Machine Learning in Automated Text Categorization

Fabrizio Sebastiani
Consiglio Nazionale delle Ricerche, Italy

The automated categorization (or classification) of texts into predefined categories has witnessed a booming interest in the last ten years, due to the increased availability of documents in digital form and the ensuing need to organize them. In the research community the dominant approach to this problem is based on machine learning techniques: a general inductive process automatically builds a classifier by learning, from a set of preclassified documents, the characteristics of the categories. The advantages of this approach over the knowledge engineering approach (consisting in the manual definition of a classifier by domain experts) are a very good effectiveness, considerable savings in terms of expert manpower, and straightforward portability to different domains. This survey discusses the main approaches to text categorization that fall within the machine learning paradigm. We will discuss in detail issues pertaining to three different problems, namely document representation, classifier construction, and classifier evaluation.

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 categorization, text classification

1. INTRODUCTION

In the last ten years content-based document management tasks (collectively known as *information retrieval* – IR) have gained a prominent status in the information systems field, due to the increased availability of documents in digital form and the ensuing need to access them in flexible ways. *Text categorization* (TC – aka *text classification*, or *topic spotting*), the activity of labelling natural language texts with thematic categories from a predefined set, is one such task. TC dates back to the early '60s, but only in the early '90s it became a major subfield of the information systems discipline, thanks to increased applicative interest and to the availability of more powerful hardware. TC is now being applied in many contexts, ranging from document indexing based on a controlled vocabulary, to document filtering, automated metadata generation, word sense disambiguation, population of hierarchical catalogues of Web resources, and in general any application requiring document organization or selective and adaptive document dispatching.

Until the late '80s the most popular approach to TC, at least in the "operational" (i.e. real-world applications) community, was a *knowledge engineering* (KE) one, consisting in manually defining a set of rules encoding expert knowledge on

Address: Istituto di Elaborazione dell'Informazione, Consiglio Nazionale delle Ricerche, Via G. Moruzzi, 1, 56124 Pisa (Italy). E-mail: fabrizio@iei.pi.cnr.it

how to classify documents under the given categories. In the '90s this approach has increasingly lost popularity (especially in the research community) in favour of the *machine learning* (ML) paradigm, according to which a general inductive process automatically builds an automatic text classifier by learning, from a set of preclassified documents, the characteristics of the categories of interest. The advantages of this approach are an accuracy comparable to that achieved by human experts, and a considerable savings in terms of expert manpower, since no intervention from either knowledge engineers or domain experts is needed for the construction of the classifier or for its porting to a different set of categories. It is the ML approach to TC that this paper concentrates on.

Current-day TC is thus a discipline at the crossroads of ML and IR, and as such it shares a number of characteristics with other tasks such as *information/knowledge extraction from texts* and *text mining* [Knight 1999; Pazienza 1997]. There is still considerable debate on where the exact border between these disciplines lies, and the terminology is still evolving. "Text mining" is increasingly being used to denote all the tasks that, by analyzing large quantities of text and detecting usage patterns, try to extract probably useful (although only probably correct) information. According to this view, TC is an instance of text mining. TC enjoys quite a rich literature now, but this is still fairly scattered[1]. Although two international journals have devoted special issues to this topic [Joachims and Sebastiani 2001; Lewis and Hayes 1994], there are no systematic treatments of the subject: there are neither textbooks nor journals entirely devoted to TC yet, and [Manning and Schütze 1999, Chapter16] is the only chapter-length treatment of the subject. As a note, we should warn the reader that the term "automatic text classification" has sometimes been used in the literature to mean things quite different from the ones discussed here. 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 identification of such a set of categories (e.g. [Borko and Bernick 1963]), or (iii) the automatic identification of such a set of categories *and* the grouping of documents under them (e.g. [Merkl 1998]), a task usually called *text clustering*, or (iv) any activity of placing text items into groups, a task that has thus both TC and text clustering as particular instances [Manning and Schütze 1999].

This paper is organized as follows. In Section 2 we formally define TC and its various subcases, and in Section 3 we review its most important applications. Section 4 describes the main ideas underlying the ML approach to classification. Our discussion of *text* classification starts in Section 5 by introducing *text indexing*, i.e. the transformation of textual documents into a form that can be interpreted by a classifier-building algorithm and by the classifier eventually built by it. Section 6 tackles the inductive construction of a text classifier from a "training" set of preclassified documents. Section 7 discusses the evaluation of text classifiers. Section 8 concludes, discussing open issues and possible avenues of further research for TC.

2. TEXT CATEGORIZATION

[1]A fully searchable bibliography on TC created and maintained by this author is available at `http://liinwww.ira.uka.de/bibliography/Ai/automated.text.categorization.html`

2.1 A definition of text categorization

Text categorization is the task of assigning a Boolean value to each pair $\langle d_j, c_i \rangle \in \mathcal{D} \times \mathcal{C}$, where $\mathcal{D}$ is a domain of documents and $\mathcal{C} = \{c_1, \ldots, c_{|\mathcal{C}|}\}$ is a set of predefined *categories*. A value of T assigned to $\langle d_j, c_i \rangle$ indicates a decision to file d_j under c_i, while a value of F indicates a decision not to file d_j under c_i. More formally, the task is to approximate the unknown *target function* $\breve{\Phi} : \mathcal{D} \times \mathcal{C} \rightarrow \{T, F\}$ (that describes how documents ought to be classified) by means of a function $\Phi : \mathcal{D} \times \mathcal{C} \rightarrow \{T, F\}$ called the *classifier* (aka *rule*, or *hypothesis*, or *model*) such that $\breve{\Phi}$ and Φ "coincide as much as possible". How to precisely define and measure this coincidence (called *effectiveness*) will be discussed in Section 7.1. From now on we will assume that:

—The categories are just symbolic labels, and no additional knowledge (of a procedural or declarative nature) of their meaning is available.

—No *exogenous* knowledge (i.e. data provided for classification purposes by an external source) is available; therefore, classification must be accomplished on the basis of *endogenous* knowledge only (i.e. knowledge extracted from the documents). In particular, this means that *metadata* such as e.g. publication date, document type, publication source, etc. is not assumed to be available.

The TC methods we will discuss are thus completely general, and do not depend on the availability of special-purpose resources that might be unavailable or costly to develop. Of course, these assumptions need not be verified in operational settings, where it is legitimate to use any source of information that might be available or deemed worth developing [Díaz Esteban et al. 1998; Junker and Abecker 1997]. Relying only on endogenous knowledge means classifying a document based solely on its semantics, and given that the semantics of a document is a *subjective* notion, it follows that the membership of a document in a category (pretty much as the relevance of a document to an information need in IR [Saracevic 1975]) cannot be decided deterministically. This is exemplified by the phenomenon of *inter-indexer inconsistency* [Cleverdon 1984]: when two human experts decide 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 Clinton attending Dizzy Gillespie's funeral could be filed under `Politics`, or under `Jazz`, or under both, or even under neither, depending on the subjective judgment of the expert.

2.2 Single-label vs. multi-label text categorization

Different constraints may be enforced on the TC task, depending on the application. For instance we might need that, for a given integer k, exactly k (or $\leq k$, or $\geq k$) elements of $\mathcal{C}$ be assigned to each $d_j \in \mathcal{D}$. The case in which exactly 1 category must be assigned to each $d_j \in \mathcal{D}$ is often called the *single-label* (aka *non-overlapping categories*) case, while the case in which any number of categories from 0 to $|\mathcal{C}|$ may be assigned to the same $d_j \in \mathcal{D}$ is dubbed the *multi-label* (aka *overlapping categories*) case. A special case of single-label TC is *binary* TC, in which each $d_j \in \mathcal{D}$ must be assigned either to category c_i or to its complement $\overline{c}_i$.

From a theoretical point of view, the binary case (hence, the single-label case too) is more general than the multi-label, since an algorithm for binary classification can also be used for multi-label classification: one needs only transform the problem

of multi-label classification under $\{c_1, \ldots, c_{|\mathcal{C}|}\}$ into $|\mathcal{C}|$ independent problems of binary classification under $\{c_i, \overline{c}_i\}$, for $i = 1, \ldots, |\mathcal{C}|$. However, this requires that categories are stochastically independent of each other, i.e. that for any c', c'' the value of $\breve{\Phi}(d_j, c')$ does not depend on the value of $\breve{\Phi}(d_j, c'')$ and viceversa; this is usually assumed to be the case (applications in which this is not the case are discussed in Section 3.5). The converse is not true: an algorithm for multi-label classification cannot be used for either binary or single-label classification. In fact, given a document d_j to classify, (i) the classifier might attribute $k > 1$ categories to d_j, and it might not be obvious how to choose a "most appropriate" category from them; or (ii) the classifier might attribute to d_j no category at all, and it might not be obvious how to choose a "least inappropriate" category from $\mathcal{C}$.

In the rest of the paper, unless explicitly mentioned, we will deal with the binary case. There are various reasons for this:

—The binary case is important in itself because important TC applications, including filtering (see Section 3.3), consist of binary classification problems (e.g. deciding whether d_j is about Jazz or not). In TC, most binary classification problems feature unevenly populated categories (e.g. much fewer documents are about Jazz than are not) and unevenly characterized categories (e.g. what is about Jazz can be characterized much better than what is not).

—Solving the binary case also means solving the multi-label case, which is also representative of important TC applications, including automated indexing for Boolean systems (see Section 3.1).

—Most of the TC literature is couched in terms of the binary case.

—Most techniques for binary classification are just special cases of existing techniques for the single-label case, and are simpler to illustrate than these latter.

This ultimately means that we will view classification under $\mathcal{C} = \{c_1, \ldots, c_{|\mathcal{C}|}\}$ as consisting of $|\mathcal{C}|$ independent problems of classifying the documents in $\mathcal{D}$ under a given category c_i, for $i = 1, \ldots, |\mathcal{C}|$. A *classifier for* c_i is then a function $\Phi_i : \mathcal{D} \rightarrow \{T, F\}$ that approximates an unknown target function $\breve{\Phi}_i : \mathcal{D} \rightarrow \{T, F\}$.

2.3 Category-pivoted vs. document-pivoted text categorization

There are two different ways of using a text classifier. Given $d_j \in \mathcal{D}$, we might want to find all the $c_i \in \mathcal{C}$ under which it should be filed (*document-pivoted categorization* – DPC); alternatively, given $c_i \in \mathcal{C}$, we might want to find all the $d_j \in \mathcal{D}$ that should be filed under it (*category-pivoted categorization* – CPC). This distinction is more pragmatic than conceptual, but is important since the sets $\mathcal{C}$ and $\mathcal{D}$ might not be available in their entirety right from the start. It is also relevant to the choice of the classifier-building method, as some of these methods (see e.g. Section 6.9) allow the construction of classifiers with a definite slant towards one or the other style.

DPC is thus suitable when documents become available at different moments in time, e.g. in filtering e-mail. CPC is instead suitable when (i) a new category $c_{|\mathcal{C}|+1}$ may be added to an existing set $\mathcal{C} = \{c_1, \ldots, c_{|\mathcal{C}|}\}$ after a number of documents have already been classified under $\mathcal{C}$, *and* (ii) these documents need to be reconsidered for classification under $c_{|\mathcal{C}|+1}$ (e.g. [Larkey 1999]). DPC is used more often than CPC, as the former situation is more common than the latter.

Although some specific techniques apply to one style and not to the other (e.g. the proportional thresholding method discussed in Section 6.1 applies only to CPC), this is more the exception than the rule: most of the techniques we will discuss allow the construction of classifiers capable of working in either mode.

2.4 "Hard" categorization vs. ranking categorization

While a complete automation of the TC task requires a T or F decision for each pair $\langle d_j, c_i \rangle$, a partial automation of this process might have different requirements.

For instance, given $d_j \in \mathcal{D}$ a system might simply *rank* the categories in $\mathcal{C} = \{c_1, \ldots, c_{|\mathcal{C}|}\}$ according to their estimated appropriateness to d_j, without taking any "hard" decision on any of them. Such a ranked list would be of great help to a human expert in charge of taking the final categorization decision, since she could thus restrict the choice to the category (or categories) at the top of the list, rather than having to examine the entire set. Alternatively, given $c_i \in \mathcal{C}$ a system might simply rank the documents in $\mathcal{D}$ according to their estimated appropriateness to c_i; symmetrically, for classification under c_i a human expert would just examine the top-ranked documents instead than the entire document set. These two modalities are sometimes called *category-ranking TC* and *document-ranking TC* [Yang 1999], respectively, and are the obvious counterparts of DPC and CPC.

Semi-automated, "interactive" classification systems [Larkey and Croft 1996] are useful especially in critical applications in which the effectiveness of a fully automated system may be expected to be significantly lower than that of a human expert. This may be the case when the quality of the training data (see Section 4) is low, or when the training documents cannot be trusted to be a representative sample of the unseen documents that are to come, so that the results of a completely automatic classifier could not be trusted completely.

In the rest of the paper, unless explicitly mentioned, we will deal with "hard" classification; however, many of the algorithms we will discuss naturally lend themselves to ranking TC too (more details on this in Section 6.1).

3. APPLICATIONS OF TEXT CATEGORIZATION

TC goes back to Maron's [1961] seminal work on probabilistic text classification. Since then, it has been used for a number of different applications, of which we here briefly review the most important ones. Note that the borders between the different classes of applications listed here are fuzzy and somehow artificial, and some of these may be considered special cases of others. Other applications we do not explicitly discuss are speech categorization by means of a combination of speech recognition and TC [Myers et al. 2000; Schapire and Singer 2000], multimedia document categorization through the analysis of textual captions [Sable and Hatzivassiloglou 2000], author identification for literary texts of unknown or disputed authorship [Forsyth 1999], language identification for texts of unknown language [Cavnar and Trenkle 1994], automated identification of text genre [Kessler et al. 1997], and automated essay grading [Larkey 1998].

3.1 Automatic indexing for Boolean information retrieval systems

The application that has spawned most of the early research in the field [Borko and Bernick 1963; Field 1975; Gray and Harley 1971; Heaps 1973; Maron 1961], is that

of automatic document indexing for IR systems relying on a controlled dictionary, the most prominent example of which is that of Boolean systems. In these latter each document is assigned one or more keywords or keyphrases describing its content, where these keywords and keyphrases belong to a finite set called *controlled dictionary*, often consisting of a thematic hierarchical thesaurus (e.g. the NASA thesaurus for the aerospace discipline, or the MESH thesaurus for medicine). Usually, this assignment is done by trained human indexers, and is thus a costly activity.

If the entries in the controlled vocabulary are viewed as categories, text indexing is an instance of TC, and may thus be addressed by the automatic techniques described in this paper. Recalling Section 2.2, note that this application may typically require that $k_1 \leq x \leq k_2$ keywords are assigned to each document, for given k_1, k_2. Document-pivoted TC is probably the best option, so that new documents may be classified as they become available. Various text classifiers explicitly conceived for document indexing have been described in the literature; see e.g. [Fuhr and Knorz 1984; Robertson and Harding 1984; Tzeras and Hartmann 1993].

Automatic indexing with controlled dictionaries is closely related to *automated metadata generation*. In digital libraries one is usually interested in tagging documents by metadata that describe them under a variety of aspects (e.g. creation date, document type or format, availability, etc.). 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 TC techniques.

3.2 Document organization

Indexing with a controlled vocabulary is an instance of the general problem of document base organization. In general, many other issues pertaining to document organization and filing, be it for purposes of personal organization or structuring of a corporate document base, may be addressed by TC techniques. For instance, at the offices of a newspaper incoming "classified" ads must be, prior to publication, categorized under categories such as `Personals`, `Cars for Sale`, `Real Estate`, etc. Newspapers dealing with a high volume of classified ads would benefit from an automatic system that chooses the most suitable category for a given ad. Other possible applications are the organization of patents into categories for making their search easier [Larkey 1999], 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.

3.3 Text filtering

Text filtering is the activity of classifying 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 is a newsfeed, where the producer is a news agency and the consumer is a newspaper [Hayes et al. 1990]. In this case the filtering system should block the delivery of the documents the consumer is likely not 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 TC, i.e. the classification of incoming documents in two disjoint categories, the relevant and the

irrelevant. Additionally, a filtering system may also further classify the documents deemed relevant to the consumer into thematic categories; in the example above, all articles about sports should be further classified according e.g. to which sport they deal with, so as to allow journalists specialized in individual sports to access only documents of prospective interest for them. Similarly, an e-mail filter might be trained to discard "junk" mail [Androutsopoulos et al. 2000; Drucker et al. 1999] and further classify non-junk mail into topical categories of interest to the user.

A filtering system may be installed at the producer end, in which case it must route the documents to the interested consumers only, or at the consumer end, in which case it must block the delivery of documents deemed uninteresting to the consumer. In the former case the system builds and updates a "profile" for each consumer [Liddy et al. 1994], while 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 updated by the system by using feedback information provided (either implicitly or explicitly) by the user on the relevance or non-relevance of the delivered messages. In the TREC community [Lewis 1995c] 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. Batch filtering thus coincides with single-label TC under $|\mathcal{C}| = 2$ categories; since this latter is a completely general TC task some authors [Hull 1994; Hull et al. 1996; Schapire et al. 1998; Schütze et al. 1995], somewhat confusingly, use the term "filtering" in place of the more appropriate term "categorization".

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

Information filtering by ML techniques is widely discussed in the literature: see e.g. [Amati and Crestani 1999; Iyer et al. 2000; Kim et al. 2000; Tauritz et al. 2000; Yu and Lam 1998].

3.4 Word sense disambiguation

Word sense disambiguation (WSD) is the activity of finding, given the occurrence in a text of an ambiguous (i.e. polysemous or homonymous) word, the sense of this particular word occurrence. For instance, `bank` may have (at least) two different senses in English, 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 which of the above senses the occurrence of `bank` in `Last week I borrowed some money from the bank` has. WSD is very important for many applications, including natural language processing, and indexing documents by word senses rather than by words for IR purposes. WSD may be seen as a TC task (see e.g [Gale et al. 1993; Escudero et al. 2000]) once we view word occurrence contexts as documents and word senses as categories. Quite obviously, this is a single-label TC

case, and one in which document-pivoted TC is usually the right choice.

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 examples, 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 [Roth 1998] for an introduction.

3.5 Hierarchical categorization of Web pages

TC has recently aroused a lot of interest also for its possible application to automatically classifying Web pages, or sites, under the hierarchical catalogues hosted by popular Internet portals. When Web documents are catalogued in this way, rather than issuing a query to a general-purpose Web search engine a searcher may find it easier to first navigate in the hierarchy of categories and then restrict her search to a particular category of interest.

Classifying Web pages automatically has obvious advantages, since the manual categorization of a large enough subset of the Web is infeasible. Unlike in the previous applications, it is typically the case that each category must be populated by a set of $k_1 \leq x \leq k_2$ documents. CPC should be chosen so as to allow new categories to be added and obsolete ones to be deleted.

With respect to previously discussed TC applications, automatic Web page categorization has two essential peculiarities:

(1) *The hypertextual nature of the documents*: links are a rich source of information, as they may be understood as stating the relevance of the linked page to the linking page. Techniques exploiting this intuition in a TC context have been presented in [Attardi et al. 1998; Chakrabarti et al. 1998b; Fürnkranz 1999; Gövert et al. 1999; Oh et al. 2000] and experimentally compared in [Yang et al. 2001].

(2) *The hierarchical structure of the category set*: this may be used e.g. by decomposing the classification problem into a number of smaller classification problems, each corresponding to a branching decision at an internal node. Techniques exploiting this intuition in a TC context have been presented in [Dumais and Chen 2000; Chakrabarti et al. 1998a; Koller and Sahami 1997; McCallum et al. 1998; Ruiz and Srinivasan 1999; Weigend et al. 1999].

4. THE MACHINE LEARNING APPROACH TO TEXT CATEGORIZATION

In the '80s the most popular approach (at least in operational settings) for the creation of automatic document classifiers consisted in manually building, by means of *knowledge engineering* (KE) techniques, an expert system capable of taking TC decisions. Such an expert system would typically consist of a set of manually defined logical rules, one per category, of type

$$\textbf{if } \langle \textit{DNF formula} \rangle \textbf{ then } \langle \textit{category} \rangle$$

A DNF ("disjunctive normal form") formula is a disjunction of conjunctive clauses; the document is classified under $\langle \textit{category} \rangle$ iff it satisfies the formula, i.e. iff it satisfies at least one of the clauses. The most famous example of this approach is

if	((*wheat* & *farm*)	**or**
	(*wheat* & *commodity*)	**or**
	(*bushels* & *export*)	**or**
	(*wheat* & *tonnes*)	**or**
	(*wheat* & *winter* & ¬ *soft*))	**then** WHEAT **else** ¬ WHEAT

Fig. 1. Rule-based classifier for the WHEAT category; keywords are indicated in *italic*, categories are indicated in SMALL CAPS (from [Apté et al. 1994]).

the CONSTRUE system [Hayes et al. 1990], built by Carnegie Group for the Reuters news agency. A sample rule of the type used in CONSTRUE is illustrated in Figure 1.

The drawback of this approach is the *knowledge acquisition bottleneck* well-known from the expert systems literature. That is, the rules must be manually defined by a knowledge engineer with the aid of a domain expert (in this case, an expert in the membership of documents in the chosen set of categories): if the set of categories is updated, then these two professionals must intervene again, and if the classifier is ported to a completely different domain (i.e. set of categories) a different domain expert needs to intervene and the work has to be repeated from scratch.

On the other hand, it was originally suggested that this approach can give very good effectiveness results: Hayes et al. [1990] reported a .90 "breakeven" result (see Section 7) on a subset of the Reuters test collection, a figure that outperforms even the best classifiers built in the late '90s by state-of-the-art ML techniques. However, no other classifier has been tested on the same dataset as CONSTRUE, and it is not clear whether this was a randomly chosen or a favourable subset of the entire Reuters collection. As argued in [Yang 1999], the results above do not allow us to state that these effectiveness results may be obtained in general.

Since the early '90s, the ML approach to TC has gained popularity and has eventually become the dominant one, at least in the research community (see [Mitchell 1996] for a comprehensive introduction to ML). In this approach a general inductive process (also called the *learner*) automatically builds a classifier for a category c_i by observing the characteristics of a set of documents manually classified under c_i or $\overline{c}_i$ by a domain expert; from these characteristics, the inductive process gleans the characteristics that a new unseen document should have in order to be classified under c_i. In ML terminology, the classification problem is an activity of *supervised* learning, since the learning process is "supervised" by the knowledge of the categories and of the training instances that belong to them[2].

The advantages of the ML approach over the KE approach are evident. The engineering effort goes towards the construction not of a classifier, but of an automatic builder of classifiers (the *learner*). This means that if a learner is (as it often is) available off-the-shelf, all that is needed is the inductive, *automatic* construction of a classifier from a set of manually classified documents. The same happens if a classifier already exists and the original set of categories is updated, or if the classifier is ported to a completely different domain.

[2]Within the area of content-based document management tasks, an example of an *unsupervised* learning activity is *document clustering* (see Section 1).

In the ML approach the preclassified documents are then the key resource. In the most favourable case they are already available; this typicaly happens for organizations which have previously carried out the same categorization activity manually and decide to automate the process. The less favourable case is when no manually classified documents are available; this typicaly happens for organizations which start a categorization activity and opt for an automated modality straightaway. The ML approach is more convenient than the KE approach also in this latter case. In fact, it is easier to manually classify a set of documents than to build and tune a set of rules, since it is easier to characterize a concept extensionally (i.e. to select instances of it) than intensionally (i.e. to describe the concept in words, or to describe a procedure for recognizing its instances).

Classifiers built by means of ML techniques nowadays achieve impressive levels of effectiveness (see Section 7), making automatic classification a *qualitatively* (and not only economically) viable alternative to manual classification.

4.1 Training set, test set, and validation set

The ML approach relies on the availability of an *initial corpus* $\Omega = \{d_1, \ldots, d_{|\Omega|}\} \subset \mathcal{D}$ of documents preclassified under $\mathcal{C} = \{c_1, \ldots, c_{|\mathcal{C}|}\}$. That is, the values of the total function $\breve{\Phi} : \mathcal{D} \times \mathcal{C} \rightarrow \{T, F\}$ are known for every pair $\langle d_j, c_i \rangle \in \Omega \times \mathcal{C}$. A document d_j is a *positive example* of c_i if $\breve{\Phi}(d_j, c_i) = T$, a *negative example* of c_i if $\breve{\Phi}(d_j, c_i) = F$.

In research settings (and in most operational settings too), once a classifier Φ has been built it is desirable to evaluate its effectiveness. In this case, prior to classifier construction the initial corpus is split in two sets, not necessarily of equal size:

—a *training(-and-validation) set* $TV = \{d_1, \ldots, d_{|TV|}\}$. The classifier Φ for categories $\mathcal{C} = \{c_1, \ldots, c_{|\mathcal{C}|}\}$ is inductively built by observing the characteristics of these documents;

—a *test set* $Te = \{d_{|TV|+1}, \ldots, d_{|\Omega|}\}$, used for testing the effectiveness of the classifiers. Each $d_j \in Te$ is fed to the classifier, and the classifier decisions $\Phi(d_j, c_i)$ are compared with the expert decisions $\breve{\Phi}(d_j, c_i)$. A measure of classification effectiveness is based on how often the $\Phi(d_j, c_i)$ values match the $\breve{\Phi}(d_j, c_i)$ values.

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 likely be unrealistically good, and the evaluation would thus have no scientific character [Mitchell 1996, page 129]. In an operational setting, after evaluation has been performed one would typically re-train the classifier on the entire initial corpus, in order to boost effectiveness. In this case the results of the previous evaluation would be a pessimistic estimate of the real performance, since the final classifier has been trained on more data than the classifier evaluated.

This is called the *train-and-test* approach. An alternative is the *k-fold cross-validation* approach (see e.g. [Mitchell 1996, page 146]), in which k different classifiers $\Phi_1, \ldots, \Phi_k$ are built 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 $\langle TV_i = \Omega - Te_i, Te_i \rangle$. The final effectiveness figure is obtained by individually computing the effectiveness of $\Phi_1, \ldots, \Phi_k$, and then averaging the individual re-

sults in some way.

In both approaches it is often the case that the internal parameters of the classifiers must be tuned, by testing which values of the parameters yield the best effectiveness. In order to make this optimization possible, in the train-and-test approach the set $\{d_1, \ldots, d_{|TV|}\}$ is further split into a *training set* $Tr = \{d_1, \ldots, d_{|Tr|}\}$, from which the classifier is built, and a *validation set* $Va = \{d_{|Tr|+1}, \ldots, d_{|TV|}\}$ (sometimes called a *hold-out set*), on which the repeated tests of the classifier aimed at parameter optimization are performed; the obvious variant may be used in the k-fold cross-validation case. Note that, for the same reason why we do not test a classifier on the documents it has been trained on, we do not test it on the documents it has been optimized on: test set and validation set must be kept separate[3].

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

$$g_\Omega(c_i) \;=\; \frac{|\{d_j \in \Omega \mid \breve{\Phi}(d_j, c_i) = T\}|}{|\Omega|}$$

The *training set generality* $g_{Tr}(c_i)$, *validation set generality* $g_{Va}(c_i)$, and *test set generality* $g_{Te}(c_i)$ of c_i may be defined in the obvious way.

4.2 Information retrieval techniques and text categorization

Text categorization heavily relies on the basic machinery of IR. The reason is that TC is a content-based document management task, and as such it shares many characteristics with other IR tasks such as text search.

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 classified during the operational phase;

(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 (2), although in a few cases non-standard approaches to (1) and (3) are also used. Indexing, induction and evaluation are the themes of Sections 5, 6 and 7, respectively.

5. DOCUMENT INDEXING AND DIMENSIONALITY REDUCTION

5.1 Document indexing

Texts cannot be directly interpreted by a classifier or by a classifier-building algorithm. Because of this, an *indexing* procedure that maps a text d_j into a compact representation of its content needs to be uniformly applied to training, validation and test documents. The choice of a representation for text depends on what one regards as the meaningful units of text (the problem of *lexical semantics*) and the meaningful natural language rules for the combination of these units (the problem

[3]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 the operational phase.

of *compositional semantics*). Similarly to what happens in IR, in TC this latter problem is usually disregarded[4], and a text d_j is usually represented as a vector of term *weights* $\vec{d_j} = \langle w_{1j}, \ldots, w_{|\mathcal{T}|j} \rangle$, where $\mathcal{T}$ is the set of *terms* (sometimes called *features*) that occur at least once in at least one document of Tr, and $0 \leq w_{kj} \leq 1$ represents, loosely speaking, how much term t_k contributes to the semantics of document d_j. Differences among approaches are accounted for by

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

(2) different ways to compute term weights.

A typical choice for (1) is to identify terms with words. This is often called either the *set of words* or the *bag of words* approach to document representation, depending on whether weights are binary or not.

In a number of experiments [Apté et al. 1994; Dumais et al. 1998; Lewis 1992a] it has been found that representations more sophisticated than this do not yield significantly better effectiveness, thereby confirming similar results from IR [Salton and Buckley 1988]. In particular, some authors have used *phrases*, rather than individual words, as indexing terms [Fuhr et al. 1991; Schütze et al. 1995; Tzeras and Hartmann 1993], but the experimental results found to date have not been uniformly 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. [Lewis 1992a]);

—*statistically*, i.e. the phrase is not grammatically such, but is composed of a set/sequence of words whose patterns of contiguous occurrence in the collection are statistically significant (see e.g. [Caropreso et al. 2001]).

Lewis [1992a] 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 word-only indexing languages: a phrase-only indexing language has "more terms, more synonymous or nearly synonymous terms, lower consistency of assignment (since synonymous terms are not assigned to the same documents), and lower document frequency for terms" [Lewis 1992a, page 40]. Although his remarks are about syntactically motivated phrases, they also apply to statistically motivated ones, although perhaps to a smaller degree. A combination of the two approaches is probably the best way to go: Tzeras and Hartmann [1993] obtained significant improvements by using noun phrases obtained through a combination of syntactic and statistical criteria, where a "crude" syntactic method was complemented by a statistical filter (only those syntactic phrases that occurred at least three times in the positive examples of a category c_i were retained). It is likely that the final word on the usefulness of phrase indexing in TC has still to be told, and investigations in this direction are still being actively pursued [Caropreso et al. 2001; Mladenić and Grobelnik 1998].

As for issue (2), weights usually range between 0 and 1 (an exception is [Lewis et al. 1996]), and for ease of exposition we will assume they always do. As a special case, binary weights may be used (1 denoting presence and 0 absence of the term

[4]An exception to this is represented by learning approaches based on *Hidden Markov Models* [Denoyer et al. 2001; Frasconi et al. 2001].

in the document); whether binary or non-binary weights are used depends on the classifier learning algorithm used. In the 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 $tfidf$ function is used (see e.g. [Salton and Buckley 1988]), defined as

$$tfidf(t_k, d_j) = \#(t_k, d_j) \cdot \log \frac{|Tr|}{\#_{Tr}(t_k)} \tag{1}$$

where $\#(t_k, d_j)$ denotes the number of times t_k occurs in d_j, and $\#_{Tr}(t_k)$ denotes the *document frequency* of term t_k, i.e. the number of documents in Tr in which t_k occurs. This function embodies the intuitions that (i) the more often a term occurs in a document, the more it is representative of its content, and (ii) the more documents a term occurs in, the less discriminating it is[5]. 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 occur in the document and the syntactic role they play. In other words, the semantics of a document is reduced to the collective lexical semantics of the terms that occur in it, thereby disregarding the issue of compositional semantics (an exception are the representation techniques used for FOIL [Cohen 1995a] and SLEEPING EXPERTS [Cohen and Singer 1999]).

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

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

Although normalized $tfidf$ is the most popular one, other indexing functions have also been used, including probabilistic techniques [Gövert et al. 1999] or techniques for indexing structured documents [Larkey and Croft 1996]. Functions different from $tfidf$ are especially needed when Tr is not available in its entirety from the start and $\#_{Tr}(t_k)$ cannot thus be computed, as e.g. in adaptive filtering; in this case approximations 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 almost always performed (exceptions include [Lewis et al. 1996; Nigam et al. 2000; Riloff 1995])[6]. Concerning *stemming* (i.e. grouping words that share the same morphological root), its suitability to TC is controversial. Although, similarly to unsupervised term clustering (see Section 5.5.1) 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,

[5]There exist many variants of $tfidf$, that differ from each other in terms of logarithms, normalization or other correction factors. Formula 1 is just one of the possible instances of this class; see [Salton and Buckley 1988; Singhal et al. 1996] for variations on this theme.

[6]One application of TC in which it would be inappropriate to remove function words is author identification for documents of disputed paternity. In fact, as noted in [Manning and Schütze 1999, page 589], "it is often the 'little' words that give an author away (for example, the relative frequencies of words like because or though)".

as it reduces both the dimensionality of the term space (see Section 5.3) and the stochastic dependence between terms (see Section 6.2).

Depending on the application, either the full text of the document or selected parts of it are indexed. While the former option is the rule, exceptions exist. For instance, in a patent categorization application Larkey [1999] indexes only the title, the abstract, the first 20 lines of the 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, one can pay extra importance to the words it contains [Apté et al. 1994; Cohen and Singer 1999; Weiss et al. 1999]. When documents are flat, identifying the most relevant part of a document is instead a non-obvious task.

5.2 The Darmstadt Indexing Approach

The AIR/X system [Fuhr et al. 1991] occupies a special place in the literature on indexing for TC. This system is the final result of the AIR project, one of the most important efforts in the history of TC: spanning a duration of more than ten years [Knorz 1982; Tzeras and Hartmann 1993], it has produced a system operatively employed since 1985 in the classification of corpora of scientific literature of $O(10^5)$ documents and $O(10^4)$ categories, and has had important theoretical spin-offs in the field of probabilistic indexing [Fuhr 1989; Fuhr and Buckley 1991][7].

The approach to indexing taken in AIR/X is known as the *Darmstadt Indexing Approach* (DIA) [Fuhr 1985]. Here, "indexing" is used 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 free terms [Fuhr and Buckley 1991]). The idea that underlies the DIA is the use of a much wider set of "features" than described in Section 5.1. All other approaches mentioned in this paper view *terms* as the dimensions of the learning space, where terms may be single words, stems, phrases, or (see Sections 5.5.1 and 5.5.2) combinations of any of these. In contrast, the DIA considers *properties* (of terms, documents, categories, or pairwise relationships among these) as basic dimensions of the learning space. Examples of these are

—*properties of a term* t_k: e.g. the *idf* of t_k;

—*properties of the relationship between a term* t_k *and a document* d_j: e.g. the tf of t_k in d_j; or the location (e.g. in the title, or in the abstract) of t_k within d_j;

—*properties of a document* d_j: e.g. the length of d_j;

—*properties of a category* c_i: e.g. the training set generality of c_i.

For each possible document-category pair, the values of these features are collected in a so-called *relevance description* vector $\vec{rd}(d_j, c_i)$. The size of this vector is determined by the number of properties considered, and is thus independent of specific terms, categories or documents (for multivalued features, appropriate aggregation

[7]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.

functions are applied in order to yield a single value to be included in $\vec{rd}(d_j, c_i)$); in this way an abstraction from specific terms, categories or documents is achieved.

The main advantage of this approach is the possibility to consider additional features that can hardly be accounted for in the usual term-based approaches, e.g. the location of a term within a document, or the certainty with which a phrase was identified in a document. The term-category relationship is described by estimates, derived from the training set, of the probability $P(c_i|t_k)$ that a document belongs to category c_i, given that it contains term t_k (the *DIA association factor*)[8]. Relevance description vectors $\vec{rd}(d_j, c_i)$ are then the final representations that are used for the classification of document d_j under category c_i.

The essential ideas of the DIA – transforming the classification space by means of abstraction and using a more detailed text representation than the standard bag-of-words approach – have not been taken up by other researchers so far. For new TC applications dealing with structured documents or categorization of Web pages, these ideas may become of increasing importance.

5.3 Dimensionality reduction

Unlike in text retrieval, in TC the high dimensionality of the term space (i.e. the large value of $|\mathcal{T}|$) may be problematic. In fact, while typical algorithms used in text retrieval (such as cosine matching) can scale to high values of $|\mathcal{T}|$, the same does not hold of many sophisticated learning algorithms used for classifier induction (e.g. the LLSF algorithm of [Yang and Chute 1994]). Because of this, before classifier induction one often applies a pass of *dimensionality reduction* (DR), whose effect is to reduce the size of the vector space from $|\mathcal{T}|$ to $|\mathcal{T}'| \ll |\mathcal{T}|$; the set $\mathcal{T}'$ is called the *reduced term set*.

DR is also beneficial since it tends to reduce *overfitting*, i.e. the phenomenon by which a classifier is tuned also to the *contingent* characteristics of the training data rather than just the *constitutive* characteristics of the categories. Classifiers which overfit the training data are good at re-classifying the data they have been trained on, but much worse at classifying previously unseen data. Experiments have shown that in order to avoid overfitting a number of training examples roughly proportional to the number of terms used is needed; Fuhr and Buckley [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. However, in removing terms the risk is to remove potentially useful information on the meaning of the documents. It is then clear that, in order to obtain optimal (cost-)effectiveness, the reduction process must be performed with care. Various DR methods have been proposed, either from the information theory or from the linear algebra literature, and their relative merits have been tested by experimentally evaluating the variation in effectiveness that a given classifier undergoes after application of the function to the term space.

There are two distinct ways of viewing DR, depending on whether the task is performed locally (i.e. for each individual category) or globally:

[8] Association factors are called *adhesion coefficients* in many early papers on TC; see e.g. [Field 1975; Robertson and Harding 1984].

—*local DR*: for each category c_i, a set $\mathcal{T}_i'$ of terms, with $|\mathcal{T}_i'| \ll |\mathcal{T}|$, is chosen for classification under c_i (see e.g. [Apté et al. 1994; Lewis and Ringuette 1994; Li and Jain 1998; Ng et al. 1997; Sable and Hatzivassiloglou 2000; Schütze et al. 1995; Wiener et al. 1995]). This means that different subsets of $\vec{d}_j$ are used when working with the different categories. Typical values are $10 \leq |\mathcal{T}_i'| \leq 50$.

—*global DR*: a set $\mathcal{T}'$ of terms, with $|\mathcal{T}'| \ll |\mathcal{T}|$, is chosen for the classification under all categories $\mathcal{C} = \{c_1, \ldots, c_{|\mathcal{C}|}\}$ (see e.g. [Caropreso et al. 2001; Mladenić 1998; Yang 1999; Yang and Pedersen 1997]).

This distinction usually does not impact on the choice of DR technique, since most such techniques can be used (and have been used) for local and global DR alike (*supervised* DR techniques – see Section 5.5.1 – are exceptions to this rule). In the rest of this section we will assume that the global approach is used, although everything we will say also applies to the local approach.

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

—*DR by term selection*: $\mathcal{T}'$ is a subset of $\mathcal{T}$;

—*DR by term extraction*: the terms in $\mathcal{T}'$ are not of the same type of the terms in $\mathcal{T}$ (e.g. if the terms in $\mathcal{T}$ are words, the terms in $\mathcal{T}'$ may not be words at all), but are obtained by combinations or transformations of the original ones.

Unlike in the previous distinction, these two ways of doing DR are tackled by very different techniques; we will address them separately in the next two sections.

5.4 Dimensionality reduction by term selection

Given a predetermined integer r, techniques for term selection (also called *term space reduction* – TSR) attempt to select, from the original set $\mathcal{T}$, the set $\mathcal{T}'$ of terms (with $|\mathcal{T}'| \ll |\mathcal{T}|$) that, when used for document indexing, yields the highest effectiveness. Yang and Pedersen [1997] have shown that TSR may even result in a moderate ($\leq 5\%$) increase in effectiveness, depending on the classifier, on the *aggressivity* $\frac{|\mathcal{T}|}{|\mathcal{T}'|}$ of the reduction, and on the TSR technique used.

Moulinier et al. [1996] have used a so-called *wrapper* approach, i.e. one in which $\mathcal{T}'$ is identified by means of the same learning method which will be used for building 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 built and then tested on a validation set. The term set that results in the best effectiveness is chosen. This approach has the advantage of being tuned to the learning algorithm being used; moreover, if local DR is performed, 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, the sheer size of the space of different term sets makes its cost prohibitive for standard TC applications.

A computationally easier alternative is the *filtering* approach [John et al. 1994], i.e. keeping the $|\mathcal{T}'| \ll |\mathcal{T}|$ terms that receive the highest score according to a function that measures the "importance" of the term for the TC task. We will explore this solution in the rest of this section.

5.4.1 *Document frequency.* A simple and effective global TSR function is the *document frequency* $\#_{Tr}(t_k)$ of a term t_k, i.e. only the terms that occur in the highest number of documents are retained. In a series of experiments Yang and Pedersen [1997] have shown that with $\#_{Tr}(t_k)$ it is possible to reduce the dimensionality by a factor of 10 with no loss in effectiveness (a reduction by a factor of 100 bringing about just a small loss).

This seems to indicate that the terms occurring most frequently in the collection are the most valuable for TC. As such, this would seem to contradict a well-known "law" of IR, according to which the terms with low-to-medium document frequency are the most informative ones [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 large majority of the words occurring in a corpus have a *very* low document frequency; this means that by reducing the term set 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 words need to be removed in advance, lest only topic-neutral words are retained [Mladenić 1998].

Finally, note that a slightly more empirical form of TSR by document frequency is adopted by many authors, who remove all terms occurring in at most x training documents (popular values for x range from 1 to 3), either as the only form of DR [Maron 1961; Ittner et al. 1995] or before applying another more sophisticated form [Dumais et al. 1998; Li and Jain 1998]. A variant of this policy is removing all terms that occur at most x times in the training set (e.g. [Dagan et al. 1997; Joachims 1997]), with popular values for x ranging from 1 (e.g. [Baker and McCallum 1998]) to 5 (e.g. [Apté et al. 1994; Cohen 1995a]).

5.4.2 *Other information-theoretic term selection functions.* Other more sophisticated information-theoretic functions have been used in the literature, among which the *DIA association factor* [Fuhr et al. 1991], *chi-square* [Caropreso et al. 2001; Galavotti et al. 2000; Schütze et al. 1995; Sebastiani et al. 2000; Yang and Pedersen 1997; Yang and Liu 1999], *NGL coefficient* [Ng et al. 1997; Ruiz and Srinivasan 1999], *information gain* [Caropreso et al. 2001; Larkey 1998; Lewis 1992a; Lewis and Ringuette 1994; Mladenić 1998; 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* [Caropreso et al. 2001; Mladenić 1998; Ruiz and Srinivasan 1999], *relevancy score* [Wiener et al. 1995], and *GSS coefficient* [Galavotti et al. 2000]. The mathematical definitions of these measures are summarized for convenience in Table 1[9]. Here, probabilities are interpreted on an event space of documents (e.g. $P(\overline{t}_k, c_i)$ denotes 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

[9]For better uniformity Table 1 views all the TSR functions in terms of subjective probability. In some cases such as $\#(t_k, c_i)$ and $\chi^2(t_k, c_i)$ this is slightly artificial, since these two functions are not usually viewed in probabilistic terms. The formulae refer to the "local" (i.e. category-specific) forms of the functions, which again is slightly artificial in some cases (e.g. $\#(t_k, c_i)$). Note that the NGL and GSS coefficients are here named after their authors, since they had originally been given names that might generate some confusion if used here.

Function	Denoted by	Mathematical form
Document frequency	$\#(t_k, c_i)$	$P(t_k\|c_i)$
DIA association factor	$z(t_k, c_i)$	$P(c_i\|t_k)$
Information gain	$IG(t_k, c_i)$	$\sum_{c\in\{c_i,\overline{c}_i\}}\sum_{t\in\{t_k,\overline{t}_k\}} P(t,c)\cdot\log\frac{P(t,c)}{P(t)\cdot P(c)}$
Mutual information	$MI(t_k, c_i)$	$\log\frac{P(t_k,c_i)}{P(t_k)\cdot P(c_i)}$
Chi-square	$\chi^2(t_k, c_i)$	$\frac{\lvert Tr\rvert\cdot[P(t_k,c_i)\cdot P(\overline{t}_k,\overline{c}_i)-P(t_k,\overline{c}_i)\cdot P(\overline{t}_k,c_i)]^2}{P(t_k)\cdot P(\overline{t}_k)\cdot P(c_i)\cdot P(\overline{c}_i)}$
NGL coefficient	$NGL(t_k, c_i)$	$\frac{\sqrt{\lvert Tr\rvert}\cdot[P(t_k,c_i)\cdot P(\overline{t}_k,\overline{c}_i)-P(t_k,\overline{c}_i)\cdot P(\overline{t}_k,c_i)]}{\sqrt{P(t_k)\cdot P(\overline{t}_k)\cdot P(c_i)\cdot P(\overline{c}_i)}}$
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	$OR(t_k, c_i)$	$\frac{P(t_k\|c_i)\cdot(1-P(t_k\|\overline{c}_i))}{(1-P(t_k\|c_i))\cdot P(t_k\|\overline{c}_i)}$
GSS coefficient	$GSS(t_k, c_i)$	$P(t_k,c_i)\cdot P(\overline{t}_k,\overline{c}_i)-P(t_k,\overline{c}_i)\cdot P(\overline{t}_k,c_i)$

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

counting occurrences in the training set. All functions are specified "locally" to a specific category c_i; in order to assess the value of a term t_k in a "global", category-independent sense, either the sum $f_{sum}(t_k) = \sum_{i=1}^{|\mathcal{C}|} f(t_k, c_i)$, or the weighted average $f_{wavg}(t_k) = \sum_{i=1}^{|\mathcal{C}|} P(c_i) f(t_k, c_i)$, or the maximum $f_{max}(t_k) = \max_{i=1}^{|\mathcal{C}|} f(t_k, c_i)$ of their category-specific values $f(t_k, c_i)$ are usually computed.

These functions try to capture the intuition that the best terms for c_i are the ones distributed most differently in the sets of positive and negative examples of c_i. However, interpretations of this principle vary across different functions. For instance, in the experimental sciences χ^2 is used to measure how the results of an observation differ (i.e. are independent) from the results expected according to an initial hypothesis (lower values indicate lower dependence). In DR we measure how independent t_k and c_i are. The terms t_k with the lowest value for $\chi^2(t_k, c_i)$ are thus the most independent from c_i; since we are interested in the terms which are not, we select the terms for which $\chi^2(t_k, c_i)$ is highest.

While each TSR function has its own rationale, the ultimate word on its value is the effectiveness it brings about. Various experimental comparisons of TSR functions have thus been carried out [Caropreso et al. 2001; Galavotti et al. 2000; Mladenić 1998; Yang and Pedersen 1997]. In these experiments most functions listed in Table 1 (with the possible exception of MI) have improved on the results of document frequency. For instance, Yang and Pedersen [1997] have shown that, with various classifiers and various initial corpora, sophisticated techniques such

as $IG_{sum}(t_k, c_i)$ or $\chi^2_{max}(t_k, c_i)$ can reduce the dimensionality of the term space by a factor of 100 with no loss (or even with a small increase) of effectiveness. Collectively, the experiments reported in the above-mentioned papers seem to indicate that $\{OR_{sum}, NGL_{sum}, GSS_{max}\} > \{\chi^2_{max}, IG_{sum}\} > \{\#_{wavg}, \chi^2_{wavg}\} \gg \{MI_{max}, MI_{wavg}\}$, where ">" means "performs better than". However, it should be noted that these results are just indicative, and that more general statements on the relative merits of these functions could be made only as a result of comparative experiments performed in thoroughly controlled conditions and on a variety of different situations (e.g. different classifiers, different initial corpora, ...).

5.5 Dimensionality reduction by term extraction

Given a predetermined $|\mathcal{T}'| \ll |\mathcal{T}|$, *term extraction* attempts to generate, from the original set $\mathcal{T}$, a set $\mathcal{T}'$ of "synthetic" terms that maximize effectiveness. The rationale for using synthetic (rather than naturally occurring) terms is that, due to the pervasive problems of polysemy, homonymy and synonymy, the original terms may not be optimal dimensions for document content representation. Methods for term extraction try to solve these problems by creating artificial terms that do not suffer from them. Any term extraction method consists in (i) a method for extracting the new terms from the old ones, and (ii) a method for converting the original document representations into new representations based on the newly synthesized dimensions. Two term extraction methods have been experimented in TC, namely term clustering and latent semantic indexing.

5.5.1 *Term clustering.* *Term clustering* tries to group words with a high degree of pairwise semantic relatedness, so that the groups (or their centroids, or a representative of them) may be used instead of the terms as dimensions of the vector space. Term clustering is different from term selection, since the former tends to address terms *synonymous* (or near-synonymous) with other terms, while the latter targets *non-informative* terms[10].

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

Li and Jain [1998] 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 6.11) hardly allows any definitive conclusion to be reached.

Both [Lewis 1992a; Li and Jain 1998] are examples of *unsupervised* clustering, since clustering is not affected by the category labels attached to the documents. Baker and McCallum [1998] provide instead an example of *supervised* clustering, as

[10]Some term selection methods, such as wrapper methods, also address the problem of redundancy.

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 (see Section 6.2), showed only a 2% effectiveness loss with an aggressivity of 1000, and even showed some effectiveness improvement with less aggressive levels of reduction. Later experiments by Slonim and Tishby [2001] have confirmed the potential of supervised clustering methods for term extraction.

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

One characteristic of LSI is that the newly obtained dimensions are not, unlike in term selection and term clustering, intuitively interpretable. However, they work well in bringing out the "latent" semantic structure of the vocabulary used in the corpus. For instance, Schütze et al. [1995, page 235] discuss the classification under category `Demographic shifts in the U.S. with economic impact` 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 DR 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 discrimination power may be lost in the new vector space.

Wiener et al. [1995] use LSI in two alternative ways: (i) for local DR, thus creating several category-specific LSI representations, and (ii) for global DR, thus creating a single LSI representation for the entire category set. Their experiments showed the former approach to perform better than the latter, and both approaches to perform better than simple TSR based on Relevancy Score (see Table 1).

Schütze et al. [1995] experimentally compared LSI-based term extraction with χ^2-based TSR using three different classifier learning techniques (namely, linear

discriminant analysis, logistic regression and neural networks). Their experiments showed LSI to be far more effective than χ^2 for the first two techniques, while both methods performed equally well for the neural network classifier.

For other TC works that use LSI or similar term extraction techniques see e.g. [Hull 1994; Li and Jain 1998; Schütze 1998; Weigend et al. 1999; Yang 1995].

6. INDUCTIVE CONSTRUCTION OF TEXT CLASSIFIERS

The inductive construction of text classifiers has been tackled in a variety of ways. Here we will deal only with the methods that have been most popular in TC, but we will also briefly mention the existence of alternative, less standard approaches.

We start by discussing the general form that a text classifier has. Let us recall from Section 2.4 that there are two alternative ways of viewing classification: "hard" (fully automated) classification and ranking (semi-automated) classification.

The inductive construction of a ranking classifier for category $c_i \in \mathcal{C}$ usually consists in the definition of a function $CSV_i : \mathcal{D} \rightarrow [0, 1]$ that, given a document d_j, returns a *categorization status value* for it, i.e. a number between 0 and 1 that, roughly speaking, represents the evidence for the fact that $d_j \in c_i$. Documents are then ranked according to their CSV_i value. This works for "document-ranking TC"; "category-ranking TC" is usually tackled by ranking, for a given document d_j, its CSV_i scores for the different categories in $\mathcal{C} = \{c_1, \ldots, c_{|\mathcal{C}|}\}$.

The CSV_i function takes up different meanings according to the learning method used: for instance, in the "Naïve Bayes" approach of Section 6.2 $CSV_i(d_j)$ is defined in terms of a probability, whereas in the "Rocchio" approach discussed in Section 6.7 $CSV_i(d_j)$ is a measure of vector closeness in $|\mathcal{T}|$-dimensional space.

The construction of a "hard" classifier may follow two alternative paths. The former consists in the definition of a function $CSV_i : \mathcal{D} \rightarrow \{T, F\}$. The latter consists instead in the definition of a function $CSV_i : \mathcal{D} \rightarrow [0, 1]$, analogous to the one used for ranking classification, followed by the definition of a *threshold* τ_i such that $CSV_i(d_j) \geq \tau_i$ is interpreted as T while $CSV_i(d_j) < \tau_i$ is interpreted as F[11].

The definition of thresholds will be the topic of Section 6.1. In Sections 6.2 to 6.12 we will instead concentrate on the definition of CSV_i, discussing a number of approaches that have been applied in the TC literature. In general we will assume we are dealing with "hard" classification; it will be evident from the context how and whether the approaches can be adapted to ranking classification. The presentation of the algorithms will be mostly qualitative rather than quantitative, i.e. will focus on the methods for classifier learning rather than on the effectiveness and efficiency of the classifiers built by means of them; this will instead be the focus of Section 7.

6.1 Determining thresholds

There are various policies for determining the threshold τ_i, also depending on the constraints imposed by the application. The most important distinction is whether the threshold is derived *analytically* or *experimentally*.

The former method is possible only in the presence of a theoretical result that indicates how to compute the threshold that maximizes the expected value of the

[11] Alternative methods are possible, such as training a classifier for which some standard, predefined value such as 0 is the threshold. For ease of exposition we will not discuss them.

effectiveness function [Lewis 1995a]. This is typical of classifiers that output *probability* estimates of the membership of d_j in c_i (see Section 6.2) and whose effectiveness is computed by decision-theoretic measures such as *utility* (see Section 7.1.3); we thus defer the discussion of this policy (which is called *probability thresholding* in [Lewis 1995a]) to Section 7.1.3.

When such a theoretical result is not available one has to revert to the latter method, which consists in testing different values for τ_i on a validation set and choosing the value which maximizes effectiveness. We call this policy *CSV thresholding* [Cohen and Singer 1999; Schapire et al. 1998; Wiener et al. 1995]; it is also called *Scut* in [Yang 1999]. Different τ_i's are typically chosen for the different c_i's.

A second, popular experimental policy is *proportional thresholding* [Iwayama and Tokunaga 1995; Larkey 1998; Lewis 1992a; Lewis and Ringuette 1994; Wiener et al. 1995], also called *Pcut* in [Yang 1999]. This policy consists in choosing the value of τ_i for which $g_{Va}(c_i)$ is closest to $g_{Tr}(c_i)$, and embodies the principle that the same percentage of documents of both training and test set should be classified under c_i. For obvious reasons, this policy does not lend itself to document-pivoted TC.

Sometimes, depending on the application, a *fixed thresholding* policy (aka "k-per-doc" thresholding [Lewis 1992a] 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 TC. However, it suffers from a certain coarseness, as the fact that k is equal for all documents (nor could this be otherwise) allows no fine-tuning.

In his experiments Lewis [1992a] found the proportional policy to be superior to probability thresholding when microaveraged effectiveness was tested but slightly inferior with macroaveraging (see Section 7.1.1). Yang [1999] found instead CSV thresholding to be superior to proportional thresholding (possibly due to her category-specific optimization on a validation set), and found fixed thresholding to be consistently inferior to the other two policies. The fact that these latter 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 hierarchical catalogues.

6.2 Probabilistic classifiers

Probabilistic classifiers (see [Lewis 1998] for a thorough discussion) view $CSV_i(d_j)$ in terms of $P(c_i|\vec{d_j})$, i.e. the probability that a document represented by a vector $\vec{d_j} = \langle w_{1j}, \ldots, w_{|\mathcal{T}|j} \rangle$ of (binary or weighted) terms belongs to c_i, and compute this probability by an application of Bayes' theorem, given by

$$P(c_i|\vec{d_j}) = \frac{P(c_i)P(\vec{d_j}|c_i)}{P(\vec{d_j})} \tag{3}$$

In (3) the event space is the space of documents: $P(\vec{d_j})$ is thus the probability that a randomly picked document has vector $\vec{d_j}$ as its representation, and $P(c_i)$ the probability that a randomly picked document belongs to c_i.

The estimation of $P(\vec{d_j}|c_i)$ in (3) is problematic, since the number of possible vectors $\vec{d_j}$ is too high (the same holds for $P(\vec{d_j})$, 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

$$P(\vec{d_j}|c_i) = \prod_{k=1}^{|\mathcal{T}|} P(w_{kj}|c_i) \tag{4}$$

Probabilistic classifiers that use this assumption are called *Naïve Bayes* classifiers, and account for most of the probabilistic approaches to TC in the literature (see e.g. [Joachims 1998; Koller and Sahami 1997; Larkey and Croft 1996; Lewis 1992a; 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 practice.

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 (4) may be written as

$$P(w_{kj}|c_i) = p_{ki}^{w_{kj}}(1-p_{ki})^{1-w_{kj}} = (\frac{p_{ki}}{1-p_{ki}})^{w_{kj}}(1-p_{ki}) \tag{5}$$

We may further observe that in TC the document space is partitioned into two categories[12], c_i and its complement $\overline{c}_i$, such that $P(\overline{c}_i|\vec{d_j}) = 1 - P(c_i|\vec{d_j})$. If we plug in (4) and (5) into (3) and take logs we obtain

$$\log P(c_i|\vec{d_j}) = \log P(c_i) + \sum_{k=1}^{|\mathcal{T}|} w_{kj} \log \frac{p_{ki}}{1-p_{ki}} + \sum_{k=1}^{|\mathcal{T}|} \log(1-p_{ki}) - \log P(\vec{d_j}) \tag{6}$$

$$\log(1-P(c_i|\vec{d_j})) = \log(1-P(c_i)) + \sum_{k=1}^{|\mathcal{T}|} w_{kj} \log \frac{p_{k\bar{i}}}{1-p_{k\bar{i}}} + \sum_{k=1}^{|\mathcal{T}|} \log(1-p_{k\bar{i}}) - \log P(\vec{d_j}) \tag{7}$$

where we write $p_{k\bar{i}}$ as short for $P(w_{kx} = 1|\overline{c}_i)$. We may convert (6) and (7) into a single equation by subtracting componentwise (7) from (6), thus obtaining

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

[12] Cooper [1995] has pointed out that in this case the full independence assumption of (4) is not actually made in the Naïve Bayes classifier; the assumption needed here is instead the weaker *linked dependence assumption*, which may be written as $\frac{P(\vec{d_j}|c_i)}{P(\vec{d_j}|\overline{c}_i)} = \prod_{k=1}^{|\mathcal{T}|} \frac{P(w_{kj}|c_i)}{P(w_{kj}|\overline{c}_i)}$.

Note that $\frac{P(c_i|\vec{d_j})}{1-P(c_i|\vec{d_j})}$ is an increasing monotonic function of $P(c_i|\vec{d_j})$, and may thus be used directly as $CSV_i(d_j)$. Note also that $\log \frac{P(c_i)}{1-P(c_i)}$ and $\sum_{k=1}^{|\mathcal{T}|} \log \frac{1-p_{ki}}{1-p_{k\bar{i}}}$ are constant for all documents, and may thus be disregarded[13]. Defining a classifier for category c_i thus basically requires estimating the $2|\mathcal{T}|$ parameters $\{p_{1i}, p_{1\bar{i}}, \ldots, p_{|\mathcal{T}|i}, p_{|\mathcal{T}|\bar{i}}\}$ from the training data, which may be done in the obvious way. Note that in general the classification of a given document does not require to compute a sum of $|\mathcal{T}|$ factors, as the presence of $\sum_{k=1}^{|\mathcal{T}|} w_{kj} \log \frac{p_{ki}(1-p_{k\bar{i}})}{p_{k\bar{i}}(1-p_{ki})}$ would imply; in fact, all the factors for which $w_{kj} = 0$ may be disregarded, and this accounts for the vast majority of them, since document vectors are usually very sparse.

The method we have illustrated is just one of the many variants of the Naïve Bayes approach, the common denominator of which is (4). A recent paper by Lewis [1998] is an excellent roadmap on the various directions that research on Naïve Bayes classifiers has taken; among these are the ones aiming

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

—*to introduce document length normalization.* The value of $\log \frac{P(c_i|\vec{d_j})}{1-P(c_i|\vec{d_j})}$ tends to be more extreme (i.e. very high or very low) for long documents (i.e. documents such that $w_{kj} = 1$ for many values of k), irrespectively of their semantic relatedness to c_i, thus calling for length normalization. Taking length into account is easy in non-probabilistic approaches to classification (see e.g. Section 6.7), but is problematic in probabilistic ones (see [Lewis 1998, Section 5]). One possible answer is to switch from an interpretation of Naïve Bayes in which documents are events to one in which terms are events [Baker and McCallum 1998; McCallum et al. 1998; Chakrabarti et al. 1998a; Guthrie et al. 1994]. 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 implausible than (4).

—*to relax the independence assumption.* This may be the hardest route to follow, since this produces classifiers of higher computational cost and characterized by harder parameter estimation problems [Koller and Sahami 1997]. Earlier efforts in this direction within probabilistic text search (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 effectiveness has also been given some theoretical justification [Domingos and Pazzani 1997].

The quotation of text search in the last paragraph is not casual. Unlike other types of classifiers, the literature on probabilistic classifiers is inextricably intertwined with that on probabilistic search systems (see [Crestani et al. 1998] for a review), since these latter attempt to determine the probability that a document falls in the

[13]This is not true, however, if the "fixed thresholding" method of Section 6.1 is adopted. In fact, 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.

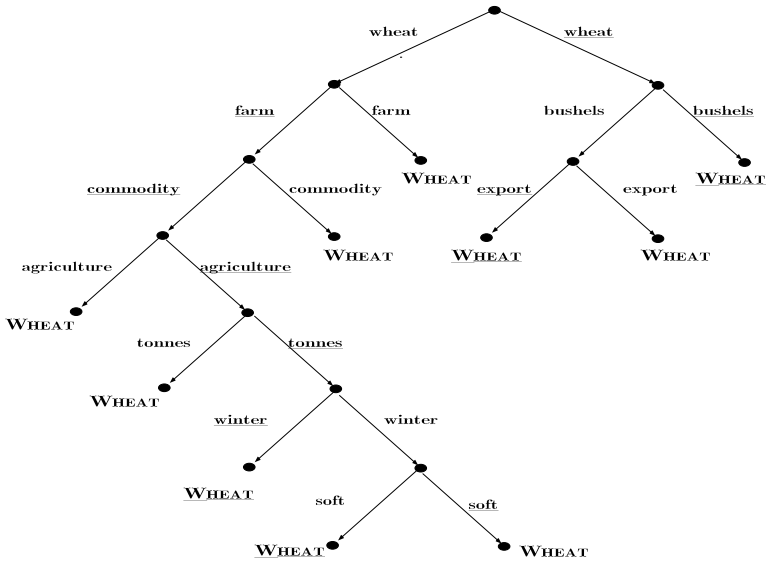

Fig. 2. A decision tree equivalent to the DNF rule of Figure 1. Edges are labelled by terms and leaves are labelled by categories (underlining denotes negation).

category denoted by the query, and since they are the only search systems that take *relevance feedback*, a notion essentially involving supervised learning, as central.

6.3 Decision tree classifiers

Probabilistic methods are quantitative (i.e. numeric) in nature, and as such have sometimes been criticized since, effective as they may be, are not easily interpretable by humans. A class of algorithms that do not suffer from this problem are *symbolic* (i.e. non-numeric) algorithms, among which inductive rule learners (which we will discuss in Section 6.4) and decision tree learners are the most important examples.

A *decision tree* (DT) text classifier (see e.g. [Mitchell 1996, Chapter 3]) is a tree in which internal nodes are labelled by terms, branches departing from them are labelled by tests on the weight that the term has in the test document, and leafs are labelled by categories. Such a classifier categorizes a test document d_j by recursively testing for the weights that the terms labeling the internal nodes have in vector $\vec{d_j}$, until a leaf node is reached; the label of this node is then assigned to d_j. Most such classifiers use binary document representations, and thus consist of binary trees. An example DT is illustrated in Figure 2.

There are a number of standard packages for DT learning, and most DT approaches to TC have made use of one such package. Among the most popular ones are ID3 (used in [Fuhr et al. 1991]), C4.5 (used in [Cohen and Hirsh 1998; Cohen and Singer 1999; Joachims 1998; Lewis and Catlett 1994]) and C5 (used in [Li and Jain 1998]). TC efforts based on experimental DT packages include [Dumais et al. 1998; Lewis and Ringuette 1994; Weiss et al. 1999].

A possible method for learning a DT for category c_i consists in a "divide and conquer" strategy of (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 Tr 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 of the tree so generated contains training examples assigned to the same category c_i, which is then chosen as the label for the leaf. The key step 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 too specific to the training data. Most DT learning methods thus include a method for growing the tree and one for pruning it, i.e. for removing the overly specific branches. Variations on this basic schema for DT learning abound [Mitchell 1996, Section 3].

DT 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 and Singer 2000; Weiss et al. 1999].

6.4 Decision rule classifiers

A classifier for category c_i built by an *inductive rule learning* method consists of a *DNF rule*, i.e. of a conditional rule with a premise in disjunctive normal form (DNF), of the type illustrated in Figure 1[14]. The literals (i.e. possibly negated keywords) in the premise denote the presence (non-negated keyword) or absence (negated keyword) of the keyword in the test document d_j, while the clause head denotes the decision to classify d_j under c_i. DNF rules are similar to DTs in that they can encode any Boolean function. However, an advantage of DNF rule learners is that they tend to generate more compact classifiers than DT learners.

Rule learning methods usually attempt to select from all the possible covering rules (i.e. rules that correctly classify all the training examples) the "best" one according to some minimality criterion. While DTs are typically built by a top-down, "divide-and-conquer" strategy, DNF rules are often built in a bottom-up fashion. Initially, every training example d_j is viewed as a clause $\eta_1, \ldots, \eta_n \rightarrow \gamma_i$, where $\eta_1, \ldots, \eta_n$ are the terms contained in d_j and γ_i equals c_i or $\overline{c}_i$ according to whether d_j is a positive or negative example of c_i. This set of clauses is already a DNF classifier for c_i, but obviously scores high in terms of overfitting. The learner applies then a process of generalization in which the rule is simplified through a series of modifications (e.g. removing premises from clauses, or merging clauses) that maximize its compactness while at the same time not affecting the "covering" property of the classifier. At the end of this process, a "pruning" phase similar in spirit to that employed in DTs is applied, where the ability to correctly classify *all* the training examples is traded for more generality.

DNF rule learners vary widely in terms of the methods, heuristics and criteria employed for generalization and pruning. Among the 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 Hirsh 1998; Cohen and Singer

[14]Many inductive rule learning algorithms build *decision lists* (i.e. arbitrarily nested `if-then-else` clauses) instead of DNF rules; since the former may always be rewritten as the latter we will disregard the issue.

1999], SCAR [Moulinier et al. 1996], and SWAP-1 [Apté et al. 1994].

While the methods above use rules of propositional logic (PL), research has also been carried out using rules of first order logic (FOL), obtainable through the use of *inductive logic programming* methods. Cohen [1995a] has extensively compared PL and FOL learning in TC (for instance, comparing the PL learner RIPPER with its FOL version FLIPPER), and has found that the additional representational power of FOL brings about only modest benefits.

6.5 Regression methods

Various TC efforts have used regression models (see e.g. [Fuhr and Pfeifer 1994; Ittner et al. 1995; Lewis and Gale 1994; Schütze et al. 1995]). *Regression* denotes the approximation of a *real-valued* (instead than binary, as in the case of classification) function $\breve{\Phi}$ by means of a function Φ that fits the training data [Mitchell 1996, page 236]. Here we will describe one such model, the *Linear Least Squares Fit* (LLSF) applied to TC by Yang and Chute [1994]. In LLSF, each document d_j has two vectors associated to it: an *input vector* $I(d_j)$ of $|\mathcal{T}|$ weighted terms, and an *output vector* $O(d_j)$ of $|\mathcal{C}|$ weights representing the categories (the weights for this latter vector are binary for training documents, and are non-binary CSVs for 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 a $|\mathcal{C}| \times |\mathcal{T}|$ matrix $\hat{M}$ such that $\hat{M}I(d_j) = O(d_j)$.

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 x is minimum, $\|V\|_F \stackrel{def}{=} \sqrt{\sum_{i=1}^{|\mathcal{C}|} \sum_{j=1}^{|\mathcal{T}|} v_{ij}^2}$ represents the so-called *Frobenius norm* of a $|\mathcal{C}| \times |\mathcal{T}|$ matrix, I is the $|\mathcal{T}| \times |Tr|$ matrix whose columns are the input vectors of the training documents, and O is the $|\mathcal{C}| \times |Tr|$ 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, and its generic entry $\hat{m}_{ik}$ represents the degree of association between category c_i and term t_k.

The experiments of [Yang and Chute 1994; Yang and Liu 1999] indicate that LLSF is one of the most effective text classifiers known to date. One of its disadvantages, though, is that the cost of computing the $\hat{M}$ matrix is much higher than that of many other competitors in the TC arena.

6.6 On-line methods

A *linear classifier* for category c_i is a vector $\vec{c}_i = \langle w_{1i}, \ldots, w_{|\mathcal{T}|i} \rangle$ belonging to the same $|\mathcal{T}|$-dimensional space in which documents are also represented, and such that $CSV_i(d_j)$ corresponds to the dot product $\sum_{k=1}^{|\mathcal{T}|} w_{ki} w_{kj}$ of $\vec{d}_j$ and $\vec{c}_i$. Note that when both classifier and document weights are cosine-normalized (see (2)), the dot product between the two vectors corresponds to their *cosine similarity*, i.e.

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

which represents the cosine of the angle α that separates the two vectors. 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 built, classification can be performed by invoking such an engine. Practically all search engines have a dot product flavour to them, and can therefore be adapted to doing TC with a linear classifier.

Methods for learning linear classifiers are often partitioned in two broad classes, batch methods and on-line methods.

Batch methods build a classifier by analysing the training set all at once. Within the TC literature, one example of a batch method is *linear discriminant analysis*, a model of the stochastic dependence between terms that relies on the covariance matrices of the categories [Hull 1994; Schütze et al. 1995]. However, the foremost example of a batch method is the Rocchio method; because of its importance in the TC literature this will be discussed separately in Section 6.7. In this section we will instead concentrate on on-line classifiers.

On-line (aka *incremental*) *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 the applications in which Tr is not available in its entirety from the start, or in which the "meaning" of the category may change in time, as e.g. in adaptive filtering. This is also apt to applications (e.g. semi-automated classification, adaptive filtering) 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.

A simple on-line method is the *perceptron* algorithm, first applied to TC in [Schütze et al. 1995; Wiener et al. 1995] and subsequently used in [Dagan et al. 1997; Ng et al. 1997]. In this algorithm, the classifier for c_i is first initialized by setting all weights w_{ki} to the same positive value. When a training example d_j (represented by a vector $\vec{d}_j$ of binary weights) is examined, the classifier built so far classifies it. If the result of the classification is correct nothing is done, while if it is wrong the weights of the classifier are modified: if d_j was a positive example of c_i then the weights w_{ki} of "active terms" (i.e. the terms t_k such that $w_{kj} = 1$) are "promoted" by increasing them by a fixed quantity $\alpha > 0$ (called *learning rate*), while if d_j was a negative example of c_i then the same weights are "demoted" by decreasing them by α. Note that when 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 process so far, and may thus be discarded from the representation. We may then see the perceptron algorithm (as all other incremental learning methods) 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 is an *additive weight-updating* algorithm. A *multiplicative* variant of it 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] 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 dot product is the difference $w^+_{ki} - w^-_{ki}$. Following the misclassifica-

tion 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 demoted while w_{ki}^{-} gets promoted (for the rest, promotions and demotions are as in POSITIVE WINNOW). BALANCED WINNOW allows negative w_{ki} weights, while in the perceptron 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 perceptron but was in turn outperformed by (Dagan et al.'s own version of) BALANCED WINNOW.

Other on-line methods for building text classifiers are WIDROW-HOFF, a refinement of it called EXPONENTIATED GRADIENT (both applied for the first time to TC in [Lewis et al. 1996]) and SLEEPING EXPERTS [Cohen and Singer 1999], 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. For the former the classifier $\vec{c}_i$ is used, in a standard search engine, as a query against the set of test documents, while for the latter the vector $\vec{d}_j$ representing the test document is used as a query against the set of classifiers $\{\vec{c}_1, \ldots, \vec{c}_{|\mathcal{C}|}\}$.

6.7 The Rocchio method

Some linear classifiers consist of an explicit *profile* (or prototypical document) of the category. This has obvious advantages in terms of interpretability, as such a profile is more readily understandable by a human than, say, a neural network classifier. Learning a linear classifier is often preceded by local TSR; in this case, a profile of c_i is a weighted list of the terms whose presence or absence is most useful for discriminating c_i.

The *Rocchio method* is used for inducing linear, profile-style classifiers. It relies on an adaptation to TC of the well-known Rocchio's formula for relevance feedback in the vector-space model, and it is perhaps the only TC method rooted in the IR tradition rather than in the ML one. This adaptation was first proposed by Hull [1994], and has been used by many authors since then, either as an object of research in its own right [Ittner et al. 1995; Joachims 1997; Sable and Hatzivassiloglou 2000; Schapire et al. 1998; Singhal et al. 1997], or as a baseline classifier [Cohen and Singer 1999; Galavotti et al. 2000; 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 6.11).

Rocchio's method computes a classifier $\vec{c}_i = \langle w_{1i}, \ldots, w_{|\mathcal{T}|i} \rangle$ for category c_i by means of the formula

$$w_{ki} = \beta \cdot \sum_{\{d_j \in POS_i\}} \frac{w_{kj}}{|POS_i|} - \gamma \cdot \sum_{\{d_j \in NEG_i\}} \frac{w_{kj}}{|NEG_i|}$$

where w_{kj} is the weight of t_k in document d_j, $POS_i = \{d_j \in Tr \mid \breve{\Phi}(d_j, c_i) = T\}$ and $NEG_i = \{d_j \in Tr \mid \breve{\Phi}(d_j, c_i) = F\}$. 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;

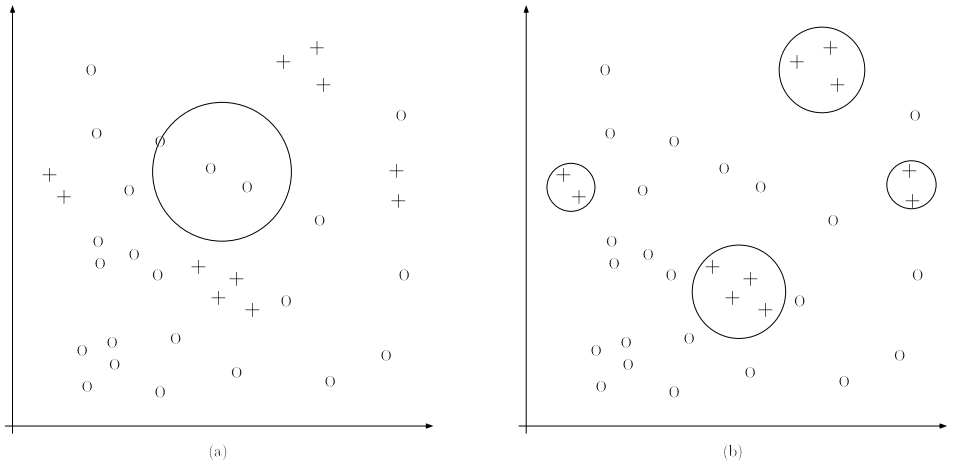

Fig. 3. A comparison between the TC behaviour of (a) the Rocchio classifier, and (b) the k-NN classifier. Small crosses and circles denote positive and negative training instances, respectively. The big circles denote the "influence area" of the classifier. Note that, for ease of illustration, document similarities are here viewed in terms of Euclidean distance rather than, as more common, in terms of dot product or cosine.

Joachims 1998; Schütze et al. 1995]), the profile of c_i is the *centroid* of its positive training examples. A classifier built by means of the Rocchio method 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. The role of negative examples is usually de-emphasized, 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 is also quite efficient, since learning a classifier basically comes down to averaging weights. In terms of effectiveness, instead, a drawback is that if the documents in the category tend to occur in disjoint clusters (e.g. a set of newspaper articles lebelled with the `Sports` category and dealing with either boxing or rock-climbing), such a classifier may miss most of them, as the centroid of these documents may fall outside all of these clusters (see Figure 3a). More generally, a classifier built by the Rocchio method, as all linear classifiers, has the disadvantage that it divides the space of documents linearly. This situation is graphically depicted in Figure 3a, 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.

6.7.1 *Enhancements to the basic Rocchio framework.* One issue in the application of the Rocchio formula to profile extraction is whether the set NEG_i should be considered in its entirety, or whether a well-chosen sample of it, such as the set $NPOS_i$ of *near-positives* (defined as "the most positive amongst the negative training examples"), should be selected from it, yielding

$$w_{ki} \;=\; \beta \cdot \sum_{\{d_j \in POS_i\}} \frac{w_{kj}}{|POS_i|} - \gamma \cdot \sum_{\{d_j \in NPOS_i\}} \frac{w_{kj}}{|NPOS_i|}$$

The $\sum_{\{d_j \in NPOS_i\}} \frac{w_{kj}}{|NPOS_i|}$ factor is more significant than $\sum_{\{d_j \in NEG_i\}} \frac{w_{kj}}{|NEG_i|}$, since near-positives are the most difficult documents to tell apart from the positives. 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 are among the ones that had scored highest in the previous ranking. Early applications of the Rocchio formula to TC (e.g. [Hull 1994; Ittner et al. 1995]) generally did not make a distinction between near-positives and generic negatives. In order to select the near-positives Schapire et al. [1998] issue a 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 are then the near-positives. Wiener et al. [1995] instead equate the near-positives of c_i to the positive examples of the *sibling* categories of c_i, as in the application they work on (TC with hierarchical category sets) the notion of a "sibling category of c_i" is well-defined. A similar policy is also adopted in [Ng et al. 1997; Ruiz and Srinivasan 1999; Weigend et al. 1999].

By using query zoning plus other enhancements (TSR, statistical phrases, and a method called *dynamic feedback optimization*), Schapire et al. [1998] have found that a Rocchio classifier can achieve an effectiveness comparable to that of a state-of-the-art ML method such as "boosting" (see Section 6.11.1) while being 60 times quicker to train. These recent results will no doubt bring about a renewed interest for the Rocchio classifier, previously considered an underperformer [Cohen and Singer 1999; Joachims 1998; Lewis et al. 1996; Schütze et al. 1995; Yang 1999].

6.8 Neural networks

A *neural network* (NN) text classifier is a network of units, where the input units represent terms, the output unit(s) represent the category or categories of interest, and the weights on the edges connecting units represent dependence relations. For classifying a test document d_j, its term weights w_{kj} are loaded into the input units; the activation of these units is propagated forward through the network, and the value of the output unit(s) determines the categorization decision(s). A typical way of training NNs 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 minimize the error.

The simplest type of NN 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 6.6. Other types of linear NN classifiers implementing a form of logistic regression have also been proposed and tested by Schütze et al. [1995] and Wiener et al. [1995], yielding very good effectiveness.

A non-linear NN [Lam and Lee 1999; 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 NNs 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.

6.9 Example-based classifiers

Example-based classifiers do not build an explicit, declarative representation of the category c_i, but rely on the category labels attached to the training documents similar to the test document. These methods have thus been called *lazy learners*, 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 application of example-based methods (aka *memory-based reasoning methods*) to TC is due to Creecy, Masand and colleagues [Creecy et al. 1992; Masand et al. 1992]; examples 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 used by Yang [1994]. For deciding whether $d_j \in c_i$, k-NN looks at whether the k training documents most similar to d_j also are in c_i; if the answer is positive for a large enough proportion of them, a positive 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 is in c_i is weighted by its similarity with the test document. Classifying d_j by means of k-NN thus comes down to computing

$$CSV_i(d_j) = \sum_{d_z \in Tr_k(d_j)} RSV(d_j, d_z) \cdot [\![\breve{\Phi}(d_z, c_i)]\!] \tag{9}$$

where $Tr_k(d_j)$ is the set of the k documents d_z which maximize $RSV(d_j, d_z)$ and

$$[\![\alpha]\!] = \begin{cases} 1 & \text{if } \alpha = T \\ 0 & \text{if } \alpha = F \end{cases}$$

The thresholding methods of Section 6.1 can then be used to convert the real-valued CSV_i's into binary categorization decisions. In (9), $RSV(d_j, d_z)$ represents some measure or semantic relatedness between a test document d_j and a training document 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 construction of a k-NN classifier also involves determining (experimentally, on a validation set) a threshold k that indicates how many top-ranked training documents have to be considered for computing $CSV_i(d_j)$. 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 divide the document space linearly, hence does not suffer from the problem discussed at the end of Section 6.7. This is graphically depicted in Figure 3b, where the more "local" character of k-NN with respect to Rocchio can be appreciated.

This method is naturally geared towards document-pivoted TC, since ranking the training documents for their similarity with the test document can be done once for all categories. For category-pivoted TC one would need to store the document ranks for each test document, which is obviously clumsy; DPC is thus *de facto* the only reasonable way to use k-NN.

A number of different experiments (see Section 7.3) have shown k-NN to be quite effective. However, its most important drawback is its inefficiency at classification time: while e.g. with a linear classifier only a dot product needs to be computed to classify a test document, k-NN requires the entire training set to be ranked for similarity with the test document, which is much more expensive. This is a drawback of "lazy" learning methods, since they do not have a true training phase and thus defer all the computation to classification time.

6.9.1 *Other example-based techniques.* Various example-based techniques have been used in the TC literature. For example, Cohen and Hirsh [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

$$CSV_i(d_j) \;=\; 1 - \prod_{d_z \in\, Tr_k(d_j)} (1 - RSV(d_j, d_z))^{[\![\breve{\Phi}(d_z, c_i)]\!]}$$

as an alternative to (9), obtaining a small but statistically significant improvement over a version of WHIRL using (9). In their experiments this technique outperformed a number of other classifiers, such as a C4.5 decision tree classifier and the RIPPER CNF rule-based classifier.

A variant of the basic k-NN approach is proposed by Galavotti et al. [2000], who reinterpret (9) by redefining $[\![\alpha]\!]$ as

$$[\![\alpha]\!] = \begin{cases} 1 & \text{if } \alpha = T \\ -1 & \text{if } \alpha = F \end{cases}$$

The difference from the original k-NN approach is that if a training document d_z similar to the test document d_j does not belong to c_i, this information is not discarded but 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 *generalized instances* (GIs) in place of training documents. This approach may be seen as the result of

—clustering the training set, thus obtaining a set of clusters $K_i = \{k_{i1}, \ldots, k_{i|K_i|}\}$;
—building a profile $G(k_{iz})$ ("generalized instance") from the documents belonging to cluster k_{iz} by means of some algorithm for learning linear classifiers (e.g. Rocchio, WIDROW-HOFF);
—applying k-NN with profiles in place of training documents, i.e. computing

$$\begin{aligned} CSV_i(d_j) &\stackrel{def}{=} \sum_{k_{iz} \in K_i} RSV(d_j, G(k_{iz})) \cdot \frac{|\{d_j \in k_{iz}|\ \breve{\Phi}(d_j, c_i) = T\}|}{|\{d_j \in k_{iz}\}|} \cdot \frac{|\{d_j \in k_{iz}\}|}{|Tr|} \\ &= \sum_{k_{iz} \in K_i} RSV(d_j, G(k_{iz})) \cdot \frac{|\{d_j \in k_{iz}|\ \breve{\Phi}(d_j, c_i) = T\}|}{|Tr|} \end{aligned} \tag{10}$$

where $\frac{|\{d_j \in k_{iz}|\ \breve{\Phi}(d_j, c_i) = T\}|}{|\{d_j \in k_{iz}\}|}$ represents the "degree" to which $G(k_{iz})$ is a positive instance of c_i, and $\frac{|\{d_j \in k_{iz}\}|}{|Tr|}$ represents its weight within the entire process.

This exploits the superior effectiveness (see Figure 3) of k-NN over linear classifiers while at the same time avoiding the sensitivity of k-NN to the presence of "outliers"

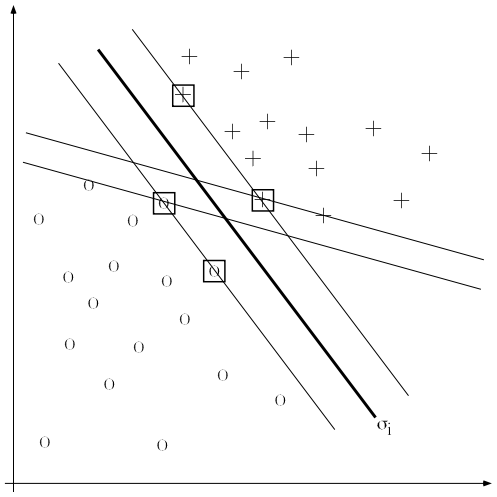

Fig. 4. Learning support vector classifiers. The small crosses and circles represent positive and negative training examples, respectively, whereas lines represent decision surfaces. Decision surface σ_i (indicated by the thicker line) is, among those shown, the best possible one, as it is the middle 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.

(i.e. positive instances of c_i that "lie out" of the region where most other positive instances of c_i are located) in the training set.

6.10 Building classifiers by support vector machines

The *support vector machine* (SVM) method has been introduced in TC by Joachims [1998, 1999] and subsequently used in [Drucker et al. 1999; Dumais et al. 1998; Dumais and Chen 2000; Klinkenberg and Joachims 2000; Taira and Haruno 1999; Yang and Liu 1999]. In geometrical terms, it may be seen as the attempt to find, among all the surfaces $\sigma_1, \sigma_2, \ldots$ in $|\mathcal{T}|$-dimensional space that separate the positive from the negative training examples (*decision surfaces*), the σ_i that separates the positives from the negatives by the widest possible margin, i.e. such that the separation property is invariant with respect to the widest possible traslation of σ_i.

This idea is best understood in the case in which the positives and the negatives are linearly separable, in which case the decision surfaces are $(|\mathcal{T}|-1)$-hyperplanes. In the 2-dimensional case of Figure 4, various lines may be chosen as decision surfaces. The SVM method chooses the middle element from the "widest" set of parallel lines, i.e. from the set in which the maximum distance between two elements in the set is highest. It is noteworthy that this "best" decision surface is determined by only a small set of training examples, called the *support vectors*.

The method described is applicable also to the case in which the positives and the negatives 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.

As argued by Joachims [1998], SVMs offer two important advantages for TC:

—term selection is often not needed, as SVMs tend to be fairly robust to 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 applied a novel algorithm for training SVMs which brings about training speeds comparable to computationally easy learners such as Rocchio.

6.11 Classifier committees

Classifier *committees* (aka *ensembles*) are based on the idea that, given a task that requires expert knowledge to perform, 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 $d_j \in c_i$, and then combine their outcome appropriately. A classifier committee is then characterized by (i) a choice of k classifiers, and (ii) a choice of a combination function.

Concerning issue (i), it is known from the ML literature that, in order to guarantee good effectiveness, the classifiers forming the committee should be as independent as possible [Tumer and Ghosh 1996]. The classifiers may differ for the indexing approach used, or for the inductive method, or both. Within TC, the avenue which has been explored most is the latter (to our knowledge the only example of the former is [Scott and Matwin 1999]).

Concerning issue (ii), various rules have been tested. The simplest one is *majority voting* (MV), whereby the binary outputs of 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 characterized by a binary decision function $CSV_i : \mathcal{D} \rightarrow \{T, F\}$. A second rule is *weighted linear combination* (WLC), whereby a weighted sum of the CSV_i's produced by the k classifiers yields the final CSV_i. The weights w_j reflect the expected relative effectiveness of classifiers Φ_j, and are usually optimized on a validation set [Larkey and Croft 1996]. Another policy is *dynamic classifier selection* (DCS), whereby among committee $\{\Phi_1, \ldots, \Phi_k\}$ the classifier Φ_t most effective 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 their effectiveness 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 outperformed the best individual classifier (k-NN), and the combination of the three classifiers 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 drawn.

Li and Jain [1998] have tested a committee formed of (various combinations of) a Naïve Bayes classifier, an example-based classifier, a decision tree classifier, and a classifier built 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 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 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.

6.11.1 *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 $\Phi_1, \ldots, \Phi_k$ forming the committee are obtained by the *same* learning method (here called the *weak learner*). The key intuition of boosting is that the k classifiers should be trained not in a conceptually parallel and independent way, as in the committees described above, but sequentially. In this way, in training classifier Φ_i one 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 learning classifier Φ_t each $\langle d_j, c_i \rangle$ pair is given an "importance weight" h_{ij}^t (where h_{ij}^1 is set to be equal for all $\langle d_j, c_i \rangle$ pairs[15]), which represents how hard to get a correct decision for this pair was for classifiers $\Phi_1, \ldots, \Phi_{t-1}$. These weights are exploited in learning Φ_t, which will be specially tuned to correctly solve the pairs with higher weight. 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 weight decreased, while pairs misclassified by Φ_t will have their weight increased. After all the k classifiers have been built, a weighted linear combination rule is applied to yield the final committee.

In the BoosTexter system [Schapire and Singer 2000], two different boosting algorithms are tested, 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 proven 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'

[15]Schapire et al. [1998] also show that a simple modification of this policy allows optimization of the classifier based on "utility" (see Section 7.1.3).

[1997] PrTFIDF classifier.

A boosting algorithm based on a "committee of classifier sub-committees" that improves on the effectiveness and (especially) the efficiency of AdaBoost.MH is presented in [Sebastiani et al. 2000]. An approach similar to boosting is also employed by Weiss et al. [1999], who 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 [Schapire and Singer 2000]), 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 used. Boosting-based approaches have also been employed in [Escudero et al. 2000; Iyer et al. 2000; Kim et al. 2000; Li and Jain 1998; Myers et al. 2000].

6.12 Other methods

Although in the previous sections we have tried to give an overview as complete as possible of the learning approaches proposed in the TC literature, it would be hardly possible to be exhaustive. Some of the learning approaches adopted do not fall squarely under one or the other class of algorithms, or have remained somehow isolated attempts. Among these, the most noteworthy are the ones based on *Bayesian inference networks* [Dumais et al. 1998; Lam et al. 1997; Tzeras and Hartmann 1993], *genetic algorithms* [Clack et al. 1997; Masand 1994], and *maximum entropy modelling* [Manning and Schütze 1999].

7. EVALUATION OF TEXT CLASSIFIERS

As for text search systems, the evaluation of document classifiers is typically conducted *experimentally*, rather than analytically. The reason 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 membership of a document in a category) is, due to its subjective character, inherently non-formalisable.

The experimental evaluation of a classifier usually measures its *effectiveness* (rather than its efficiency), i.e. its ability to take the *right* classification decisions.

7.1 Measures of text categorization effectiveness

7.1.1 *Precision and recall.* Classification effectiveness is usually measured in terms of the classic IR notions of precision (π) and recall (ρ), adapted to the case of TC. *Precision wrt* c_i (π_i) is defined as the conditional probability $P(\breve{\Phi}(d_x, c_i) = T \mid \Phi(d_x, c_i) = T)$, 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 (ρ_i) is defined as $P(\Phi(d_x, c_i) = T \mid \breve{\Phi}(d_x, c_i) = T)$, i.e. as the probability 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 π and ρ, i.e. values global to the entire category set. Borrowing terminology from logic, π may be viewed as the "degree of soundness" of the classifier wrt $\mathcal{C}$, while ρ may be viewed as its "degree of completeness" wrt $\mathcal{C}$. As defined here, π_i and ρ_i are to be understood as *subjective* probabilities, i.e. as measuring the expectation of the user that the system will behave correctly when classifying an unseen document under

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.

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

Table 3. The global contingency table.

c_i. These probabilities may be estimated in terms of the *contingency table* for c_i on a given test set (see Table 2). Here, FP_i (*false positives wrt* c_i, aka *errors of commission*) is the number of test documents 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, aka *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

$$\hat{\pi}_i = \frac{TP_i}{TP_i + FP_i} \qquad\qquad \hat{\rho}_i = \frac{TP_i}{TP_i + FN_i}$$

For obtaining estimates of π and ρ, two different methods may be adopted:

—*microaveraging*: π and ρ are obtained by summing over all individual decisions:

$$\hat{\pi}^{\mu} = \frac{TP}{TP + FP} = \frac{\sum_{i=1}^{\lvert\mathcal{C}\rvert} TP_i}{\sum_{i=1}^{\lvert\mathcal{C}\rvert}(TP_i + FP_i)}$$

$$\hat{\rho}^{\mu} = \frac{TP}{TP + FN} = \frac{\sum_{i=1}^{\lvert\mathcal{C}\rvert} TP_i}{\sum_{i=1}^{\lvert\mathcal{C}\rvert}(TP_i + FN_i)}$$

where "μ" indicates microaveraging. The "global" contingency table (Table 3) is thus obtained by summing over 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:

$$\hat{\pi}^{M} = \frac{\sum_{i=1}^{\lvert\mathcal{C}\rvert} \hat{\pi}_i}{\lvert\mathcal{C}\rvert} \qquad\qquad \hat{\rho}^{M} = \frac{\sum_{i=1}^{\lvert\mathcal{C}\rvert} \hat{\rho}_i}{\lvert\mathcal{C}\rvert}$$

where "M" indicates macroaveraging.

These two methods may give quite different results, especially if the different categories have very different generality. For instance, the ability of a classifier to behave well also on categories with low generality (i.e. categories with few positive training instances) will be emphasized by macroaveraging and much less so by

microaveraging. Whether one or the other should be used obviously depends on the application requirements. From now on, we will assume that microaveraging is used; everything we will say in the rest of Section 7 may be adapted to the case of macroaveraging in the obvious way.

7.1.2 *Other measures of effectiveness.* Measures alternative to π and ρ and commonly used in the ML literature, such as *accuracy* (estimated as $\hat{A} = \frac{TP+TN}{TP+TN+FP+FN}$) and *error* (estimated as $\hat{E} = \frac{FP+FN}{TP+TN+FP+FN} = 1 - \hat{A}$), are not widely used in TC. The reason is that, as Yang [1999] points out, the large value that their denominator typically has in TC makes them much more insensitive to variations in the number of correct decisions ($TP + TN$) than π and ρ. Besides, if A is the adopted evaluation measure, in the frequent case of a very low average generality the *trivial rejector* (i.e. the classifier Φ such that $\Phi(d_j, c_i) = F$ for all d_j and c_i) tends to outperform all non-trivial classifiers (see also [Cohen 1995a, Section 2.3]). If A is adopted, parameter tuning on a validation set may thus 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 [2000, Section 7], who suggest to base π and ρ not on "absolute" values of success and failure (i.e. 1 if $\Phi(d_j, c_i) = \breve{\Phi}(d_j, c_i)$ and 0 if $\Phi(d_j, c_i) \neq \breve{\Phi}(d_j, c_i)$), but on values of *relative success* (i.e. $CSV_i(d_j)$ if $\breve{\Phi}(d_j, c_i) = T$ and $1 - CSV_i(d_j)$ if $\breve{\Phi}(d_j, c_i) = F$). This means that for a correct (resp. wrong) decision the classifier is rewarded (resp. penalized) proportionally to its confidence in the decision. This proposed measure does not reward the choice of a good thresholding policy, and is thus unfit for autonomous ("hard") classification systems. However, it might be appropriate for interactive ("ranking") 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.

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

An important alternative to effectiveness is *utility*, a class of measures from decision theory that extend effectiveness by economic criteria such as *gain* or *loss*. Utility is based on a *utility matrix* such as that of Table 4, where the numeric values u_{TP}, u_{FP}, u_{FN} and u_{TN} represent the 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 special case of utility, i.e. the one in which $u_{TP} = u_{TN} > u_{FP} = u_{FN}$. Less trivial cases are those in which $u_{TP} \neq u_{TN}$ and/or $u_{FP} \neq u_{FN}$; this is the case e.g. in spam filtering, where failing to discard a piece of junk mail (FP) is a less serious mistake than discarding

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

Table 4. The utility matrix.

a legitimate message (FN) [Androutsopoulos et al. 2000]. If the classifier outputs probability estimates of the membership of d_j in c_i, then decision theory provides *analytical* methods to determine thresholds τ_i, thus avoiding the need to determine them experimentally (as discussed in Section 6.1). Specifically, as Lewis [1995a] reminds, the expected value of utility is maximized when

$$\tau_i = \frac{(u_{FP} - u_{TN})}{(u_{FN} - u_{TP}) + (u_{FP} - u_{TN})}$$

which, in the case of "standard" effectiveness, is equal to $\frac{1}{2}$.

The use of utility in TC is discussed in detail by Lewis [1995a]. Other works where utility is employed are [Amati and Crestani 1999; Cohen and Singer 1999; Hull et al. 1996; Lewis and Catlett 1994; Schapire et al. 1998]. Utility has become popular within the text filtering community, and the TREC "filtering track" evaluations have been using it since long [Lewis 1995c]. The values of the utility matrix are extremely application-dependent. This means that if utility is used instead of "pure" effectiveness, there is a further element of difficulty in the cross-comparison of classification systems (see Section 7.3), since for two classifiers to be experimentally comparable also the two utility matrices must be the same.

Other effectiveness measures different from the ones discussed here 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 them in detail. However, their use shows that, although the TC community is making consistent efforts at standardising experimentation protocols, we are still far from universal agreement on evaluation issues and, as a consequence, from understanding precisely the relative merits of the various methods.

7.1.4 *Combined effectiveness measures.* Neither precision nor recall make sense in isolation of each other. In fact the classifier Φ such that $\Phi(d_j, c_i) = T$ for all d_j and c_i (the *trivial acceptor*) has $\rho = 1$. When the CSV_i function has values in $[0, 1]$ one only needs to set every threshold τ_i to 0 to obtain the trivial acceptor. In this case π would usually be very low (more precisely, equal to the average test set generality $\frac{\sum_{i=1}^{|\mathcal{C}|} g_{Te}(c_i)}{|\mathcal{C}|}$)[16]. Conversely, it is well-known from everyday IR practice that higher levels of π may be obtained at the price of low values of ρ.

[16]From this one might be tempted to infer, by symmetry, that the trivial rejector always has $\pi = 1$. This is false, as π is undefined (the denominator is zero) for the trivial rejector (see Table 5). In fact, it is clear from its definition ($\pi = \frac{TP}{TP+FP}$) that π depends only on how the positives ($TP + FP$) are split between *true* positives TP and the *false* positives FP, and does not depend

		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 TC.

In practice, by tuning τ_i a function $CSV_i : \mathcal{D} \rightarrow \{T, F\}$ is tuned to be, in the words of Riloff and Lehnert [1994], more *liberal* (i.e. improving ρ_i to the detriment of π_i) or more *conservative* (improving π_i to the detriment of ρ_i)[17]. A classifier should thus be evaluated by means of a measure which combines π and ρ[18]. Various such measures have been proposed, among which the most frequent are:

(1) *11-point average precision*: threshold τ_i is repeatedly tuned so as to allow ρ_i to take up values of 0.0, .1, ..., .9, 1.0; π_i is computed for these 11 different values of τ_i, and averaged over the 11 resulting values. This is analogous to the standard evaluation methodology for ranked IR systems, and may be used
   (a) with categories in place of IR queries. This is most frequently used for document-ranking classifiers (see e.g [Schütze et al. 1995; Yang 1994; Yang 1999; Yang and Pedersen 1997]);
   (b) with test documents in place of IR queries and categories in place of documents. This is most frequently used for category-ranking classifiers (see e.g. [Lam et al. 1999; Larkey and Croft 1996; Schapire and Singer 2000; Wiener et al. 1995]). In this case if macroaveraging is used it needs to be redefined on a per-document, rather than per-category basis.

   This measure does not make sense for binary-valued CSV_i functions, since in this case ρ_i may not be varied at will.

---

at all on the cardinality of the positives. There is a breakup of "symmetry" between π and ρ 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 ρ ($\frac{TP}{TP+FN}$) is not π ($\frac{TP}{TP+FP}$) but *c-precision* ($\pi^c = \frac{TN}{FP+TN}$), the "contrapositive" of π. In fact, while ρ=1 and π^c=0 for the trivial acceptor, π^c=1 and ρ=0 for the trivial rejector.

[17] While ρ_i can *always* be increased at will by lowering τ_i, *usually* at the cost of decreasing π_i, π_i can *usually* be increased at will by raising τ_i, *always* at the cost of decreasing ρ_i. This kind of tuning is only possible for CSV_i functions with values in $[0, 1]$; for binary-valued CSV_i functions tuning is not always possible, or is anyway more difficult (see e.g. [Weiss et al. 1999, page 66]).

[18] An exception is single-label TC, in which π and ρ are not independent of each other: if a document d_j has been classified under a wrong category c_s (thus decreasing π_s) this also means that it has *not* been classified under the right category c_t (thus decreasing ρ_t). In this case either π or ρ can be used as a measure of effectiveness.

(2) the *breakeven* point, i.e. the value at which π equals ρ (e.g. [Apté et al. 1994; Cohen and Singer 1999; Dagan et al. 1997; Joachims 1998; Joachims 1999; Lewis 1992a; Lewis and Ringuette 1994; Moulinier and Ganascia 1996; Ng et al. 1997; Yang 1999]). This is obtained by a process analogous to the one used for 11-point average precision: a plot of π as a function of ρ is computed by repeatedly varying the thresholds τ_i; breakeven is the value of ρ (or π) for which the plot intersects the $\rho = \pi$ line. This idea relies on the fact that by decreasing the τ_i's from 1 to 0, ρ always increases monotonically from 0 to 1 and π usually decreases monotonically from a value near 1 to $\frac{1}{|\mathcal{C}|}\sum_{i=1}^{|\mathcal{C}|} g_{Te}(c_i)$. If for no values of the τ_i's π and ρ are exactly equal, the τ_i's are set to the value for which π and ρ are closest, and an *interpolated breakeven* is computed as the average of the values of π and ρ[19].

(3) the F_β *function* [van Rijsbergen 1979, Chapter 7], 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

$$F_\beta = \frac{(\beta^2 + 1)\pi\rho}{\beta^2\pi + \rho}$$

Here β may be seen as the relative degree of importance attributed to π and ρ. If $\beta = 0$ then F_β coincides with π, whereas if $\beta = +\infty$ then F_β coincides with ρ. Usually, a value $\beta = 1$ is used, which attributes equal importance to π and ρ. As shown in [Moulinier et al. 1996; Yang 1999], the breakeven 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 parameters can be set) so that the resulting effectiveness is the best achievable by that classifier. Tuning a parameter p (be it a threshold or other) is normally done experimentally. This means performing repeated experiments on the validation set 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. The value that has yielded the best effectiveness is chosen for p.

7.2 Benchmarks for text categorization

Standard *benchmark collections* that can be used as initial corpora for TC are publically available for experimental purposes. 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 so far. Unfortunately, this does not always translate into reliable comparative

[19] Breakeven, first proposed by Lewis [1992a, 1992b], has been recently criticized. Lewis himself (see his message of 11 Sep 1997 10:49:01 to the DDLBETA text categorization mailing list – quoted with permission of the author) points out that breakeven is not a good effectiveness measure, since (i) there may be no parameter setting that yields the breakeven; in this case the final breakeven value, obtained by interpolation, is artificial; (ii) to have ρ equal π is not necessarily desirable, and it is not clear that a system that achieves high breakeven can be tuned to score high on other effectiveness measures. Yang [1999] also notes that when for no value of the parameters π and ρ are close enough, interpolated breakeven may not be a reliable indicator of effectiveness.

results, in the sense that many of these experiments have been carried out in subtly different conditions.

In general, different sets of experiments may be used for cross-classifier comparison only if the experiments have been performed

(1) on exactly the same collection (i.e. same documents and same categories);
(2) with the same "split" between training set and test set;
(3) with the same evaluation measure and, whenever this measure depends on some parameters (e.g. the utility matrix chosen), with the same parameter values.

Unfortunately, a lot of experimentation, both on Reuters and on other collections, has not been performed with these *caveat* in mind: by testing three different classifiers on five popular versions of Reuters, Yang [1999] has shown that a lack of compliance with these three conditions may make the experimental results hardly comparable among each other. Table 6 lists the results of all experiments known to us 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)[20]. Only experiments that have computed either a breakeven or F_1 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] showed 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 is the restriction of Reuters-21578 "ModApté" to the 10 categories with the highest generality, and is thus an obviously "easier" collection.

Other test collections that have been frequently used 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][21]. The documents are titles or title-plus-abstract's from medical journals (OHSUMED is actually a subset of the Medline document base); the categories are the "postable terms" of the MESH thesaurus.
—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. 2000; Schapire and Singer 2000]. The documents are messages posted to Usenet newsgroups, and the categories are the newsgroups themselves.

[20] 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 do not cover experiments performed on variants of Reuters different from the five listed because the small number of authors that have used 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].
[21] 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 1992a]	.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 (M)	.833 (M)			
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 Reuters. 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 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].

We will not cover the experiments performed on these collections for the same reasons as those illustrated in Footnote 20, i.e. because in no case a significant enough number of authors have used the same collection in the same experimental conditions, thus making comparisons difficult.

7.3 Which text 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, we have to bear in mind that comparisons are reliable only when based on experiments performed by the same author under carefully controlled conditions. They are instead more problematic when they involve different experiments performed by different authors. In this case various "background con-

ditions", often extraneous to the learning algorithm itself, may influence the results. These 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.

Two different methods may thus be applied for comparing classifiers [Yang 1999]:

—*direct comparison*: classifiers Φ' and Φ'' may be compared when they have been tested on the same collection Ω, usually by the same researchers and with the same background conditions. This is the more reliable method.

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

(1) they have been tested on collections Ω' and Ω'', respectively, typically by different researchers and hence with possibly different background conditions;
(2) one or more "baseline" classifiers $\overline{\Phi}_1, \ldots, \overline{\Phi}_m$ have been tested on both Ω' and Ω'' by the direct comparison method.

Test 2 gives an indication on the relative "hardness" of Ω' and Ω''; 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 method is less reliable.

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 $\Omega' > \Omega''$ we indicate that Ω' is a harder collection that Ω'', 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 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 7.2.

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; efficiency considerations or application-dependent issues might play a role in breaking the 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 look the worst of the learning-based classifiers. For Rocchio, these results confirm 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. However, recent results by Schapire et al. [1998] rank Rocchio along the best performers once near-positives are used in training.

—The data in Table 6 are hardly sufficient to say anything about decision trees. However, the work by Dumais et al. [1998] in which a decision tree classifier was shown to perform nearly as well as their top performing system (a SVM classifier) will probably renew the interest in decision trees, an interest that had dwindled

after the 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 classifier implemented by Yang [1999] and not including any learning component[22].

Concerning WORD and no-learning classifiers, for completeness we should recall that one of the highest effectiveness values reported in the literature for the Reuters collection (a .90 breakeven) belongs to CONSTRUE, a manually constructed classifier. However, 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 safe scientific practice would demand. Therefore, the fact that this figure is indicative of the performance of CONSTRUE, and of the manual approach it represents, has been convincingly questioned [Yang 1999].

It is important to bear in mind that the considerations above are not absolute statements (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 similar effectiveness on categories with ≥ 300 positive training examples each. 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 shows that applicative contexts different from Reuters may well invalidate conclusions drawn on this latter.

Finally, a note is worth about statistical significance testing. Few authors have gone to the trouble of validating their 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 Hirsh 1998; Joachims 1997; Koller and Sahami 1997; Lewis et al. 1996; Wiener et al. 1995].

[22] 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 no-learning classifier is proposed in [Wong et al. 1996].

8. CONCLUSION

Automated TC is now a major research area within the information systems discipline, thanks to a number of factors

—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 required by the application make the manual alternative implausible.

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

—It has reached effectiveness levels comparable to those of trained professionals. The effectiveness of manual TC is not 100% anyway [Cleverdon 1984] and, more importantly, it is unlikely to be improved substantially by the progress of research. The levels of effectiveness of automated TC are instead growing at a steady pace, and even if they will likely 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 the effectiveness of text classifiers has dramatically improved, is the arrival in the TC arena of ML 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 weak methods such as Rocchio. In TC, ML researchers have found a challenging application, since datasets consisting of hundreds of thousands of documents and characterized by tens of thousands of terms are widely available. This means that TC is a good benchmark for checking whether a given learning technique can scale up to substantial sizes. In turn, this probably means that the active involvement of the ML community in TC 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 application. Techniques typical of automated TC have already been extended successfully to the categorization of documents expressed in slightly different media; for instance:

—very noisy text resulting from optical character recognition [Ittner et al. 1995; Junker and Hoch 1998]. In their experiments Ittner et al. [1995] have found that, 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.

—speech transcripts [Myers et al. 2000; Schapire and Singer 2000]. For instance, Schapire and Singer [2000] 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 specialized operator according to call type.

Concerning other more radically different media, the situation is not as bright (however, see [Lim 1999] for an interesting attempt at image categorization based on a

textual metaphor). The reason for this is that capturing real semantic content of non-textual media by automatic indexing is still an open 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 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 possible perspectives, from all possible distances, etc. If we had solved the multimedia indexing problem in a satisfactory way, the general methodology that we have discussed in this paper for text would also apply to automated multimedia categorization, 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

This paper owes a lot to the suggestions and constructive criticism of Norbert Fuhr and David Lewis. Thanks also to Umberto Straccia for comments on an earlier draft and to Alessandro Sperduti for many fruitful discussions.

REFERENCES

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

Androutsopoulos, I., Koutsias, J., Chandrinos, K. V., and Spyropoulos, C. D. 2000. An experimental comparison of naive Bayesian and keyword-based anti-spam filtering with personal e-mail messages. In *Proceedings of SIGIR-00, 23rd ACM International Conference on Research and Development in Information Retrieval* (Athens, GR, 2000), pp. 160–167.

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., Di Marco, S., and Salvi, D. 1998. Categorization by context. *Journal of Universal Computer Science 4*, 9, 719–736.

Baker, L. D. and McCallum, A. K. 1998. Distributional clustering of words for text classification. 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., Knorz, G., 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.

Borko, H. and Bernick, M. 1963. Automatic document classification. *Journal of the Association for Computing Machinery 10*, 2, 151–161.

Caropreso, M. F., Matwin, S., and Sebastiani, F. 2001. A learner-independent evaluation of the usefulness of statistical phrases for automated text categorization. In A. G. Chin Ed., *Text Databases and Document Management: Theory and Practice*. Hershey, US: Idea Group Publishing. Forthcoming.

Cavnar, W. B. and Trenkle, J. M. 1994. N-gram-based text categorization. In *Proceedings of SDAIR-94, 3rd Annual Symposium on Document Analysis and Information Retrieval* (Las Vegas, US, 1994), pp. 161–175.

Chakrabarti, S., Dom, B. E., Agrawal, R., and Raghavan, P. 1998a. Scalable feature selection, classification and signature generation for organizing large text databases into hierarchical topic taxonomies. *Journal of Very Large Data Bases 7*, 3, 163–178.

Chakrabarti, S., Dom, B. E., and Indyk, P. 1998b. Enhanced hypertext categorization using hyperlinks. In *Proceedings of SIGMOD-98, ACM International Conference on Management of Data* (Seattle, US, 1998), pp. 307–318.

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 1988], 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*, pp. 124–143. 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. and Hirsh, 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 EMNLP-97, 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.

Denoyer, L., Zaragoza, H., and Gallinari, P. 2001. HMM-based passage models for document classification and ranking. In *Proceedings of ECIR-01, 23rd European Colloquium on Information Retrieval Research* (Darmstadt, DE, 2001).

Díaz Esteban, A., de Buenaga Rodríguez, M., Ureña López, L. A., and García Vega, M. 1998. Integrating linguistic resources in an uniform way for text classification tasks. In *Proceedings of LREC-98, 1st International Conference on Language Resources and Evaluation* (Grenada, ES, 1998), pp. 1197–1204.

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.

Drucker, H., Vapnik, V., and Wu, D. 1999. Automatic text categorization and its applications to text retrieval. *IEEE Transactions on Neural Networks 10*, 5, 1048–1054.

Dumais, S. T. and Chen, H. 2000. Hierarchical classification of Web content. In *Proceedings of SIGIR-00, 23rd ACM International Conference on Research and Development in Information Retrieval* (Athens, GR, 2000), pp. 256–263.

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 (Bethesda, US, 1998), pp. 148–155.

Escudero, G., Màrquez, L., and Rigau, G. 2000. Boosting applied to word sense disambiguation. In *Proceedings of ECML-00, 11th European Conference on Machine Learning* (Barcelona, ES, 2000), pp. 129–141.

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.

Frasconi, P., Soda, G., and Vullo, A. 2001. Text categorization for multi-page documents: A hybrid naive Bayes HMM approach. *Journal of Intelligent Information Systems*. Forthcoming.

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., 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* (Cambridge, UK, 1984), pp. 391–408.

Fuhr, N. and Pfeifer, U. 1994. Probabilistic information retrieval as combination of abstraction inductive learning and probabilistic assumptions. *ACM Transactions on Information Systems 12*, 1, 92–115.

Fürnkranz, J. 1999. Exploiting structural information for text classification on the WWW. In *Proceedings of IDA-99, 3rd Symposium on Intelligent Data Analysis* (Amsterdam, NL, 1999), pp. 487–497.

Galavotti, L., Sebastiani, F., and Simi, M. 2000. Experiments on the use of feature selection and negative evidence in automated text categorization. In *Proceedings of ECDL-00, 4th European Conference on Research and Advanced Technology for Digital Libraries* (Lisbon, PT, 2000), pp. 59–68.

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.

Gövert, N., Lalmas, M., and Fuhr, N. 1999. A probabilistic description-oriented approach for categorising Web documents. In *Proceedings of CIKM-99, 8th ACM International Conference on Information and Knowledge Management* (Kansas City, US, 1999), pp. 475–482.

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.

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.

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.

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., 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.

Iyer, R. D., Lewis, D. D., Schapire, R. E., Singer, Y., and Singhal, A. 2000. Boosting for document routing. In *Proceedings of CIKM-00, 9th ACM International Conference on Information and Knowledge Management* (McLean, US, 2000), pp. 70–77.

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), pp. 200–209.

Joachims, T. and Sebastiani, F. 2001. Guest editors' introduction to the special issue on automated text categorization. *Journal of Intelligent Information Systems*. Forthcoming.

John, G. H., 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.

Junker, M. and Abecker, A. 1997. Exploiting thesaurus knowledge in rule induction for text classification. In *Proceedings of RANLP-97, 2nd International Conference on Recent Advances in Natural Language Processing* (Tzigov Chark, BL, 1997), pp. 202–207.

Junker, M. and Hoch, R. 1998. An experimental evaluation of OCR text representations for learning document classifiers. *International Journal on Document Analysis and Recognition 1*, 2, 116–122.

Kessler, B., Nunberg, G., and Schütze, H. 1997. Automatic detection of text genre. In *Proceedings of ACL-97, 35th Annual Meeting of the Association for Computational Linguistics* (Madrid, ES, 1997), pp. 32–38.

Kim, Y.-H., Hahn, S.-Y., and Zhang, B.-T. 2000. Text filtering by boosting naive Bayes classifiers. In *Proceedings of SIGIR-00, 23rd ACM International Conference on Research and Development in Information Retrieval* (Athens, GR, 2000), pp. 168–75.

Klinkenberg, R. and Joachims, T. 2000. Detecting concept drift with support vector machines. In *Proceedings of ICML-00, 17th International Conference on Machine Learning* (Stanford, US, 2000), pp. 487–494.

Knight, K. 1999. Mining online text. *Communications of the ACM 42*, 11, 58–61.

Knorz, G. 1982. A decision theory approach to optimal automated indexing. In *Proceedings of SIGIR-82, 5th ACM International Conference on Research and Development in Information Retrieval* (Berlin, DE, 1982), pp. 174–193.

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, S. L. AND LEE, D. L. 1999. Feature reduction for neural network based text categorization. In *Proceedings of DASFAA-99, 6th IEEE International Conference on Database Advanced Systems for Advanced Application* (Hsinchu, TW, 1999), pp. 195–202.

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 11*, 6, 865–879.

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. 1992a. 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. 1992b. *Representation and learning in information retrieval*. Ph. D. thesis, Department of Computer Science, University of Massachusetts, Amherst, US.

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.

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.

Manning, C. and Schütze, H. 1999. *Foundations of Statistical Natural Language Processing*. The MIT Press, Cambridge, US.

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

Masand, B. 1994. Optimising confidence of text classification by evolution of symbolic expressions. In K. E. Kinnear Ed., *Advances in genetic programming*, Chapter 21, pp. 459–476. Cambridge, US: The MIT Press.

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.

Merkl, D. 1998. Text classification with self-organizing maps: Some lessons learned. *Neurocomputing 21*, 1/3, 61–77.

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

Mladenić, D. 1998. 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. and Grobelnik, M. 1998. Word sequences as features in text-learning. In *Proceedings of ERK-98, the Seventh Electrotechnical and Computer Science Conference* (Ljubljana, SL, 1998), pp. 145–148.

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.

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), pp. 87–99.

Myers, K., Kearns, M., Singh, S., and Walker, M. A. 2000. A boosting approach to topic spotting on subdialogues. In *Proceedings of ICML-00, 17th International Conference on Machine Learning* (Stanford, US, 2000).

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. 2000. Text classification from labeled and unlabeled documents using EM. *Machine Learning 39*, 2/3, 103–134.

Oh, H.-J., Myaeng, S. H., and Lee, M.-H. 2000. A practical hypertext categorization method using links and incrementally available class information. In *Proceedings of SIGIR-00, 23rd ACM International Conference on Research and Development in Information Retrieval* (Athens, GR, 2000), pp. 264–271.

Pazienza, M. T. Ed. 1997. *Information extraction*. Number 1299 in Lecture Notes in Computer Science. Springer, Heidelberg, DE.

Riloff, E. 1995. Little words can make a big difference for text classification. In *Proceedings of SIGIR-95, 18th ACM International Conference on Research and Development in Information Retrieval* (Seattle, US, 1995), pp. 130–136.

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 1988], 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. 2000. Text-based approaches for non-topical image categorization. *International Journal of Digital Libraries 3*, 3, 261–275.

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 39*, 2/3, 135–168.

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.

Scott, S. and Matwin, S. 1999. Feature engineering for text classification. In *Proceedings of ICML-99, 16th International Conference on Machine Learning* (Bled, SL, 1999), pp. 379–388.

Sebastiani, F., Sperduti, A., and Valdambrini, N. 2000. An improved boosting algorithm and its application to automated text categorization. In *Proceedings of CIKM-00, 9th ACM International Conference on Information and Knowledge Management* (McLean, US, 2000), pp. 78–85.

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.

Singhal, A., Salton, G., Mitra, M., and Buckley, C. 1996. Document length normalization. *Information Processing and Management 32*, 5, 619–633.

Slonim, N. and Tishby, N. 2001. The power of word clusters for text classification. In *Proceedings of ECIR-01, 23rd European Colloquium on Information Retrieval Research* (Darmstadt, DE, 2001).

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.

Tauritz, D. R., Kok, J. N., and Sprinkhuizen-Kuyper, I. G. 2000. Adaptive information filtering using evolutionary computation. *Information Sciences 122*, 2–4, 121–140.

Tumer, K. and Ghosh, J. 1996. Error correlation and error reduction in ensemble classifiers. *Connection Science 8*, 3-4, 385–403.

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.

van Rijsbergen, C. J. 1979. *Information Retrieval* (Second ed.). Butterworths, London, UK. Available at `http://www.dcs.gla.ac.uk/Keith`.

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., Apté, 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. D., 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. 1988. *Document retrieval systems*. Taylor Graham, London, UK.

Wong, J. W., Kan, W.-K., and Young, G. H. 1996. Action: automatic classification for full-text documents. *SIGIR Forum 30*, 1, 26–41.

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.

Yang, Y., Slattery, S., and Ghani, R. 2001. A study of approaches to hypertext categorization. *Journal of Intelligent Information Systems*. Forthcoming.

Yu, K. L. and Lam, W. 1998. A new on-line learning algorithm for adaptive text filtering. In *Proceedings of CIKM-98, 7th ACM International Conference on Information and Knowledge Management* (Bethesda, US, 1998), pp. 156–160.