u_{TP}	u_{FP}	u_{FN}	u_{TN}
Matrix 1	0	-1	-1	0
Matrix 2	3	-2	0	0
Matrix 3	3	-1	-1	0

among which the first corresponds to effectiveness.

The use of utility measures in TC is discussed in detail by Lewis [1995a]. Other works where utility measures are employed are [Amati and Crestani 1999; Cohen and Singer 1999; Hull et al. 1996; Lewis and Catlett 1994]. As a matter of fact, utility is being more and more used within the text filtering community, as the TREC "filtering track" evaluations have recently been using utility measures [Lewis 1995c; Hull 1998]. The problem with using utility instead of "pure" effectiveness is that the values of the utility matrix are extremely application-dependent. Needless to say, this evaluation measure adds a further element of difficulty in the cross-comparison of classification systems (see Section 8.3), since for two classifiers to be experimentally comparable the two utility matrices must be the same.

Finally, we should at least mention the fact that other effectiveness measures different from the ones discussed in this section have occasionally been used in the literature; these include *adjacent score* [Larkey 1998], *coverage* [Schapire and Singer 2000], *one-error* [Schapire and Singer 2000], *Pearson product-moment correlation* [Larkey 1998], *recall at n* [Larkey and Croft 1996], *top candidate* [Larkey and Croft 1996], *top n* [Larkey and Croft 1996]. We will not attempt to discuss in detail these other effectiveness measures and their relationships with the more standard ones. This only points to the fact that, although the TC discipline is making consistent efforts at standardising experimentation protocols, we are still far from universal agreement on evaluation issues and, as a consequence, far from understanding precisely the relative merits of the various methods.

8.1.4 *Combined effectiveness measures.* Neither precision nor recall make sense in isolation of the other. In fact, in order to obtain a classifier with 100% recall, one would only need to set every threshold τ_i to 0, thereby obtaining the *trivial acceptor* (i.e. the classifier that sets $a_{ij} = 1$ for all $1 \leq i \leq m, 1 \leq j \leq n$, i.e. that classifies all documents under all categories). Quite obviously, in this case precision would usually be very low (more precisely, equal to the average test set generality $\frac{\sum_{i=1}^{m} g_{Te}(c_i)}{m}$)[12]. Conversely, it is well-known from everyday IR practice that higher

[12]From what we said about the trivial acceptor, one might be tempted to infer by symmetry that the trivial rejector has 100% precision. This is not true, as precision is undefined (the denominator is zero) in the case of the trivial rejector (see Table 5). In fact, it is also clear from its definition ($Pr = \frac{TP}{TP+FP}$) that precision depends only on how the positives ($TP + FP$) are split between *true* positives TP and the *false* positives FP, and does not depend at all on the cardinality

		Precision $\frac{TP}{TP+FP}$	Recall $\frac{TP}{TP+FN}$	C-Precision $\frac{TN}{FP+TN}$	C-Recall $\frac{TN}{TN+FN}$
Trivial Rejector	TP=FP=0	undefined	$\frac{0}{FN}=0$	$\frac{TN}{TN}=1$	$\frac{TN}{TN+FN}$
Trivial Acceptor	FN=TN=0	$\frac{TP}{TP+FP}$	$\frac{TP}{TP}=1$	$\frac{0}{FP}=0$	undefined
Trivial "Yes" Collection	FP=TN=0	$\frac{TP}{TP}=1$	$\frac{TP}{TP+FN}$	undefined	$\frac{0}{FN}=0$
Trivial "No" Collection	TP=FN=0	$\frac{0}{FP}=0$	undefined	$\frac{TN}{FP+TN}$	$\frac{TN}{TN}=1$

Table 5. Trivial cases in text categorisation.

levels of precision may be obtained at the price of a low recall[13].

In practice, by tuning thresholds τ_i a classification algorithm is tuned so as to be, in the words of Riloff and Lehnert [1994], more *liberal* (i.e. improving Re to the detriment of Pr) or more *conservative* (improving Pr to the detriment of Re). A classifier should thus be measured by means of a "combined" effectiveness measure which both Pr and Re concur to determine. Various such measures have been proposed, among which the following are most frequent:

(1) effectiveness is computed as *(interpolated) 11-point average precision*. That is, each threshold τ_i is successively set to the values for which recall takes up values of 0.0, 0.1, ..., 0.9, 1.0; for these 11 different thresholds precision is computed and averaged over the 11 resulting values. This methodology is completely analogous to the standard evaluation methodology for ranked retrieval systems, and may be used
   (a) with categories used in place of IR queries. This is most frequently used for classifiers that rank test documents according to their appropriateness to a category (see e.g [Schütze et al. 1995; Yang 1994; Yang 1999; Yang and Pedersen 1997]);
   (b) with test documents used in place of IR queries and categories in place of documents. This is most frequently used for classifiers that rank categories according to their appropriateness to a test document (see e.g. [Lam et al. 1999; Larkey and Croft 1996; Schapire and Singer 2000; Wiener et al. 1995]). Note that in this case if macroaveraging is used it needs to be

of the positives. There is a breakup of "symmetry" between precision and recall here because, from the point of view of classifier judgment (positives vs. negatives; this is the dichotomy of interest in trivial acceptor vs. trivial rejector) the "symmetric" of recall ($\frac{TP}{TP+FN}$) is not precision ($\frac{TP}{TP+FP}$) but *c-precision* ($\frac{TN}{FP+TN}$), the "contrapositive" of precision. In fact, while recall=1 and c-precision=0 for the trivial acceptor, c-precision=1 and recall=0 for the trivial rejector.

[13]Of course, the fact that we may arbitrarily set the threshold so as to increase recall at the expense of precision and viceversa is true only for classifiers that employ thresholds. Decision tree classifiers and DNF rule classifiers do not have thresholds, as they already produce binary decisions. Hence, with them this tuning is not possible, or is anyway more difficult (see [Weiss et al. 1999, page 66]).

redefined on a per-document, rather than per-category, basis.

Note that this measure does not make much sense for classifiers that do not perform any ranking, for which recall is an absolute number and thus may not be varied at will.

(2) effectiveness is computed as the *breakeven* point, i.e. the value at which Pr equals Re (e.g. [Apté et al. 1994; Cohen and Singer 1999; Dagan et al. 1997; Joachims 1998; Joachims 1999; Lewis 1992; Lewis and Ringuette 1994; Moulinier and Ganascia 1996; Ng et al. 1997; Yang 1999]). In all reasonable classifiers a value for each τ_i for which Pr and Re are (almost) equal does exist, since by increasing the τ_i's from 0 to 1 Pr usually increases monotonically from $apc \stackrel{def}{=} \sum_{i=1}^{m} g_{Te}(c_i)$ to a value near 1 and Re always decreases monotonically from 1 to 0. If for no values of the τ_i's Pr and Re are exactly equal, the τ_i's are set to the value for which Pr and Re are closest, and an *interpolated breakeven* is computed as the average of the values of Pr and Re. As noted in [Yang 1999], when for no value of τ_i Pr and Re are close enough, interpolated breakeven may not be a reliable indicator of the effectiveness of the classifier;

(3) effectiveness is computed as the value of the F_β *function*, for some $0 \leq \beta \leq +\infty$ (e.g. [Cohen 1995a; Cohen and Singer 1999; Lewis and Gale 1994; Lewis 1995a; Moulinier et al. 1996; Ruiz and Srinivasan 1999]), where

$$F_\beta = \frac{(\beta^2 + 1) \cdot Pr \cdot Re}{\beta^2 \cdot Pr + Re}$$

In this formula β may be seen as the relative degree of importance attributed to Pr and Re: if $\beta = 0$ then F_β coincides with Pr, whereas if $\beta = +\infty$ then F_β coincides with Re. Usually, a value $\beta = 1$ is used, which attributes equal importance to Pr and Re. As shown in [Moulinier et al. 1996; Yang 1999], the breakeven value of a classifier Φ is always less or equal than its F_1 value.

Once an effectiveness measure is chosen, a classifier can be tuned (e.g. thresholds and other internal parameters can be set) so that the resulting effectiveness is the best achievable by that classifier. The tuning of a parameter p (be it a threshold or other) is normally done experimentally. This means performing repeated experiments on the validation set in the same experimental conditions, with the values of the other parameters p_k fixed (at a default value, in the case of a yet-to-be-tuned parameter p_k, or at the chosen value, if the parameter p_k has already been tuned) and with different values for parameter p. At the end of the process, the value that has yielded the best effectiveness is chosen for p.

8.2 Benchmark collections

For experimentation purposes, standard *benchmark collections* that can be used as initial corpora for TC are available in the public domain. The most widely used is the Reuters collection, consisting of a set of newswire stories classified under categories related to economics. The Reuters collection accounts for most of the experimental work in TC accomplished so far. Unfortunately, this does not always translate into reliable comparative results, in the sense that many of these experiments have been carried out in subtly different experimental conditions. In order

for the experimental results on different classifiers to be directly comparable, the experiments should be performed under the following conditions:

(1) All classifiers are experimented on the same collection (i.e. same documents and same categories), and their effectiveness is measured on the complete set of categories.
(2) The same choice ("split") of training set and test set is made for all classifiers, and their effectiveness is measured on the complete test set.
(3) The same effectiveness measure is used for all classifiers, and if it depends on some parameters (e.g. utility, which depends on the utility matrix chosen) the same parameter choice is made for all classifiers.

Unfortunately, a lot of experimentation, both on Reuters and on other collections, has not been performed with these *caveat* in mind. By experimenting three different classifiers on five versions of Reuters, Yang [1999] has shown that a lack of compliance with these three conditions may significantly influence the experimental results. Table 6 lists the results of all experiments known to us that were performed on five major versions of the Reuters benchmark: Reuters-22173 "ModLewis" (column #1), Reuters-22173 "ModApté" (column #2), Reuters-22173 "ModWiener" (column #3), Reuters-21578 "ModApté" (column #4) and Reuters-21578[10] "ModApté" (column #5)[14]. Only experiments that have computed either a breakeven or an F_1 result have been listed, since other less popular effectiveness measures do not readily compare with these.

Note that only results belonging to the same column are directly comparable. In particular, Yang [1999] shows that experiments carried out on Reuters-22173 "ModLewis" (column #1) are not directly comparable with those using the other three versions, since the former strangely includes a significant percentage (58%) of "unlabelled" test documents which, being negative examples of all categories, tend to depress effectiveness. Also, experiments performed on Reuters-21578[10] "ModApté" (column #5) are not comparable with the others, since this collection consists in the restriction of Reuters-21578 "ModApté" to its 10 most populated categories , and is thus an obviously "easier" collection.

Other collections that have been frequently used for TC evaluation are

—the OHSUMED collection, set up by Hersh et al. [1994] and used in [Joachims 1998; Lam and Ho 1998; Lam et al. 1999; Lewis et al. 1996; Ruiz and Srinivasan 1999; Yang and Pedersen 1997][15]. The documents are titles or title-plus-abstract's from medical journals (OHSUMED actually consists of a subset of the Medline document base); the categories are the postable terms of the MESH thesaurus.

[14] The Reuters-21578 collection may be freely downloaded for experimentation purposes from `http://www.research.att.com/~lewis/reuters21578.html` and is now considered the "standard" variant of Reuters. We have decided not to cover experiments performed on variants of the Reuters benchmark different from the five listed because the small number of authors that have experimented on the same variant makes the reported results difficult to interpret. This includes experiments performed on the original Reuters-22173 "ModHayes" [Hayes et al. 1990] and Reuters-21578 "ModLewis" [Cohen and Singer 1999].

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

			#1	#2	#3	#4	#5
		# of documents	21,450	14,347	13,272	12,902	12,902
		# of training documents	14,704	10,667	9,610	9,603	9,603
		# of test documents	6,746	3,680	3,662	3,299	3,299
		# of categories	135	93	92	90	10
System	**Type**	**Results reported by**					
WORD	(non-learning)	[Yang 1999]	.150	.310	.290		
	probabilistic	[Dumais et al. 1998]				.752	.815
	probabilistic	[Joachims 1998]					.720
	probabilistic	[Lam et al. 1997]	.443 (MF_1)				
PROPBAYES	probabilistic	[Lewis 1992]	.650				
BIM	probabilistic	[Li and Yamanishi 1999]				.747	
	probabilistic	[Li and Yamanishi 1999]				.773	
NB	probabilistic	[Yang and Liu 1999]				.795	
	decision trees	[Dumais et al. 1998]					.884
C4.5	decision trees	[Joachims 1998]					.794
IND	decision trees	[Lewis and Ringuette 1994]	.670				
SWAP-1	decision rules	[Apté et al. 1994]		.805			
RIPPER	decision rules	[Cohen and Singer 1999]	.683	.811		.820	
SLEEPINGEXPERTS	decision rules	[Cohen and Singer 1999]	**.753**	.759		.827	
DL-ESC	decision rules	[Li and Yamanishi 1999]				.820	
CHARADE	decision rules	[Moulinier and Ganascia 1996]		.738			
CHARADE	decision rules	[Moulinier et al. 1996]		.783 (F_1)			
LLSF	regression	[Yang 1999]		.855	.810		
LLSF	regression	[Yang and Liu 1999]				.849	
BALANCEDWINNOW	on-line linear	[Dagan et al. 1997]	.747	.833			
WIDROW-HOFF	on-line linear	[Lam and Ho 1998]				.822	
ROCCHIO	batch linear	[Cohen and Singer 1999]	.660	.748		.776	
FINDSIM	batch linear	[Dumais et al. 1998]				.617	.646
ROCCHIO	batch linear	[Joachims 1998]					.799
ROCCHIO	batch linear	[Lam and Ho 1998]				.781	
ROCCHIO	batch linear	[Li and Yamanishi 1999]				.625	
CLASSI	neural network	[Ng et al. 1997]		.802			
NNET	neural network	[Yang and Liu 1999]				.838	
	neural network	[Wiener et al. 1995]			**.820**		
GIS-W	example-based	[Lam and Ho 1998]				.860	
k-NN	example-based	[Joachims 1998]					.823
k-NN	example-based	[Lam and Ho 1998]				.820	
k-NN	example-based	[Yang 1999]	.690	.852	**.820**		
k-NN	example-based	[Yang and Liu 1999]				.856	
	SVM	[Dumais et al. 1998]				.870	**.920**
SVMLIGHT	SVM	[Joachims 1998]					.864
SVMLIGHT	SVM	[Li and Yamanishi 1999]				.841	
SVMLIGHT	SVM	[Yang and Liu 1999]				.859	
ADABOOST.MH	committee	[Schapire and Singer 2000]		**.860**			
	committee	[Weiss et al. 1999]				**.878**	
	Bayesian net	[Dumais et al. 1998]				.800	.850
	Bayesian net	[Lam et al. 1997]	.542 (MF_1)				

Table 6. Comparative results among different classifiers obtained on five different version of the Reuters collection. Unless otherwise noted, entries indicate the microaveraged breakeven point; within parentheses, "M" indicates macroaveraging and "F_1" indicates use of the F_1 measure. **Boldface** indicates the best performer on the collection.

—the 20 Newsgroups collection, set up by Lang [1995] and used in [Baker and McCallum 1998; Joachims 1997; McCallum and Nigam 1998; McCallum et al. 1998; Nigam et al. 1998; Schapire and Singer 2000]. The documents are messages posted to Usenet newsgroups, and the categories are the newsgroups themselves.

—the AP collection, used in [Cohen 1995a; Cohen 1995b; Cohen and Singer 1999; Lewis and Catlett 1994; Lewis and Gale 1994; Lewis et al. 1996; Schapire and Singer 2000; Schapire et al. 1998].

—the TREC-3 "Routing" collection, used in [Lewis et al. 1996; Schapire et al. 1998; Schütze et al. 1995].

We will not cover the experiments performed on these collections for the same reasons as those illustrated in Footnote 14, i.e. becauset in no case a significant enough number of authors have experimented with the same collection and in the same experimental conditions, thus making comparisons difficult.

8.3 Which classifier is best?

The published experimental results, and especially those listed in Table 6, allow us to attempt some considerations on the comparative performance of the TC methods discussed. However, in order to do this we have to bear in mind that comparisons tend to be reliable only when they concern experiments performed by the same author under carefully controlled conditions. They are instead more problematic when they involve different experiments performed by different people. In this case various "background conditions", often extraneous to the learning algorithm proper, may have influenced the experimental results. This may include, among others, different choices in pre-processing (stemming, etc.), indexing, dimensionality reduction, classifier parameter values, etc., but also different standards of compliance with safe scientific practice (such as tuning parameters on the test set rather than on a separate validation set), which often are not discussed in the published papers.

As a result, there are two different methods that may be applied for the comparison of classifiers [Yang 1999]:

—*direct comparison*: classifiers Φ' and Φ'' may be compared as they have been tested on the same benchmark collection BC, typically by the same team of researchers and under the same background conditions. This is the method that yields the more reliable results.

—*indirect comparison*: classifiers Φ' and Φ'' may be compared as

(1) they have been tested on collections BC' and BC'', respectively, typically by two different teams of researchers and hence under possibly different background conditions.

(2) one or more "baseline" classifiers $\overline{\Phi}_1, \ldots, \overline{\Phi}_m$ have been tested on both TC' and TC'' by the direct comparison method.

Test 2 gives an indication on the relative "hardness" of the two collections; using this and the results from Test 1 we may obtain an indication on the relative effectiveness of Φ' and Φ''. For the reasons discussed above, this is the method that yields the less reliable results.

A number of interesting conclusions can be drawn from Table 6 by using these two methods. Concerning the relative "hardness" of the five collections, if by $BC' >$

BC'' we indicate that BC' is a harder collection that BC'', there seems to be enough evidence that Reuters-22173 "ModLewis" $\gg$ Reuters-22173 "ModWiener" $>$ Reuters-22173 "ModApté" $\approx$ Reuters-21578 "ModApté" $>$ Reuters-21578[10] "ModApté". These facts are not unsurprising; in particular, the first and the last inequalities are a direct consequence of the peculiar characteristics of Reuters-22173 "ModLewis" and Reuters-21578[10] "ModApté" discussed in Section 8.2.

Concerning the relative performance of the classifiers, remembering the considerations above we may attempt a few conclusions:

—Boosting-based classifier committees, support vector machines, example-based methods, and regression methods deliver top-notch performance. There seems to be no sufficient evidence to decidedly opt for either method, and it looks that efficiency considerations or application-dependent issues might play a role in breaking such a close tie.

—Neural networks and on-line linear classifiers work very well, although slightly worse than the previously mentioned methods.

—Batch linear classifiers (Rocchio) and probabilistic Naïve Bayes classifiers definitely look the worst of learning-based classifiers. For Rocchio, these results confirm other earlier results by Schütze et al. [1995], who had found three classifiers based on linear discriminant analysis, linear regression, and neural networks, to perform about 15% better than Rocchio. It should be mentioned, however, that recent results by Schapire and Singer [2000] rank Rocchio along the best performers once near-positives are used in training.

—The data in Table 6 seem hardly sufficient to say anything about decision tree classifiers. However, it should be noted that the recent work by Dumais et al. [1998] in which a version of decision tree classifiers was shown to perform nearly as well as their top performing system (a support vector machine classifier) will probably renew the interest in this type of text classifiers, an interest that had dwindled after previous unimpressive results reported in earlier literature [Cohen and Singer 1999; Joachims 1998; Lewis and Catlett 1994; Lewis and Ringuette 1994].

—By far the lowest performance is displayed by WORD, a "mock" classifier implemented by Yang [1999] and not including any learning component[16]. This unequivocably shows that learning techniques are definitely the way to go for automatic TC.

Concerning this last point, for completeness we should recall that one of the highest performances reported in the literature for the Reuters collection (a .90 breakeven) belongs to CONSTRUE, a manually constructed classifier. Unfortunately, this classifier has never been tested on the *standard* variants of Reuters mentioned in Table 6, and it is not clear [Yang 1999] whether the (small) test set of Reuters-22173 "ModHayes" on which the .90 breakeven value was obtained was chosen randomly, as

[16]WORD is based on the comparison between documents and category names, each treated as a vector of weighted terms in the vector space model. WORD was implemented by Yang with the only purpose of determining the difference in effectiveness that adding a learning component to a classifier brings about. WORD is actually called STR in [Yang 1994; Yang and Chute 1994]. Another non-learning classifier is proposed in [Wong et al. 1996].

safe scientific practice would demand. As a consequence, the fact that this figure may be indicative of the performance of CONSTRUE, and of the manual approach it represents, has been convincingly questioned [Yang 1999].

It is however important to bear in mind that the considerations above are not absolute and final judgments (if there may be any) on the comparative effectiveness of these TC methods. One of the reasons is that a particular applicative context may exhibit very different characteristics from the ones to be found in Reuters, and different classifiers may respond differently to these characteristics. An experimental study by Joachims [1998] involving support vector machines, k-NN, decision trees, Rocchio and Naïve Bayes, showed all these classifiers to have a similar effectiveness on categories with ≥ 300 positive training examples per category. The fact that this experiment involved the methods which have scored best (support vector machines, k-NN) and worst (Rocchio and Naïve Bayes) according to Table 6 results is indicative of the fact that conditions different from those of Reuters may very well invalidate conclusions drawn on this latter.

Finally, a note is worth about statistical significance testing. Very few authors have gone to the trouble of validating their experimental results by means of such tests. These tests are useful for verifying how strongly the experimental results support the claim that a given system Φ' is better than another system Φ'', or for verifying how much a difference in the experimental setup affects the measured effectiveness of a system Φ. Hull [1994] and Schütze et al. [1995] have been among the first to work in this direction, validating their results by means of the ANOVA test and the Friedman test; the former is aimed at determining the significance of the difference in effectiveness between two methods in terms of the ratio between this difference and the effectiveness variability across categories, while the latter conducts a similar test by using instead the rank positions of each method within a category. Yang and Liu [1999] define a full suite of significance tests, some of which apply to microaveraged and some to macroaveraged effectiveness. They apply them systematically to the comparison between five different classifiers, and are thus able to infer fine-grained conclusions about their relative effectiveness. For other examples of significance testing in TC see [Cohen 1995a; Cohen 1995b; Cohen and Hirsch 1998; Joachims 1997; Koller and Sahami 1997; Lewis et al. 1996; Wiener et al. 1995].

9. AUTOMATIC CATEGORISATION OF WEB PAGES

One recent application of TC which has raised considerable interest is the categorisation of Web pages, or sites, into hierarchically organised sets of categories. The reason why we discuss it separately from the rest of the paper is that it has given rise to a whole set of techniques specific to it, tackling problems as diverse as indexing, term selection, classifier induction and evaluation. One of the reasons this application has given rise to specific techniques is that Web pages are special kinds of documents, as they consist not only of a text but also of a set of incoming and outgoing pointers.

9.1 Indexing and dimensionality reduction

Attardi et al. [1999] propose an indexing technique specific to Web documents which is based on the notion of the *blurb* of a document (see Figure 7). Given a test doc-

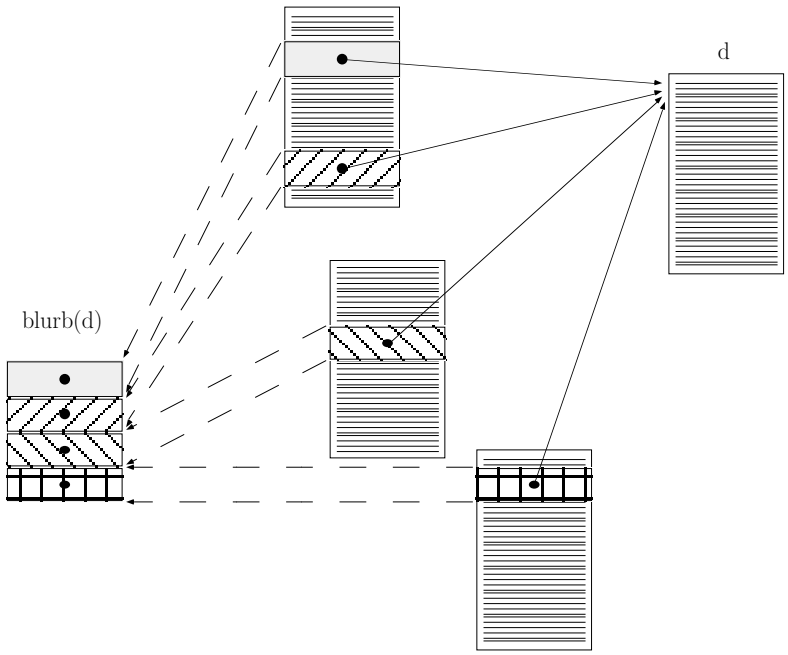

Fig. 7. The blurb of document d.

ument d_j, $blurb(d_j)$ is another "artificial" document formed of the juxtaposition of the text windows $w_1, \ldots, w_n$ containing hypertextual pointers from other documents $d_1, \ldots, d_n$ to d_j ($j \notin \{1, \ldots, n\}$), and is thus akin to a "compilation of short reviews" that Web authors have written on d_j. The authors claim that the blurb of d_j is a denser, and often more faithful representation of d_j's content than d_j itself. Document d_j is classified by means of a traditional learning method (Rocchio), with the only difference that $blurb(d_j)$, instead of d_j itself, is indexed, by means of a traditional text indexing method ($tfidf$), in order to obtain a representation of d_j.

Fuhr et al. [1998] propose instead another variant on the traditional indexing approach, as they obtain a representation of document d_j from the combined (standard) indexing of both d_j and the documents (directly) pointed by d_j. This choice is dictated by the fact that, rather than in Web *page* categorisation, we are often interested in Web *site* categorisation. Many "root" pages of interesting Web sites are actually just collections of entry points to children pages in a tree-shaped site. By using a traditional indexing method a classifier might easily dismiss these root pages as contentless, while by using this "radius ≤ 1" method their subordinate pages contribute to define their meaning.

Koller and Sahami [1997] propose a new method for organising term selection and classifier induction which is specifically addressed to the situation in which the categories are hierarchically organised. According to their method, for each *internal* node c_i of the hierarchy (starting from the root) a different set of r' terms is selected based on the training documents belonging to c_i. For category c_i, the r'

terms are selected that are the best in discriminating among the children categories $c_{i1}, \ldots, c_{is}$ of c_i. Out of the r' terms selected for c_i a classifier Φ_i is then built that is able to decide under which among categories $c_{i1}, \ldots, c_{is}$ a document classified under c_i can be further classified. Note that this is different from "local dimensionality reduction" as discussed in Section 5. In fact, in that case the r' terms selected for c_i were meant to support the decision whether to classify documents under c_i or not, while in this case these terms are meant to support the decision under which among $c_{i1}, \ldots, c_{is}$ a document classified under c_i can be further classified. In this way, each document d_j is classified under one and only one "leaf" category by being first submitted to the classifier Φ_r associated to the root node c_r, then to the classifier associated to the first level internal node under which Φ_r has classified d_j, and so on; in general, the classification of each document involves then the invocation of l classifiers, where l is the length of the path between the root node and the leaf node. By means of this method, the authors are able to decompose the classification task into a number of much simpler classification tasks, and radically reduce the number r' of terms on which each individual classifier must be based. This latter factor means being able to avoid overfitting more easily and to afford using more sophisticated classifiers (whose complexity usually depends on the number of terms used) at each individual step.

9.2 Classifier induction

Chakrabarti et al. [1998] propose a novel method for the induction of a Web page classifier. Their ARC system for automatic Web page categorisation is based on the hypothesis that to each pair $\langle d_j, c_i \rangle$ two parameters may be associated: its *authority value* $a(d_j, c_i)$, which measures the "authoritativeness" of d_j on c_i in terms of the number of c_i-related pages pointing to it, and its *hub value* $h(d_j, c_i)$, which measures the "informativeness" of d_j on c_i in terms of the number of c_i-related pages it points to. The purpose of ARC is thus to identify, given a topic c_i, the k most authoritative and k most informative Web pages that deal with it; this may thus be seen as a specialisation of category-pivoted categorisation, since this latter just aims at finding the k Web pages most relevant to c_i. ARC is "kick-started" by issuing query c_i to the ALTAVISTA search engine, which provides an initial set of documents presumably relevant to c_i (the 200 top-ranked ones are picked); an expansion phase follows, in which documents distant ≤ 2 steps from d_j in the forward citation graph are added to the set. An iterative algorithm is then applied in which, at each iteration, for each page d_j the $a(d_j, c_i)$ value is substituted by the sum of the $h(d_x, c_i)$ values of the pages d_x pointing to it, and $h(d_j, c_i)$ is substituted by the sum of the $a(d_y, c_i)$ values of the pages d_y to which it points. This formalizes the intuition that the hub value of a page is high if the page points to many c_i-*authoritative* pages, and that the the authority value of a page is high if the page is referred by many c_i-*informative* pages. The authors report that for $k = 15$ the algorithm usually near-converges (i.e. the identity of the top 15 authority and hub pages stabilizes) in approximately 5 iterations.

A novel method for the induction of classifiers for a hierarchically structured category set is proposed by Ruiz and Srinivasan [1999]. Their system is composed of a hierarchically structured set of *gating networks* and *expert networks*. A gating network for category c_i is a neural network that decides whether document d_j might

be a plausible candidate for categorisation under any children categories of c_i; if this decision is positive, d_j is propagated to *all* the children nodes of c_i, while if the decision is negative the document is not propagated to any of them. An expert network for category c_i is instead a neural network that, quite traditionally, decides whether document d_j should be classified under c_j. While a leaf node consists of an expert network only, an internal node always includes a gating network, and possibly includes an expert network too. When a document is received by a node c_i, it is processed by whichever of gating and/or expert network is present. This allows a document d_j to be classified, if desired, under internal nodes and leaf nodes at the same time, which may desirable in the case in which internal nodes represent meaningful higher level concepts. In experiments performed by the authors, this system yielded a significant increase in effectiveness over a "flat" neural network built with the same technology as the individual gating and expert networks.

9.3 Evaluation

Larkey and Croft [1996, Section 2.6], although not working in a Web context, define an effectiveness measure explicitly geared towards the case of hierarchically structured categories, hence apt to be used in the Web case. Their idea is to redefine effectiveness by considering not only success ($a_{ij} = ca_{ij}$, in the notation of Section 4.1) and failure ($a_{ij} \neq ca_{ij}$) but also *near success*; this is defined as the case in which a category c_i attributed by the system to d_j is not correct, but a category c_k *sibling* of c_i would have been correct.

10. CONCLUSION

Automated TC has now established itself as one of the major areas in the information systems discipline, because of a number of reasons:

—Its domains of application are numerous and important, and given the proliferation of documents in digital form they are bound to increase dramatically in both number and importance.

—It is indispensable in many applications in which the sheer number of the documents to be classified and the short response time imposed by the application make the manual alternative implausible.

—It can dramatically improve the productivity of human classifiers in those applications in which no classification decision can be taken without a final human expert judgment [Larkey and Croft 1996], by providing tools that quickly "suggest" plausible decisions.

—It has reached effectiveness levels comparable to those obtained in manual TC with the involvement of trained professionals. The effectiveness of manual TC is not 100% anyway [Cleverdon 1984] and, perhaps more importantly, it is not going to be improved by the progress of research. On the contrary, the levels of effectiveness of automated TC are growing at a steady pace, and even if it is plausible to hypothesise that they will eventually reach a plateau well below the 100% level, this plateau will probably be higher that the effectiveness levels of manual TC.

One of the reasons why from the early '90s onwards the effectiveness levels of text classifiers have dramatically improved, is the entrance in the TC arena of machine

learning methods that are backed by strong theoretical motivations. Examples of these are multiplicative weight updating (e.g. the WINNOW family, WIDROW-HOFF, etc.), adaptive resampling (e.g. boosting) and support vector machines, which provide a sharp contrast with relatively unsophisticated and theoretically weak methods such as Rocchio. In TC, the field of machine learning has found a challenging application, since datasets consisting of hundreds of thousands of documents and characterised by tens of thousands of terms are widely available. This means that TC is a good benchmark for checking whether a given learning technique, rather than just being applicable to fairly undemanding domains, can scale up to substantial sizes. In turn, this probably means that the active involvement of the machine learning community in the automated TC discipline is bound to grow.

The success story of automated TC is also going to encourage an extension of its methods and techniques to neighbouring fields of applications. Techniques typical of automated TC have already been extended successfully to the automated categorisation of documents expressed in slightly different media; for instance:

—the classification of very noisy text resulting from optical character recognition, tackled by Ittner et al. [1995]. These authors have proven how, by employing noisy texts also in the training phase (i.e. texts affected by the same source of noise that is also at work in the test documents), effectiveness levels comparable to those obtainable in the case of standard text can be achieved.

—the classification of speech transcripts, tackled by Schapire and Singer [2000]. In this work, the authors classify answers given to a phone operator's request "`How may I help you?`", so as to be able to route the call to a specialised operator according to call type.

Concerning other more radically different media, the current situation is not as bright. Current research on automatic document categorisation under *thematic* categories mostly focuses on the text medium (however, see [Lim 1999] for an interesting attempt at image categorisation based on a textual metaphor). The reason for this is that capturing real semantic content in the automatic indexing of non-textual media is still an open research problem. While there are systems that attempt to detect content e.g. in images by recognising shapes, colour distributions and texture, the general problem of image semantics is still unsolved. The main reason for this fact is that natural language, the language of the text medium, admits far fewer variations than the "languages" employed by the other media. For instance, while the concept of a house can be "triggered" by relatively few natural language expressions such as `house`, `houses`, `home`, `housing`, `inhabiting`, etc., it can be triggered by *far more* images: the images of all the different houses that exist, of all possible colours and shapes, viewed from all the possible perspectives, from all the possible distances, etc. If we had solved the multimedia indexing problem in a satisfactory way, the general methodology (i.e. classifier induction, experimental evaluation, etc.) that we have discussed in this paper for text would also apply to automated multimedia categorisation, and there are reasons to believe that the effectiveness levels could be as high. This only adds to the common sentiment that more research in automated content-based indexing for multimedia documents is needed.

Acknowledgements

Thanks to Umberto Straccia for many useful comments on an earlier draft.

REFERENCES

Amati, G. and Crestani, F. 1999. Probabilistic learning for selective dissemination of information. *Information Processing and Management 35*, 5, 633–654.

Apté, C., Damerau, F. J., and Weiss, S. M. 1994. Automated learning of decision rules for text categorization. *ACM Transactions on Information Systems 12*, 3, 233–251.

Attardi, G., Gullí, A., and Sebastiani, F. 1999. Automatic Web page categorization by link and context analysis. In C. Hutchison and G. Lanzarone Eds., *Proceedings of THAI-99, European Symposium on Telematics, Hypermedia and Artificial Intelligence* (Varese, IT, 1999), pp. 105–119.

Baker, L. D. and McCallum, A. K. 1998. Distributional clustering of words for text categorisation. In *Proceedings of SIGIR-98, 21st ACM International Conference on Research and Development in Information Retrieval* (Melbourne, AU, 1998), pp. 96–103.

Belkin, N. J. and Croft, W. B. 1992. Information filtering and information retrieval: two sides of the same coin? *Communications of the ACM 35*, 12, 29–38.

Biebricher, P., Fuhr, N., Lustig, G., and Schwantner, M. 1988. The automatic indexing system AIR/PHYS. From research to application. In *Proceedings of SIGIR-88, 11th ACM International Conference on Research and Development in Information Retrieval* (Grenoble, FR, 1988), pp. 333–342. Also reprinted in [Sparck Jones and Willett 1997], pp. 513–517.

Blosseville, M., Hebrail, G., Montell, M., and Penot, N. 1992. Automatic document classification: natural langage processing and expert system techniques used together. In *Proceedings of SIGIR-92, 15th ACM International Conference on Research and Development in Information Retrieval* (Kobenhavn, DK, 1992), pp. 51–57.

Borko, H. and Bernick, M. 1963. Automatic document classification. *Journal of the Association for Computing Machinery 10*, 1, 151–161.

Carbonell, J., Cohen, W. W., and Yang, Y. 2000. Guest editorial for the special issue on text categorization. *Machine Learning*. Forthcoming.

Chakrabarti, S., Dom, B., Raghavan, P., Rajagopalan, S., Gibson, D., and Kleinberg, J. 1998. Automatic resource compilation by analyzing hyperlink structure and associated text. *Computer Networks and ISDN Systems 30*, 1-7, 65–74.

Clack, C., Farringdon, J., Lidwell, P., and Yu, T. 1997. Autonomous document classification for business. In *Proceedings of the 1st International Conference on Autonomous Agents* (Marina del Rey, US, 1997), pp. 201–208.

Cleverdon, C. 1984. Optimizing convenient online access to bibliographic databases. *Information Services and Use 4*, 1, 37–47. Also reprinted in [Willett 1988a], pp. 32–41.

Cohen, W. W. 1995a. Learning to classify English text with ILP methods. In L. De Raedt Ed., *Advances in inductive logic programming*. Amsterdam, NL: IOS Press.

Cohen, W. W. 1995b. Text categorization and relational learning. In *Proceedings of ICML-95, 12th International Conference on Machine Learning* (Lake Tahoe, US, 1995), pp. 124–132.

Cohen, W. W. 1996. Learning rules that classify e-mail. In *Proceedings of the AAAI Spring Symposium on Machine Learning in Information Access* (Palo Alto, US, 1996), pp. 18–25.

Cohen, W. W. and Hirsch, H. 1998. Joins that generalize: text classification using Whirl. In *Proceedings of KDD-98, 4th International Conference on Knowledge Discovery and Data Mining* (New York, US, 1998), pp. 169–173.

Cohen, W. W. and Singer, Y. 1999. Context-sensitive learning methods for text categorization. *ACM Transactions on Information Systems 17*, 2, 141–173.

Cooper, W. S. 1995. Some inconsistencies and misnomers in probabilistic information retrieval. *ACM Transactions on Information Systems 13*, 1, 100–111.

Creecy, R. M., Masand, B. M., Smith, S. J., and Waltz, D. L. 1992. Trading MIPS and memory for knowledge engineering: classifying census returns on the Connection Machine. *Communications of the ACM 35*, 8, 48–63.

Crestani, F., Lalmas, M., van Rijsbergen, C. J., and Campbell, I. 1998. "Is this document relevant? ... probably". A survey of probabilistic models in information retrieval. *ACM Computing Surveys 30*, 4, 528–552.

Dagan, I., Karov, Y., and Roth, D. 1997. Mistake-driven learning in text categorization. In *Proceedings of the 2nd Conference on Empirical Methods in Natural Language Processing* (Providence, US, 1997), pp. 55–63.

Deerwester, S., Dumais, S. T., Furnas, G. W., Landauer, T. K., and Harshman, R. 1990. Indexing by latent semantic indexing. *Journal of the American Society for Information Science 41*, 6, 391–407.

Domingos, P. and Pazzani, M. J. 1997. On the the optimality of the simple Bayesian classifier under zero-one loss. *Machine Learning 29*, 2-3, 103–130.

Dumais, S. T., Platt, J., Heckerman, D., and Sahami, M. 1998. Inductive learning algorithms and representations for text categorization. In *Proceedings of CIKM-98, 7th ACM International Conference on Information and Knowledge Management* (Washington, US, 1998), pp. 148–155.

Fangmeyer, H. and Lustig, G. 1968. The EURATOM automatic indexing project. In *Proceedings of the IFIP Congress (Booklet J)* (Edinburgh, UK, 1968), pp. 66–70.

Field, B. 1975. Towards automatic indexing: automatic assignment of controlled-language indexing and classification from free indexing. *Journal of Documentation 31*, 4, 246–265.

Forsyth, R. S. 1999. New directions in text categorization. In A. Gammerman Ed., *Causal models and intelligent data management*, pp. 151–185. Heidelberg, DE: Springer.

Fuhr, N. 1985. A probabilistic model of dictionary-based automatic indexing. In *Proceedings of RIAO-85, 1st International Conference "Recherche d'Information Assistee par Ordinateur"* (Grenoble, FR, 1985), pp. 207–216.

Fuhr, N. 1989. Models for retrieval with probabilistic indexing. *Information Processing and Management 25*, 1, 55–72.

Fuhr, N. and Buckley, C. 1991. A probabilistic learning approach for document indexing. *ACM Transactions on Information Systems 9*, 3, 223–248.

Fuhr, N., Gövert, N., Lalmas, M., and Sebastiani, F. 1998. Categorisation tool: Final prototype. Deliverable 4.3, Project LE4-8303 "EUROSEARCH", Commission of the European Communities, 1999.

Fuhr, N., Hartmann, S., Knorz, G., Lustig, G., Schwantner, M., and Tzeras, K. 1991. AIR/X – a rule-based multistage indexing system for large subject fields. In *Proceedings of RIAO-91, 3rd International Conference "Recherche d'Information Assistee par Ordinateur"* (Barcelona, ES, 1991), pp. 606–623.

Fuhr, N. and Knorz, G. 1984. Retrieval test evaluation of a rule-based automated indexing (AIR/PHYS). In *Proceedings of SIGIR-84, 7th ACM International Conference on Research and Development in Information Retrieval* (1984), pp. 391–408.

Galavotti, L. 1999. Un sistema modulare per la classificazione di testi basato sull'apprendimento automatico. Master's thesis, Dipartimento di Informatica, Università di Pisa, Pisa, IT.

Gale, W. A., Church, K. W., and Yarowsky, D. 1993. A method for disambiguating word senses in a large corpus. *Computers and the Humanities 26*, 5, 415–439.

Goodman, M. 1990. Prism: a case-based telex classifier. In *Proceedings of IAAI-90, 2nd Conference on Innovative Applications of Artificial Intelligence* (1990), pp. 86–90.

Gray, W. A. and Harley, A. J. 1971. Computer-assisted indexing. *Information Storage and Retrieval 7*, 4, 167–174.

Guthrie, L., Walker, E., and Guthrie, J. A. 1994. Document classification by machine: theory and practice. In *Proceedings of COLING-94, 15th International Conference on Computational Linguistics* (Kyoto, JP, 1994), pp. 1059–1063.

Hamill, K. A. and Zamora, A. 1978. An automatic document classification system using pattern recognition techniques. In *Proceedings of ASIS-78, 41st Annual Meeting of the American Society for Information Science* (New York, US, 1978), pp. 152–155.

Hamill, K. A. and Zamora, A. 1980. The use of titles for automatic document classification. *Journal of the American Society for Information Science 33*, 6, 396–402.

Hayes, P. J., Andersen, P. M., Nirenburg, I. B., and Schmandt, L. M. 1990. Tcs: a shell for content-based text categorization. In *Proceedings of CAIA-90, 6th IEEE Conference on Artificial Intelligence Applications* (Santa Barbara, US, 1990), pp. 320–326.

Heaps, H. 1973. A theory of relevance for automatic document classification. *Information and Control 22*, 3, 268–278.

Hearst, M. A. 1991. Noun homograph disambiguation using local context in large corpora. In *Proceedings of the 7th Annual Conference of the University of Waterloo Centre for the New Oxford English Dictionary* (Oxford, UK, 1991), pp. 1–22.

Hersh, W., Buckley, C., Leone, T., and Hickman, D. 1994. Ohsumed: an interactive retrieval evaluation and new large text collection for research. In *Proceedings of SIGIR-94, 17th ACM International Conference on Research and Development in Information Retrieval* (Dublin, IE, 1994), pp. 192–201.

Hoyle, W. G. 1973. Automatic indexing and generation of classification by algorithm. *Information Storage and Retrieval 9*, 4, 233–242.

Hull, D. A. 1994. Improving text retrieval for the routing problem using latent semantic indexing. In *Proceedings of SIGIR-94, 17th ACM International Conference on Research and Development in Information Retrieval* (Dublin, IE, 1994), pp. 282–289.

Hull, D. A. 1998. The TREC-7 filtering track: description and analysis. In *Proceedings of TREC-7, 7th Text Retrieval Conference* (Gaithersburg, US, 1998), pp. 33–56.

Hull, D. A., Pedersen, J. O., and Schütze, H. 1996. Method combination for document filtering. In *Proceedings of SIGIR-96, 19th ACM International Conference on Research and Development in Information Retrieval* (Zürich, CH, 1996), pp. 279–288.

Ittner, D. J., Lewis, D. D., and Ahn, D. D. 1995. Text categorization of low quality images. In *Proceedings of SDAIR-95, 4th Annual Symposium on Document Analysis and Information Retrieval* (Las Vegas, US, 1995), pp. 301–315.

Iwayama, M. and Tokunaga, T. 1995. Cluster-based text categorization: a comparison of category search strategies. In *Proceedings of SIGIR-95, 18th ACM International Conference on Research and Development in Information Retrieval* (Seattle, US, 1995), pp. 273–281.

Joachims, T. 1997. A probabilistic analysis of the Rocchio algorithm with TFIDF for text categorization. In *Proceedings of ICML-97, 14th International Conference on Machine Learning* (Nashville, US, 1997), pp. 143–151.

Joachims, T. 1998. Text categorization with support vector machines: learning with many relevant features. In *Proceedings of ECML-98, 10th European Conference on Machine Learning* (Chemnitz, DE, 1998), pp. 137–142.

Joachims, T. 1999. Transductive inference for text classification using support vector machines. In *Proceedings of ICML-99, 16th International Conference on Machine Learning* (Bled, SL, 1999).

John, G., Kohavi, R., and Pfleger, K. 1994. Irrelevant features and the subset selection problem. In *Proceedings of ICML-94, 11th International Conference on Machine Learning* (New Brunswick, US, 1994), pp. 121–129.

Klingbiel, P. H. 1973a. Machine-aided indexing of technical literature. *Information Storage and Retrieval 9*, 2, 79–84.

Klingbiel, P. H. 1973b. A technique for machine-aided indexing. *Information Storage and Retrieval 9*, 9, 477–494.

Koller, D. and Sahami, M. 1997. Hierarchically classifying documents using very few words. In *Proceedings of ICML-97, 14th International Conference on Machine Learning* (Nashville, US, 1997), pp. 170–178.

KORFHAGE, R. R. 1997. *Information storage and retrieval*. Wiley Computer Publishing, New York, US.

LAM, W. AND HO, C. Y. 1998. Using a generalized instance set for automatic text categorization. In *Proceedings of SIGIR-98, 21st ACM International Conference on Research and Development in Information Retrieval* (Melbourne, AU, 1998), pp. 81–89.

LAM, W., LOW, K. F., AND HO, C. Y. 1997. Using a Bayesian network induction approach for text categorization. In *Proceedings of IJCAI-97, 15th International Joint Conference on Artificial Intelligence* (Nagoya, JP, 1997), pp. 745–750.

LAM, W., RUIZ, M. E., AND SRINIVASAN, P. 1999. Automatic text categorization and its applications to text retrieval. *IEEE Transactions on Knowledge and Data Engineering*. Forthcoming.

LANG, K. 1995. NEWSWEEDER: learning to filter netnews. In *Proceedings of ICML-95, 12th International Conference on Machine Learning* (Lake Tahoe, US, 1995), pp. 331–339.

LARKEY, L. S. 1998. Automatic essay grading using text categorization techniques. In *Proceedings of SIGIR-98, 21st ACM International Conference on Research and Development in Information Retrieval* (Melbourne, AU, 1998), pp. 90–95.

LARKEY, L. S. 1999. A patent search and classification system. In *Proceedings of DL-99, 4th ACM Conference on Digital Libraries* (Berkeley, US, 1999), pp. 179–187.

LARKEY, L. S. AND CROFT, W. B. 1996. Combining classifiers in text categorization. In *Proceedings of SIGIR-96, 19th ACM International Conference on Research and Development in Information Retrieval* (Zürich, CH, 1996), pp. 289–297.

LEWIS, D. D. 1992. An evaluation of phrasal and clustered representations on a text categorization task. In *Proceedings of SIGIR-92, 15th ACM International Conference on Research and Development in Information Retrieval* (Kobenhavn, DK, 1992), pp. 37–50.

LEWIS, D. D. 1995a. Evaluating and optmizing autonomous text classification systems. In *Proceedings of SIGIR-95, 18th ACM International Conference on Research and Development in Information Retrieval* (Seattle, US, 1995), pp. 246–254.

LEWIS, D. D. 1995b. A sequential algorithm for training text classifiers: corrigendum and additional data. *SIGIR Forum 29*, 2, 13–19.

LEWIS, D. D. 1995c. The TREC-4 filtering track: description and analysis. In *Proceedings of TREC-4, 4th Text Retrieval Conference* (Gaithersburg, US, 1995), pp. 165–180.

LEWIS, D. D. 1998. Naive (Bayes) at forty: The independence assumption in information retrieval. In *Proceedings of ECML-98, 10th European Conference on Machine Learning* (Chemnitz, DE, 1998), pp. 4–15.

LEWIS, D. D. AND CATLETT, J. 1994. Heterogeneous uncertainty sampling for supervised learning. In *Proceedings of ICML-94, 11th International Conference on Machine Learning* (New Brunswick, US, 1994), pp. 148–156.

LEWIS, D. D. AND GALE, W. A. 1994. A sequential algorithm for training text classifiers. In *Proceedings of SIGIR-94, 17th ACM International Conference on Research and Development in Information Retrieval* (Dublin, IE, 1994), pp. 3–12. See also [Lewis 1995b].

LEWIS, D. D. AND HAYES, P. J. 1994. Guest editorial for the special issue on text categorization. *ACM Transactions on Information Systems 12*, 3, 231.

LEWIS, D. D. AND RINGUETTE, M. 1994. A comparison of two learning algorithms for text categorization. In *Proceedings of SDAIR-94, 3rd Annual Symposium on Document Analysis and Information Retrieval* (Las Vegas, US, 1994), pp. 81–93.

LEWIS, D. D., SCHAPIRE, R. E., CALLAN, J. P., AND PAPKA, R. 1996. Training algorithms for linear text classifiers. In *Proceedings of SIGIR-96, 19th ACM International Conference on Research and Development in Information Retrieval* (Zürich, CH, 1996), pp. 298–306.

LEWIS, D. D., STERN, D. L., AND SINGHAL, A. 1999. ATTICS: a software platform for online text classification. In *Proceedings of SIGIR-99, 22nd ACM International Conference on Research and Development in Information Retrieval* (Berkeley, US, 1999), pp. 267–268.

LI, H. AND YAMANISHI, K. 1999. Text classification using ESC-based stochastic decision lists. In *Proceedings of CIKM-99, 8th ACM International Conference on Information and Knowledge Management* (Kansas City, US, 1999), pp. 122–130.

Li, Y. H. and Jain, A. K. 1998. Classification of text documents. *The Computer Journal 41*, 8, 537–546.

Liddy, E. D., Paik, W., and Yu, E. S. 1994. Text categorization for multiple users based on semantic features from a machine-readable dictionary. *ACM Transactions on Information Systems 12*, 3, 278–295.

Liere, R. and Tadepalli, P. 1997. Active learning with committees for text categorization. In *Proceedings of AAAI-97, 14th Conference of the American Association for Artificial Intelligence* (Providence, US, 1997), pp. 591–596.

Lim, J. H. 1999. Learnable visual keywords for image classification. In *Proceedings of DL-99, 4th ACM Conference on Digital Libraries* (Berkeley, US, 1999), pp. 139–145.

Maron, M. 1961. Automatic indexing: an experimental inquiry. *Journal of the Association for Computing Machinery 8*, 3, 404–417.

Masand, B., Linoff, G., and Waltz, D. 1992. Classifying news stories using memory-based reasoning. In *Proceedings of SIGIR-92, 15th ACM International Conference on Research and Development in Information Retrieval* (Kobenhavn, DK, 1992), pp. 59–65.

McCallum, A. K. and Nigam, K. 1998. Employing EM in pool-based active learning for text classification. In *Proceedings of ICML-98, 15th International Conference on Machine Learning* (Madison, US, 1998), pp. 350–358.

McCallum, A. K., Rosenfeld, R., Mitchell, T. M., and Ng, A. Y. 1998. Improving text classification by shrinkage in a hierarchy of classes. In *Proceedings of ICML-98, 15th International Conference on Machine Learning* (Madison, US, 1998), pp. 359–367.

Mitchell, T. M. 1996. *Machine learning*. McGraw Hill, New York, US.

Mladenić, D. 1998a. Feature subset selection in text learning. In *Proceedings of ECML-98, 10th European Conference on Machine Learning* (Chemnitz, DE, 1998), pp. 95–100.

Mladenić, D. 1998b. Turning Yahoo! into an automatic Web page classifier. In *Proceedings of ECAI-98, 13th European Conference on Artificial Intelligence* (Brighton, UK, 1998), pp. 473–474.

Moulinier, I. and Ganascia, J.-G. 1996. Applying an existing machine learning algorithm to text categorization. In S. Wermter, E. Riloff, and G. Scheler Eds., *Connectionist, statistical, and symbolic approaches to learning for natural language processing* (Heidelberg, DE, 1996), pp. 343–354. Springer Verlag. Published in the "Lecture Notes for Computer Science" series, number 1040.

Moulinier, I., Raškinis, G., and Ganascia, J.-G. 1996. Text categorization: a symbolic approach. In *Proceedings of SDAIR-96, 5th Annual Symposium on Document Analysis and Information Retrieval* (Las Vegas, US, 1996).

Ng, H. T., Goh, W. B., and Low, K. L. 1997. Feature selection, perceptron learning, and a usability case study for text categorization. In *Proceedings of SIGIR-97, 20th ACM International Conference on Research and Development in Information Retrieval* (Philadelphia, US, 1997), pp. 67–73.

Nigam, K., McCallum, A. K., Thrun, S., and Mitchell, T. M. 1998. Learning to classify text from labeled and unlabeled documents. In *Proceedings of AAAI-98, 15th Conference of the American Association for Artificial Intelligence* (Madison, US, 1998), pp. 792–799. An extended version is forthcoming on *Machine Learning*.

Ragas, H. and Koster, C. H. 1998. Four text classification algorithms compared on a Dutch corpus. In *Proceedings of SIGIR-98, 21st ACM International Conference on Research and Development in Information Retrieval* (Melbourne, AU, 1998), pp. 369–370.

Rau, L. F. and Jacobs, P. S. 1991. Creating segmented databases from free text for text retrieval. In *Proceedings of SIGIR-91, 14th ACM International Conference on Research and Development in Information Retrieval* (Chicago, US, 1991), pp. 337–346.

Riloff, E. and Lehnert, W. 1994. Information extraction as a basis for high-precision text classification. *ACM Transactions on Information Systems 12*, 3, 296–333.

Robertson, S. E. and Harding, P. 1984. Probabilistic automatic indexing by learning from human indexers. *Journal of Documentation 40*, 4, 264–270.

ROBERTSON, S. E. AND SPARCK JONES, K. 1976. Relevance weighting of search terms. *Journal of the American Society for Information Science 27*, 3, 129–146. Also reprinted in [Willett 1988a], pp. 143–160.

ROTH, D. 1998. Learning to resolve natural language ambiguities: a unified approach. In *Proceedings of AAAI-98, 15th Conference of the American Association for Artificial Intelligence* (Madison, US, 1998), pp. 806–813.

RUIZ, M. E. AND SRINIVASAN, P. 1999. Hierarchical neural networks for text categorization. In *Proceedings of SIGIR-99, 22nd ACM International Conference on Research and Development in Information Retrieval* (Berkeley, US, 1999), pp. 281–282.

SABLE, C. L. AND HATZIVASSILOGLOU, V. 1999. Text-based approaches for the categorization of images. In *Proceedings of ECDL-99, 3rd European Conference on Research and Advanced Technology for Digital Libraries* (Paris, FR, 1999), pp. 19–38.

SALTON, G. AND BUCKLEY, C. 1988. Term-weighting approaches in automatic text retrieval. *Information Processing and Management 24*, 5, 513–523. Also reprinted in [Sparck Jones and Willett 1997], pp. 323–328.

SALTON, G., WONG, A., AND YANG, C. 1975. A vector space model for automatic indexing. *Communications of the ACM 18*, 11, 613–620. Also reprinted in [Sparck Jones and Willett 1997], pp. 273–280.

SARACEVIC, T. 1975. Relevance: a review of and a framework for the thinking on the notion in information science. *Journal of the American Society for Information Science 26*, 6, 321–343. Also reprinted in [Sparck Jones and Willett 1997], pp. 143–165.

SCHAPIRE, R. E. AND SINGER, Y. 2000. BOOSTEXTER: a boosting-based system for text categorization. *Machine Learning*. Forthcoming.

SCHAPIRE, R. E., SINGER, Y., AND SINGHAL, A. 1998. Boosting and Rocchio applied to text filtering. In *Proceedings of SIGIR-98, 21st ACM International Conference on Research and Development in Information Retrieval* (Melbourne, AU, 1998), pp. 215–223.

SCHÜTZE, H. 1998. Automatic word sense discrimination. *Computational Linguistics 24*, 1, 97–124.

SCHÜTZE, H., HULL, D. A., AND PEDERSEN, J. O. 1995. A comparison of classifiers and document representations for the routing problem. In *Proceedings of SIGIR-95, 18th ACM International Conference on Research and Development in Information Retrieval* (Seattle, US, 1995), pp. 229–237.

SINGHAL, A., MITRA, M., AND BUCKLEY, C. 1997. Learning routing queries in a query zone. In *Proceedings of SIGIR-97, 20th ACM International Conference on Research and Development in Information Retrieval* (Philadelphia, US, 1997), pp. 25–32.

SPARCK JONES, K. AND WILLETT, P. Eds. 1997. *Readings in information retrieval*. Morgan Kaufmann, San Mateo, US.

TAIRA, H. AND HARUNO, M. 1999. Feature selection in SVM text categorization. In *Proceedings of AAAI-99, 16th Conference of the American Association for Artificial Intelligence* (Orlando, US, 1999), pp. 480–486.

TZERAS, K. AND HARTMANN, S. 1993. Automatic indexing based on Bayesian inference networks. In *Proceedings of SIGIR-93, 16th ACM International Conference on Research and Development in Information Retrieval* (Pittsburgh, US, 1993), pp. 22–34.

VAN RIJSBERGEN, C. J. 1977. A theoretical basis for the use of co-occurrence data in information retrieval. *Journal of Documentation 33*, 2, 106–119.

WEIGEND, A. S., WIENER, E. D., AND PEDERSEN, J. O. 1999. Exploiting hierarchy in text categorization. *Information Retrieval 1*, 3, 193–216.

WEISS, S. M., APTE, C., DAMERAU, F. J., JOHNSON, D. E., OLES, F. J., GOETZ, T., AND HAMPP, T. 1999. Maximizing text-mining performance. *IEEE Intelligent Systems 14*, 4, 63–69.

WIENER, E., PEDERSEN, J. O., AND WEIGEND, A. S. 1995. A neural network approach to topic spotting. In *Proceedings of SDAIR-95, 4th Annual Symposium on Document Analysis and Information Retrieval* (Las Vegas, US, 1995), pp. 317–332.

WILLETT, P. Ed. 1988a. *Document retrieval systems*. Taylor Graham, London, UK.

WILLETT, P. 1988b. Recent trends in hierarchic document clustering: a critical review. *Information Processing and Management 24*, 5, 577–597.

WONG, J. W., KAN, W., AND YOUNG, G. 1996. ACTION: automatic classification for full-text documents. *SIGIR Forum 30*, 1, 26–41.

WONG, S. M. AND YAO, Y. 1995. On modeling information retrieval with probabilistic inference. *ACM Transactions on Information Systems 13*, 1, 38–68.

YANG, Y. 1994. Expert network: effective and efficient learning from human decisions in text categorisation and retrieval. In *Proceedings of SIGIR-94, 17th ACM International Conference on Research and Development in Information Retrieval* (Dublin, IE, 1994), pp. 13–22.

YANG, Y. 1995. Noise reduction in a statistical approach to text categorization. In *Proceedings of SIGIR-95, 18th ACM International Conference on Research and Development in Information Retrieval* (Seattle, US, 1995), pp. 256–263.

YANG, Y. 1999. An evaluation of statistical approaches to text categorization. *Information Retrieval 1*, 1-2, 69–90.

YANG, Y. AND CHUTE, C. G. 1994. An example-based mapping method for text categorization and retrieval. *ACM Transactions on Information Systems 12*, 3, 252–277.

YANG, Y. AND LIU, X. 1999. A re-examination of text categorization methods. In *Proceedings of SIGIR-99, 22nd ACM International Conference on Research and Development in Information Retrieval* (Berkeley, US, 1999), pp. 42–49.

YANG, Y. AND PEDERSEN, J. O. 1997. A comparative study on feature selection in text categorization. In *Proceedings of ICML-97, 14th International Conference on Machine Learning* (Nashville, US, 1997), pp. 412–420.