Machine Learning in Automated Text Categorization

Fabrizio Sebastiani Consiglio Nazionale delle Ricerche, Italy

The automated categorization (or classification) of texts into pre-specified categories, although dating back to the early '60s, 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 the application of machine learning techniques: a general inductive process automatically builds a classifier by learning, from a set of previously classified documents, the characteristics of one or more 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. In this survey we look at the main approaches that have been taken towards automatic text categorization 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 automated content-based document management tasks (collectively known as information retrieval – IR) have gained a prominent status in the information systems field, largely due to the increased availability of documents in digital form and the consequential need on the part of users 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 has a long history, dating back to the early '60s, but it was not until the early '90s that it became a major subfield of the information systems discipline, largely due to increased applicative interest and to the availability of more powerful hardware. Nowadays TC is used in many applicative contexts, ranging from automatic 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. Although commercial TC systems (e.g. [Dörre et al. 1999]) are

Address: Istituto di Elaborazione dell'Informazione, Consiglio Nazionale delle Ricerche, Area della Ricerca di Pisa, Località San Cataldo, 56100 Pisa (Italy). E-mail: fabrizio@iei.pi.cnr.it

not yet as widespread as commercial IR systems, experimental TC systems have achieved high levels of robustness (see e.g. [Lewis et al. 1999] for a description of a sophisticated architecture for TC).

Until the late '80s the most popular approach to TC, at least in the "operational" (i.e. commercial applications) community, was a knowledge engineering (KE) one, and consisted in manually defining a set of rules encoding expert knowledge on how to classify documents under the given categories. However, from the early '90s this approach has increasingly lost popularity, especially in the research community, in favour of the machine learning (ML) approach. In this latter approach a general inductive process automatically builds an automatic text classifier by learning, from a set of previously classified documents, the characteristics of the categories of interest. The advantages of this approach are (i) an accuracy comparable to that achieved by human experts, and (ii) 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 category set. 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 [Dörre et al. 1999; Knight 1999; Pazienza 1997]. There is still considerable debate on where the exact border between these disciplines lies, and the terminology is still evolving. Tentatively, we may observe that "text mining" is increasingly being used to denote all the tasks that, by analysing large quantities of text and detecting usage patterns, try to extract probably useful (although only probably correct) information, and that according to this view TC is an instance of text mining.

TC enjoys quite a rich literature now, but this is still fairly scattered¹. Although two international journals have devoted special issues to this topic [Joachims and Sebastiani 2001; Lewis and Hayes 1994], there are almost no systematic treatments of the subject: there are neither textbooks nor journals entirely devoted to TC yet, and [Manning and Schütze 1999] 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 quite different things 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; Papka and Allan 1998; Roussinov and Chen 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, while in Section 3 we review the most important tasks to which TC has been applied. Section 4 describes the main ideas underlying the ML approach to the automated classification of data items. Our discussion of text classification

¹A 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

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 manually classified documents. Section 7 discusses the evaluation of the indexing techniques and inductive techniques introduced in the previous sections. Section 8 concludes, discussing some of the open issues and possible avenues of further research for TC.

2. TEXT CATEGORIZATION

2.1 A definition of text categorization

Text categorization may be defined as 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 pre-defined 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 $\Phi: \mathcal{D} \times \mathcal{C} \to \{T, F\}$ (that describes how documents ought to be classified) by means of a function $\Phi: \mathcal{D} \times \mathcal{C} \to \{T, F\}$ called the classifier (aka rule, or hypothesis, or model) such that Φ and Φ "coincide as much as possible". How to precisely define and measure this degree of coincidence (that we will call effectiveness) will be discussed in detail in Section 7.1. Throughout the paper we will assume that:

- —The categories are just symbolic labels, and no additional knowledge (either of a procedural or of a declarative nature) of their meaning is available to help in building the classifier.
- —No exogenous knowledge (i.e. data that might be provided for classification purposes by an external source) is available, and the attribution of documents to categories has to be realized solely on the basis of endogenous knowledge (i.e. knowledge that can be extracted from the document itself). This means that only the document text is available, while metadata such as e.g. publication date, document type, publication source, etc. are not.

The effect of these assumptions is that the algorithms that we will discuss are completely general and do not depend on the availability of special-purpose resources that might be costly to develop or might simply be unavailable. 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 [de Buenaga Rodríguez et al. 1997; Díaz Esteban et al. 1998; Junker and Abecker 1997. Relying only on endogenous knowledge basically means trying to classify 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 cannot be decided deterministically. This is exemplified by the well-known phenomenon of inter-indexer inconsistency [Cleverdon 1984; Hamill and Zamora 1980]: when two human experts decide whether to classify document d_i under category c_i , they may disagree, and this in fact happens with relatively high frequency. A news article on the Clinton-Lewinsky case could be filed under Politics, or under Gossip, or under both, or even under neither, depending on the subjective judgment of the classifier. The notion of "membership of a document in

a category" is in many respects similar to the IR notion of "relevance of a document to an information need" [Saracevic 1975].

2.2 Single-label vs. multi-label text categorization

Different constraints may be enforced on the categorization task, depending on the application requirements. For instance, we might need to impose that, for a given integer k, exactly k (or $\leq k$, or $\geq k$) elements of \mathcal{C} must be assigned to each element of \mathcal{D} . The case in which exactly 1 category must be assigned to each document is often called the *single-label* (aka *non-overlapping categories*) case, whereas the case in which any number of categories from 0 to $|\mathcal{C}|$ may be assigned to the same document is dubbed the *multi-label* (aka *overlapping categories*) case. A special case of single-label categorization is *binary* categorization, in which each document d_j 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 case, in the sense that an algorithm for binary classification can also be used for multi-label classification: one needs only transform a problem of multi-label classification under categories $\{c_1, \ldots, c_{|\mathcal{C}|}\}$ into $|\mathcal{C}|$ independent problems of binary classification under categories $\{c_i, \overline{c_i}\}$, for $i=1,\ldots,|\mathcal{C}|$. This requires, however, that categories are stochastically independent of each other, i.e. that for any two categories c',c'' the value of $\Phi(d_j,c')$ does not depend on the value of $\Phi(d_j,c'')$ and viceversa; this is usually assumed to be the case (applicative contexts 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 be dealing with the binary case. There are various reasons for this choice:

- —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 a document is about Golf or not). In TC, most binary classification problems feature unevenly populated categories (i.e. much fewer documents are about Golf than are not) and unevenly characterized categories (e.g. what is about Golf 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 that deal with the more general single-label case, and are simpler to illustrate than these latter.

This ultimately means that we will view the classification problem for $C = \{c_1, \dots, c_{|C|}\}$ as consisting of |C| independent problems of classifying the documents in D un-

der a given category c_i , for $i = 1, ..., |\mathcal{C}|$. A classifier for c_i is then a function $\Phi_i : \mathcal{D} \to \{T, F\}$ that approximates an unknown target function $\check{\Phi}_i : \mathcal{D} \to \{T, F\}$.

2.3 Category-pivoted vs. document-pivoted text categorization

Once we have built a text classifier there are two different ways for using it. Given a document, we might want to find all the categories under which it should be filed (document-pivoted categorization – DPC); alternatively, given a category, we might want to find all the documents that should be filed under it (category-pivoted categorization – CPC). Quite obviously this distinction is more pragmatic than conceptual, but is important in the sense that the sets \mathcal{C} of categories and \mathcal{D} of documents might not always be available in their entirety right from the start. It is also of some relevance to the choice of the method for building the classifier, as some of these methods (e.g. the k-NN method of Section 6.9) allow the construction of classifiers with a definite slant towards one or the other classification style.

DPC is thus suitable when documents become available one at a time over a long span of time, e.g. in filtering e-mail. CPC is instead suitable if it is possible that (i) a new category $c_{|\mathcal{C}|+1}$ is 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 more commonly used than CPC, as the former situation is somehow 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 text categorization process requires a T or F decision for each pair $\langle d_j, c_i \rangle$, as argued in Section 2.1, a partial automation of this process might have different requirements.

For instance, given document d_j a system might simply rank the categories in $C = \{c_1, \ldots, c_{|C|}\}$ according to their estimated appropriateness to d_j , without taking any "hard" decision on either of them. Such a ranked list would be of great help to a human expert in charge of taking the final categorization decision, in that it would be possible for her to restrict the selection of the category (or categories) to the ones at the top of the list rather than having to examine the entire set. Alternatively, given category c_i 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 categorization and document-ranking categorization [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 professional. 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 repre-

sentative 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 be dealing with "hard" classification; however, many of the algorithms we will discuss naturally lend themselves to ranking categorization too (more details on this in Section 6.1).

3. APPLICATIONS OF TEXT CATEGORIZATION

Automatic TC goes back at least to the early '60s, with Maron's [1961] seminal work on probabilistic text classification. Since then, it has been used in a number of different applications. In this section we briefly review the most important ones; note that the borders between the different classes of applications mentioned here are fuzzy and somehow artificial, and some of these applications might arguably be considered special cases of others. Other applications we do not explicitly discuss for reasons of space 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], automatic identification of text genre [Kessler et al. 1997], and (gasp!) automatic essay grading [Larkey 1998].

3.1 Automatic indexing for Boolean information retrieval systems

The first use to which automatic text classifiers were put, and the application that spawned most of the early research in the field [Borko and Bernick 1963; Field 1975; Gray and Harley 1971; Hamill and Zamora 1980; Heaps 1973; Hoyle 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* and 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 the TC task, 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 categorization 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].

The issue of automatic indexing with controlled dictionaries is closely related to the topic of 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.). Usually, some of these metadata are thematic, i.e. their role is to describe the semantics of

the document by means of bibliographic codes, keywords or keyphrases. The generation of these metadata may thus be viewed as a problem of document indexing with controlled dictionary, and thus tackled by means of TC techniques. An example system for automated metadata generation by TC techniques is the KLARITY system (http://www.topic.com.au/products/klarity.html).

3.2 Document organization

Indexing with a controlled vocabulary is one 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 the categories used in the scheme adopted by the newspaper; typical categories might be Personals, Cars for Sale, Real Estate, etc. While most newspapers would handle this application manually, those dealing with a high volume of classified ads might prefer an automatic system to choose the most suitable category for a given ad. In this case a typical constraint is that exactly one category is assigned to each document. Similar applications are the organization of patents into categories for making their search easier [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 dynamic collection of texts, i.e. 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 (e.g. Reuters or Associated Press) and the consumer is a newspaper [Hayes et al. 1990]. In this case the filtering system should block the delivery to the consumer 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 categorization, i.e. the classification of incoming documents in two disjoint categories, the relevant and the irrelevant. Additionally, a filtering system may also perform a further categorization into topical categories of the documents deemed relevant to the consumer; in the example above, all articles about sports are deemed relevant, and should be further classified according e.g. to which sport they deal with, so as to allow individual journalists specialized in individual sports to access only documents of high prospective interest for them. Similarly, an e-mail filter might be trained to 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 [Cohen 1996].

A document filtering system may be installed at the producer end, in which case its role is to route the information to the interested consumers only, or at the consumer end, in which case its role is to block the delivery of information deemed uninteresting to the user. In the former case the system has to build and update a "profile" for each consumer it serves [Liddy et al. 1994], whereas in the latter case (which is the more common, and to which we will refer in the rest of this section) a single profile is needed.

A profile may be initially specified by the user, thereby resembling a standing IR query, and is usually updated by the system by using feedback information provided (either implicitly or explicitly) by the user on the relevance or non-relevance of the delivered messages. In the TREC community [Lewis 1995c; Hull 1998] this is called adaptive filtering, while the case in which no user-specified profile is available is called either routing or batch filtering, depending on whether documents have to be ranked in decreasing order of estimated relevance or just accepted/rejected. Batch filtering thus coincides with single-label categorization under $|\mathcal{C}|=2$ categories; since this latter is a completely general categorization 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 varying degrees of automation and dealing with the multi-consumer case discussed above, it was variously called *selective dissemination* of information or current awareness (see e.g. [Korfhage 1997, Chapter 6]). The explosion in the availability of digital information, particularly on the Internet, has boosted the importance of such systems. These are nowadays being used in many different contexts, including the creation of personalized Web newspapers, junk e-mail blocking, and the selection of Usenet news.

The construction of information filtering systems by means of ML techniques is widely discussed in the literature: see e.g. [Amati and Crestani 1999; Bruckner 1997; Diao et al. 2000; Tauritz et al. 2000; Tong et al. 1992; Yu and Lam 1998].

3.4 Word sense disambiguation

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

WSD may be seen as a categorization task (see e.g [Gale et al. 1993; Hearst 1991]) once we view word occurrence contexts as documents and word senses as categories. Quite obviously this is a single-label categorization case, and one in which document-pivoted categorization is most likely to be 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 instances of this problem, which may all be tackled by means of TC techniques along the lines discussed for WSD, are context-sensitive spelling correction, prepositional phrase attachment, part of speech tagging, and word choice selection in machine translation; see the excellent [Roth 1998] for an introduction.

3.5 Hierarchical categorization of Web pages

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

Automatically classifying Web pages has obvious advantages, since the manual categorization of a large enough subset of the Web is infeasible. Unlike in the previous applications, in this case one would typically want each category to be populated by a set of $k_1 \leq x \leq k_2$ documents, and would choose CPC so as to allow new categories to be added and obsolete ones to be deleted.

With respect to other previously discussed TC applications, the automatic categorization of Web pages has two essential peculiarities:

- (1) The hypertextual nature of the documents: hyperlinks constitute a rich source of information, as they may be understood as statements of 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; Gövert et al. 1999].
- (2) The hierarchical structure of the category set: this may be used e.g. by decomposing the classification problem into a series of smaller classification problems corresponding each 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].

4. THE MACHINE LEARNING APPROACH TO TEXT CATEGORIZATION

In the '80s the approach that was most popular, at least in operational settings, for the creation of automatic document classifiers consisted in their manual construction through *knowledge engineering* (KE) techniques, i.e. in manually building an expert system capable of taking categorization decisions. Such an expert system typically consisted of a set of manually defined rules (one per category) of type

if
$$\langle DNF Boolean formula \rangle$$
 then $\langle category \rangle$ else $\neg \langle category \rangle$

to the effect that the document was classified under $\langle category \rangle$ iff it satisfied $\langle DNF Boolean\ formula \rangle$, DNF standing for "disjunctive normal form". The most famous example of this approach is 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, and its effectiveness as measured on a benchmark selected in [Apté et al. 1994] is reported in Figure 2. Other examples of this approach are [Goodman 1990; Rau and Jacobs 1991].

The drawback of this "manual" approach to the construction of automatic classifiers is the existence of a *knowledge acquisition bottleneck*, similarly to what happens in expert systems. That is, rules must be manually defined by a knowledge engineer with the aid of a domain expert (in this case, an expert in the membership of documents in the chosen set of categories). If the set of categories is updated, then these two trained professionals must intervene again, and if the classifier is ported

wheat & farm	\rightarrow	Wheat
wheat & commodity	\rightarrow	Wheat
bushels & export	\rightarrow	WHEAT
wheat & agriculture	\rightarrow	Wheat
wheat & tonnes	\rightarrow	WHEAT
wheat & winter & \neg soft	\rightarrow	$W_{\rm HEAT}$

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

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

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

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, it has to be remarked that no other classifier has been tested on the same dataset as Construe (see also Table 6), and it is not clear how this dataset was selected from the Reuters collection (i.e. whether it was a random or a favourable subset of the entire collection). As argued in [Yang 1999], the results above do not allow us to confidently say that these effectiveness results may be obtained in general.

Since the early '90s, the ML approach to the construction of text classifiers has gained popularity and 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 that have previously been classified manually under c_i or \overline{c}_i by a domain expert; from these characteristics, the inductive process gleans the characteristics that a novel document should have in order to be classified under c_i . In ML terminology, the classification problem is an activity of supervised learning, since the learning process is driven, or "supervised", by the knowledge of the categories and of the training instances that belong to them². The advantages of this approach over the previous one are evident. The engineering effort goes towards the construction not of a classifier, but of an automatic builder of classifiers (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

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

categories is updated, or if the classifier is ported to a completely different domain.

In the ML approach the manually classified documents are then the key resource. The most favourable case is the one in which they are already available; this is the typical case of an organization that had already been carrying out the same categorization activity manually, and that decides to automate the process. The less favourable case is when no manually classified documents are available; this is typically the case of an organization that starts a categorization activity and decides to opt for an automated modality straightaway. In this case, the ML approach is still more convenient than the KE approach. In fact, it is easier to manually classify a set of documents than to build and tune a set of rules, for the simple reason that it is usually easier to characterize a concept extensionally (i.e. to indicate 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 existence of an initial corpus $\Omega = \{d_1, \ldots, d_{|\Omega|}\}$ of documents previously classified under the same set of categories $\mathcal{C} = \{c_1, \ldots, c_{|\mathcal{C}|}\}$ with which the system will need to operate. This means that the values of the total function $\check{\Phi}: \mathcal{D} \times \mathcal{C} \to \{T, F\}$ are known for every pair $\langle d_j, c_i \rangle \in \Omega \times \mathcal{C}$, where $\Omega \subset \mathcal{D}$. A document d_j is called a positive example of c_i if $\check{\Phi}(d_j, c_i) = T$, a negative example of c_i if $\check{\Phi}(d_j, c_i) = F$.

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

- —a training(-and-validation) set $TV = \{d_1, \ldots, d_{|TV|}\}$. This is the set of documents observing the characteristics of which the classifiers for the various categories are inductively built;
- —a test set $Te = \{d_{|TV|+1}, \ldots, d_{|\Omega|}\}$. This set will be used for the purpose of testing the effectiveness of the classifiers. Each document in Te will be fed to the classifiers, and the classifier decisions $\Phi(d_j, c_i)$ compared with the expert decisions $\check{\Phi}(d_j, c_i)$; a measure of classification effectiveness will be based on how often the $\Phi(d_i, c_i)$ values match the $\check{\Phi}(d_i, c_i)$ values.

Note that the documents in Te cannot participate in any way in the inductive construction of the classifiers; if this condition were not satisfied the experimental results obtained would probably be unrealistically good, 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. This means that the results of the previous evaluation would be a conservative estimation of the real performance, since the final classifier has been trained on more data than the evaluated classifier.

This approach is called the train-and-test approach. An alternative approach is the k-fold cross-validation approach (see e.g. [Mitchell 1996, page 146]), whereby k

different classifiers Φ_1, \ldots, Φ_k are induced by partitioning the initial corpus into k disjoint sets Te_1, \ldots, Te_k , and then iteratively applying the train-and-test approach on pairs $\langle TV_i = \Omega - Te_i, Te_i \rangle$. The final effectiveness figure is obtained by individually computing the effectiveness of the resulting k classifiers Φ_1, \ldots, Φ_k , different among each other because they have been generated from k different training-and-validation sets, and then averaging the individual results in some way.

In both the train-and-test and k-fold cross-validation approaches, it is often the case that in order to optimize the classifier its internal parameters should 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 inductively 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, basically 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; that is, test set and validation set must be kept separate³.

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.:

$$g_{\Omega}(c_i) = \frac{|\{d_j \in \Omega \mid \check{\Phi}(d_j, c_i) = T\}|}{|\Omega|}$$
(1)

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

4.2 Information retrieval techniques and text categorization

The ML approach to classifier construction heavily relies on the basic machinery of IR. The reason is that TC is a content-based document management task, and therefore 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 Step 2, although in a few cases non-standard approaches to Steps 1 and 3 are also used. Indexing, induction and evaluation will be the main themes of Sections 5, 6 and 7, respectively.

 $^{^3}$ 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.

5. DOCUMENT INDEXING AND DIMENSIONALITY REDUCTION

5.1 Document indexing

Text documents 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 considers the meaningful textual units (the problem of lexical semantics) and the meaningful natural language rules for the combination of these units (the problem of compositional semantics). As in IR, a text document d_j is usually represented as a vector of weights $\vec{d}_j = \langle w_{1j}, \ldots, w_{|\mathcal{T}|j} \rangle$, where \mathcal{T} is the set of terms (sometimes also called features) that occur at least once in at least one document of Tr, and $0 \le w_{kj} \le 1$ represents, loosely speaking, how much term t_k contributes to the semantics of document d_j . Differences among the various 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 better effectiveness, thereby confirming similar results from IR [Salton and Buckley 1988].

In particular, a number of authors have tried to use *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 concern 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 highly 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 given 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 particular case, a few authors (e.g. [Apté et al. 1994; Koller and Sahami 1997; Lewis and Ringuette 1994; Li and Jain 1998; Moulinier et al. 1996; Moulinier and Ganascia 1996; Schapire and Singer 2000; Sebastiani et al. 2000; Schütze et al. 1995]) use binary weights, usually due to the symbolic, non-numeric nature of the learning systems they employ; in this case, 1 denotes presence and 0 absence of the term in the document. In the more frequent case of non-binary indexing, for determining the weight w_{kj} of term t_k in document d_j any IR-style indexing technique that represents a document as a vector of weighted terms may be used. Most of the times, the standard tfidf weighting function is used (see e.g. [Salton and Buckley 1988]), defined as

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

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 encodes 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 the term occurs in, the less discriminating it is⁴.

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 to this are the representation techniques used for the Foil system [Cohen 1995a] and for the Sleeping Experts system [Cohen and Singer 1999]).

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

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

Although tfidf is by far the most popular one, other indexing functions have also been used, including probabilistic indexing methods [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].

 $^{^4}$ There exist many variants of tfidf, that differ from each other in terms of logarithms, normalization or other correction factors. Formula 2 is just one of the possible instances of this class; see [Salton and Buckley 1988; Singhal et al. 1996] for variations on this theme.

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])⁵. Concerning stemming (i.e. collapsing 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 (e.g. [Larkey and Croft 1996; Ng et al. 1997; Schütze et al. 1995; Wiener et al. 1995; Yang 1999]), as it reduces both the dimensionality of the term space (see Section 5.3) and the level of stochastic dependence between terms (see Section 6.2).

Depending on the application, either the full text of the document or selected parts of it may be indexed. While the former option is the rule, exceptions exist. For instance, in a patent categorization application Larkey [1999] considers only the title, the abstract, the first twenty 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, it is possible to attribute extra importance to the words appearing therein [Apté et al. 1994; Cohen and Singer 1999; Weiss et al. 1999]. In those applications in which documents are flat the identification of the most relevant part of a document is instead a non-obvious task.

5.2 The Darmstadt Indexing Approach

Within the literature on indexing for TC, a special place is occupied by the AIR/X system [Fuhr et al. 1991]. This system is important since it constitutes the final result of the AIR project, one of the most important efforts in the history of TC. The AIR project, spanning a duration of more than ten years [Knorz 1982; Tzeras and Hartmann 1993], 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]⁶.

The approach to indexing taken in AIR/X is known as the *Darmstadt Indexing Approach* (DIA) [Fuhr 1985]. Here, "indexing" is to be taken 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 makes the DIA unique 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,

⁵One 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)".

⁶The 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.

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_i : e.g. the length of d_i ;
- —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 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*)⁷. 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 fact that the set \mathcal{T} of terms that occur at least once in the corpus Ω is large) may be problematic. In fact, while typical algorithms used in text retrieval (such as cosine matching) are reasonably efficient even for high values of $|\mathcal{T}|$, the same cannot be said of many sophisticated learning algorithms used for classifier induction (e.g. the LLSF algorithm of [Yang and Chute 1994]). Because of this, previous to classifier induction one often applies a pass of dimensionality reduction (DR), whose effect is to reduce the dimensionality 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*, rather than just the *constitutive* characteristics of the training data⁸. Classifiers which overfit the training data tend to be good at re-classifying the data they have been trained on, but much worse at classifying previously unseen data. Experimentation has shown that in order to

 $^{^7}$ Association factors are called *adhesion coefficients* in many early papers on TC; see e.g. [Field 1975; Robertson and Harding 1984].

⁸The overfitting problem is often referred to as "the curse of dimensionality".

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 from consideration one always risks 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 functions 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:

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

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

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 $|\mathcal{T}'| \ll |\mathcal{T}|$, techniques for term selection (also called term space reduction – TSR) attempt to select, from the original set \mathcal{T} terms, the set \mathcal{T}' terms that, when used for document indexing, yield the highest effectiveness. Recent results [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* term selection method, i.e. one in which the term set is identified by means of the same learning method which

will be used for 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 that it is by definition tuned to the learning algorithm being used, and that, if used for local DR, different numbers of terms for different categories may be chosen, depending on whether a category is or is not easily separable from the others. However, this approach surely qualifies as a brute force method, and the number of possible different term sets makes its cost prohibitive for standard TC applications.

A computationally easier alternative to the wrapper approach is the *filtering* approach [John et al. 1994], i.e. keeping the $|\mathcal{T}'| \ll |\mathcal{T}|$ terms that receive the highest score according to a predetermined numerical 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 , which means that only the terms that occur in the highest number of documents are retained. We may also write document frequency in probabilistic terms as $P(t_k)$ once we interpret the event space as the training set. Yang and Pedersen [1997] have shown that, irrespectively of the adopted classifier and of the initial corpus used, by using $\#_{Tr}(t_k)$ for TSR it is possible to reduce the dimensionality of the term space by a factor of 10 with no loss in effectiveness (a reduction by a factor of 100 brings about just a small loss).

This result seems to state that the most valuable terms for TC are those that occur most frequently in the collection. As such, it would seem at first to contradict a truism of IR, according to which the most informative terms are those with low-to-medium document frequency [Salton and Buckley 1988]. But these two results do not contradict each other, since it is well-known (see e.g. [Salton et al. 1975]) that the overwhelming majority of the words that occur 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 word removal needs to be performed before this form of DR is attempted, lest only topic-neutral words are retained [Mladenić 1998].

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

5.4.2 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 19. Probabilities are interpreted as usual on an event space of documents (e.g. $P(\overline{t}_k, c_i)$ thus means the probability that, for a random document x, term t_k does not occur in x and x belongs to category c_i), and are estimated by counting occurrences in the training set. The functions $f(t_k, c_i)$ 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 $f_{avg}(t_k) = \sum_{i=1}^{|\mathcal{C}|} P(c_i) f(t_k, c_i)$, or the maximum $f_{max}(t_k) = \max_{i=1}^{|\mathcal{C}|} f(t_k, c_i)$ of their category-specific values are usually computed.

Most of these functions try to capture the intuition according to which the most valuable terms for categorization under c_i are those that are distributed most differently in the sets of positive and negative examples of c_i . However, interpretations of this basic principle may vary subtly across different functions. For instance, the rationale of using χ^2 is that in the experimental sciences it 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 TC 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 ; as one is interested in the terms which are not, one selects 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 given by the effectiveness that results from applying it to a corpus and then training a classifier on the reduced representation. Various experimental comparisons of TSR functions applied to TC contexts have thus been carried out [Caropreso et al. 2001; Galavotti et al. 2000; Mladenić 1998; Ng et al. 1997; 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 mentioned in Section 5.4.1. 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

⁹The reader should be warned that, for better uniformity, Table 1 views all the TSR functions listed in terms of 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 terms of subjective probability. 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 called like this 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	$IG(t_k, c_i)$	$P(t_k, c_i) \log \frac{P(t_k, c_i)}{P(c_i) \cdot P(t_k)} + P(t_k, c_i) \log \frac{P(\overline{t}_k, c_i)}{P(c_i) \cdot P(\overline{t}_k)}$
Mutual information	$MI(t_k, c_i)$	$\log \frac{P(t_k, c_i)}{P(t_k) \cdot P(c_i)}$
$Chi ext{-}square$	$\chi^2(t_k,c_i)$	$\frac{ Tr \cdot [P(t_k, c_i) \cdot P(\overline{t}_k, \overline{c}_i) - P(t_k, \overline{c}_i) \cdot P(\overline{t}_k, c_i)]^2}{P(t_k) \cdot P(\overline{t}_k) \cdot P(c_i) \cdot P(\overline{c}_i)}$
NGL coefficient	$NGL(t_k, c_i)$	$\frac{\sqrt{ Tr } \cdot [P(t_k, c_i) \cdot P(\overline{t}_k, \overline{c}_i) - P(t_k, \overline{c}_i) \cdot P(\overline{t}_k, c_i)]}{\sqrt{P(t_k) \cdot P(\overline{t}_k) \cdot P(c_i) \cdot P(\overline{c}_i)}}$
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.

indicate that $\{OR_{sum}, NGL_{sum}, GSS_{max}\} > \{\chi^2_{max}, IG_{sum}\} > \{\#_{avg}, \chi^2_{avg}\} \gg \{MI_{max}, MI_{avg}\}$, 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 $|T'| \ll |T|$, term extraction (aka reparameterization) attempts to synthesize, from the original set T, a set T' of new terms that maximizes 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 aim at solving these problems by creating artificial terms that do not suffer from the above-mentioned problems. Any term extraction method must specify (i) a method for extracting the new terms from the old ones, and (ii) a method for converting the original document representation 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 aims at grouping words with a high degree of pairwise semantic relatedness into clusters, so that the clusters (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 terms¹⁰.

Lewis [1992a] was the first to investigate the use of term clustering in TC. The method he employs, 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."

Another example of this approach is the work of Li and Jain [1998], who view semantic relatedness between words in terms of their co-occurrence and co-absence within training documents. By using this technique in the context of a hierarchical clustering algorithm they witnessed only a marginal effectiveness improvement; however, the small size of their experiment (see Section 6.11) hardly allows any definitive conclusion to be reached.

The works of both [Lewis 1992a; Li and Jain 1998] are examples of unsupervised clustering, since the clustering activity is not affected by the category labels attached to the documents. Baker and McCallum [1998] provide instead an example of supervised clustering, as the distributional clustering method they employ clusters together those terms that tend to indicate the presence of the same category, or group of categories. Their experiments, carried out in the context of a Naïve Bayes classifier (see Section 6.2), showed only a 2% effectiveness loss with and aggressivity of 1000, and even showed some effectiveness improvement with less aggressive levels of reduction.

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

One characteristic of LSI as a DR function is that the newly obtained dimensions are not, unlike the cases of term selection and term clustering, intuitively interpretable. However, they tend to work well in bringing out the "latent" semantic structure of the vocabulary used in the corpus. For instance, Schütze et al. [1995,

 $^{^{10}}$ Some term selection methods, such as wrapper methods, also address the problem of redundancy.

page 235] discuss the case of classification under category Demographic shifts in the U.S. with economic impact by a neural network classifier. In the experiment they report, they discuss the case of a document that was indeed a positive test instance for the category, and that contained, among others, the quite revealing sentence "The nation grew to 249.6 million people in the 1980s as more Americans left the industrial and agricultural heartlands for the South and West". The classifier decision was incorrect when local 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 LSI representations specific to individual categories, and (ii) for global DR, by creating a single LSI representation for the entire category set. Their experimental results show the former approach to perform better than the latter. Anyway, both LSI-based approaches are shown to perform better than a simple term selection technique based on the Relevancy Score measure (see Table 1).

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

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

6. INDUCTIVE CONSTRUCTION OF TEXT CLASSIFIERS

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

We start by discussing the general form that a text classifier takes. 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 of the definition of a function $CSV_i : \mathcal{D} \to [0,1]$ 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 should be classified under c_i . Documents are then ranked according to their CSV_i value. This works for "document-ranking categorization"; "category-ranking categorization" 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 classifier-induction 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 of the definition of a function $CSV_i: \mathcal{D} \to \{T, F\}$. The latter consists instead of the definition of a function $CSV_i: \mathcal{D} \to [0, 1]$, 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 T^{11} .

The definition of a threshold 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 proposed 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. In this phase the presentation of the algorithms will be mostly qualitative rather than quantitative, i.e. will focus on the methods for classifier induction rather than on the performance (effectiveness and efficiency) of the classifiers that can be induced by means of them. This latter issue will instead be the topic of Section 7.

6.1 Determining thresholds

There are various possible 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 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 yields the best value of the chosen effectiveness function. We will 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 thresholds τ_i are typically chosen for the different categories c_i .

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 quite sensible principle according to which the same percentage of documents of both training and test set should be classified under c_i . One drawback of this thresholding policy is that, for

 $^{^{11}}$ 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.

obvious reasons, it does not lend itself to document-pivoted categorization.

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 categorization. It suffers, however, from a certain coarseness, as the fact that k is equal for all documents (nor could this be otherwise) allows no fine-tuning.

In terms of experimental results, Lewis [1992a] found the proportional policy to be definitely 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. Of course, 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 falls within a category c_i , and attempt to compute this probability through an application of Bayes' theorem, given by

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

In Equation 4 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 Equation 4 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

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

Probabilistic classifiers that make use of this assumption are usually called *Naïve Bayes* classifiers, and account for most of the probabilistic approaches to TC 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 Equation 5 may be written as

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

We may further observe that in TC the document space is partitioned into two categories¹², c_i and its complement \overline{c}_i , such that $P(\overline{c}_i|d_j) = 1 - P(c_i|d_j)$. If we plug in Equations 5 and 6 into Equation 4 and take logs we obtain

$$\log P(c_{i}|\vec{d_{j}}) = \log P(c_{i}) +$$

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

$$\log(1 - P(c_{i}|\vec{d_{j}})) = \log(1 - P(c_{i})) +$$

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

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

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

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 the factor $\log \frac{P(c_i)}{1-P(c_i)}$ and the factor $\sum_{k=1}^{|\mathcal{T}|} \log \frac{1-p_{ki}}{1-p_{k\overline{i}}}$ are constant for all documents, and may therefore be disregarded¹³. Defining a classifier for category c_i thus basically requires estimating the $2|\mathcal{T}|$ parameters $\{p_{1i}, p_{1\overline{i}}, \dots, p_{|\mathcal{T}|i}, p_{|\mathcal{T}|\overline{i}}\}$ from the training data, which may be done in the

¹²Cooper [1995] has pointed out that in this case the full independence assumption of Equation 5 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 $\frac{P(\vec{d_j}|c_i)}{P(\vec{d_j}|\overline{c_i})} = \prod_{k=1}^{|T|} \frac{P(w_{kj}|c_i)}{P(w_{kj}|\overline{c_i})}.$

¹³This is not true, however, if the "fixed thresholding" method of Section 6.1 is adopted. In fact, in this case the k categories most appropriate for document d_j are chosen. This means that for a fixed document d_j the first and third factor in the formula above are different for different categories, and may therefore influence the choice of the categories under which to file d_j .

obvious way. As a matter of fact, probabilistic classifiers are also called parametric classifiers, exactly because their inductive construction consists in estimating probabilistic parameters from the training data. Note that in general the classification of a given document does not require to compute a sum of $|\mathcal{T}|$ 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 those factors for which $w_{kj} = 0$ may be disregarded, and this usually accounts for the vast majority of them, since document vectors are usually very sparse.

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

- —to relax the constraint that document vectors should be binary-valued. This looks quite natural, given that weighted indexing techniques (see e.g. [Fuhr 1989; Fuhr and Buckley 1991; Salton and Buckley 1988]) that account for the "importance" that a term t_k has to a document d_j play a key role in IR.
- —to introduce document length normalization. In fact, 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 . Taking document 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 Assumption 5.
- —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 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 induction methods are essentially quantitative (i.e. numeric) in nature, and as such have sometimes been criticized since, effective as they may be, are not

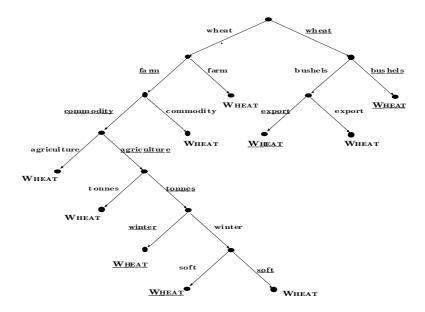


Fig. 3. A decision tree equivalent to the DNF rule of Figure 1. Edges are labelled by terms and leaves are labelled by categories (underlining denotes negation).

readily 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 inducers 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 leaf nodes 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 of such a tree is illustrated in Figure 3.

There are a number of standard packages for DT induction, 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 procedure for the induction of a DT for category c_i consists in a "divide and conquer" strategy of recursively (i) checking whether all the training examples have the same label (either c_i or $\overline{c_i}$); (ii) if not, selecting a term t_k , partitioning 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 excessively specific to the training data. Most DT induction 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 tree induction abound; the interested reader is referred to [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 et al. 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 disjunctive normal form (DNF) rule, i.e. of a conjunction of conditional formulae ("clauses") of the type illustrated in Figure 1¹⁴. Clause premises denote the presence or absence of terms in the test document, while the clause head denotes the decision whether to classify it or not under c_i . DNF rules are similar to decision trees in that they can encode any Boolean function. However, one of the advantages of DNF rule inducers is that they tend to generate more compact classifiers than DT inducers.

Rule induction methods usually attempt to select from all the possible covering rules (i.e. those rules that correctly classify all the training examples) the "best" one according to some minimality criterion. While DTs are typically induced by a top-down, "divide-and-conquer" strategy, DNF rules are often induced in a bottom-up fashion. At the start of the induction of the classifier for c_i , every training example is viewed as a clause $\eta_1, \ldots, \eta_n \to \gamma_i$, where η_1, \ldots, η_n are the terms contained in the document and γ_i equals c_i or \bar{c}_i according to whether the document is a positive or negative example of c_i . This set of clauses is already a DNF classifier for c_i , but obviously scores high in terms of overfitting. The induction algorithm employs 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 inductive DNF rule learners that have been applied to TC are Charade [Moulinier and Ganascia 1996], DL-ESC [Li and Yamanishi 1999], RIPPER [Cohen 1995a; Cohen and Hirsh 1998; Cohen and Singer 1999], SCAR [Moulinier et al. 1996], and SWAP-1 [Apté et al. 1994].

While the DNF induction methods mentioned above deal with rules at the propo-

¹⁴Many 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.

sitional logic level, research has also been carried out on using rules of first order logic, obtainable through the use of *inductive logic programming* methods. Cohen [1995a] has extensively compared propositional and first order induction in a TC application (for instance, comparing the propositional learner RIPPER with its first order version FLIPPER), and has found that the additional representational power of first order logic 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]). In the statistical learning community regression refers to the problem of approximating a real-valued function Φ 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) proposed by Yang and Chute [1994]. In LLSF, each document d_j has two vectors associated to it: an input vector $I(d_j)$, i.e. a standard vector of $|\mathcal{T}|$ weighted terms, and an output vector $O(d_j)$, consisting of a vector of $|\mathcal{C}|$ weights representing the categories (the weights for this latter vector are binary for training documents, and are standard 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, $\|V\|_F \stackrel{def}{=} \sum_{i=1}^{|\mathcal{C}|} \sum_{j=1}^{|\mathcal{T}|} v_{ij}$ 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 .

Experimentation has shown 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 representation of c_i in terms of 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 Equation 3), the dot product between the two vectors

corresponds to their cosine similarity, i.e.

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

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 induced, 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 the induction of 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 various categories [Blosseville et al. 1992; 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 those applications (e.g. semi-automated classification or 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 example of an on-line method is the *perceptron* algorithm, first proposed for TC applications in [Schütze et al. 1995; Wiener et al. 1995] and subsequently 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_i (represented by a vector \vec{d}_i of binary weights) is examined, the classifier built so far attempts to classify it, and the result of the classification is examined. If this is correct nothing is done, while if this is wrong the weights of the classifier are modified: if d_i 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_i was a negative example of c_i then the same weights are demoted, by decreasing them by α . Note that when at a certain stage of the training phase the classifier has reached a reasonable level of effectiveness, the fact that a weight w_{ki} is very low means that t_k has negatively contributed to the classification process so far, and may thus be discarded from the representation. We may then see the perceptron algorithm (as all other incremental induction methods, for that matter) as allowing for a sort of "on-the-fly term space reduction" [Dagan et al. 1997, Section 4.4]. The perceptron classifier has shown a good effectiveness in all the experiments quoted above.

The perceptron is an additive weight-updating algorithm. A multiplicative vari-

ant 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; Ragas and Koster 1998] is a further variant of Positive Winnow, in which the classifier consists of two weights w_{ki}^+ and w_{ki}^- for each term t_k ; the final weight w_{ki} used in computing the inner product is the difference $w_{ki}^+ - w_{ki}^-$. Following the misclassification of a positive instance, active terms have their w_{ki}^+ weight promoted and their w_{ki}^- weight demoted, whereas in the case of a negative instance it is w_{ki}^+ that gets promoted while w_{ki}^- gets demoted (for the rest, promotions and demotions are as in Positive Winnow). Balanced Winnow allows negative w_{ki} weights, while in the perceptron algorithm and in Positive Winnow the w_{ki} weights are always positive. In experiments conducted by Dagan et al. [1997], Positive Winnow showed a better effectiveness than the perceptron algorithm but was in turn outperformed by (Dagan et al.'s own version of) Balanced Winnow.

Other examples of 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 the Sleeping Experts algorithm [Cohen and Singer 1999; Ragas and Koster 1998], a version of Balanced Winnow. While the first is an additive weight-updating algorithm, the second and third are multiplicative. Key differences with the previously described algorithms are that these three algorithms (i) update the classifier not only after misclassifying a training example, but also after classifying it correctly, and (ii) update the weights corresponding to all terms (instead of just active ones).

Linear classifiers lend themselves to both category-pivoted and document-pivoted TC. 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 $\{\vec{c_1}, \ldots, \vec{c_{|C|}}\}$.

6.7 The Rocchio method

Some linear classifiers are *profile-based*, as they rely on the extraction of an explicit profile (or prototypical document) of the category from the training set. This has obvious advantages in terms of interpretability, as such a profile is a representation more readily interpretable by a human than, say, a neural network classifier. The induction of a linear classifier is often preceded by local term space reduction; in this case, a profile of c_i is a weighted list of the terms whose presence or absence is most useful for discriminating the category.

The Rocchio method is used for inducing linear, profile-based classifiers. It relies on an adaptation to TC of Rocchio's formula for relevance feedback in the vector-space model, and it is perhaps the only TC method whose roots lie exclusively in the IR tradition rather than in the ML one. This adaptation was first proposed by Hull [1994]; since then, the Rocchio method has been used by many authors, either as an object of research in its own right [Ittner et al. 1995; Joachims 1997; Ragas and Koster 1998; Sable and Hatzivassiloglou 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 $\vec{c}_i = \langle w_{1i}, \dots, w_{|\mathcal{T}|i} \rangle$ 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 , $POS_i = \{d_j \in Tr \mid \check{\Phi}(d_j, c_i) = T\}$ and $NEG_i = \{d_j \in Tr \mid \check{\Phi}(d_j, c_i) = F\}$. 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; Joachims 1998; Schütze et al. 1995]), the profile of c_i is the centroid of its positive training examples. In general, the Rocchio classifier rewards the closeness of a test document to the centroid of the positive training examples, and its distance from the centroid of the negative training examples. 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 it basically comes down to averaging weights. In terms of effectiveness, instead, one of its drawbacks is that if the documents in the category tend to occur in disjoint clusters (e.g. a set of newspaper articles falling under the Sports category and dealing with either boxing or rock-climbing), the Rocchio classifier may miss most of them, as the centroid of these documents may fall well outside all of these clusters (see Figure 4a). More generally, the Rocchio classifier, as all linear classifiers, has the disadvantage that it divides the space of documents linearly. This situation is graphically depicted in Figure 4a, where documents are classified within c_i if and only if they fall within the circle. Note that even most of the positive training examples would not be classified correctly by the classifier. This is due to the fact that Rocchio basically computes averages (i.e. centroid), and every average is only partly representative of the entire set.

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, 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 relevant documents. Using near-positives corresponds to the *query zoning* method proposed for IR by Singhal et al. [1997]. This method originates from the observation that when the original Rocchio formula is used for relevance feedback in IR, near-positives tend to be used rather than generic negatives, as the documents on which user judgments are available tend to be the ones that had scored highest in the previous ranking. Early applications of the Rocchio formula to TC (e.g. [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

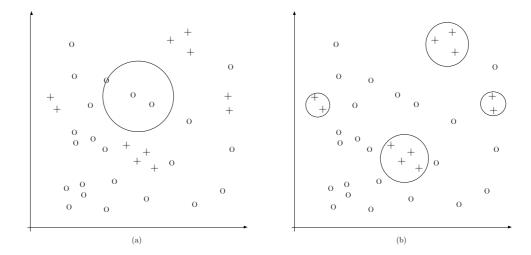


Fig. 4. A comparison between the categorization behaviour of (a) the Rocchio classifier, and (b) the k-NN classifier. Small crosses and circles denote positive and negative training instances, respectively. The big circles denote the "influence area" of the classifier. Note that, for ease of illustration, document similarities are here viewed in terms of Euclidean distance rather than, as more common, in terms of dot product or cosine.

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 can then be used as near-positives. Ng et al. [1997] 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 (Web page categorization into hierarchical catalogues) the notion of a "sibling category of c_i " is well-defined. A similar policy is also adopted in [Ruiz and Srinivasan 1999].

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

parameters of the network and eliminate or minimize the error.

The simplest type of neural network classifier is the perceptron [Dagan et al. 1997; Ng et al. 1997], which is a linear classifier and as such has been extensively discussed in Section 6.6. Other types of linear neural network classifiers implementing a form of logistic regression have also been proposed and tested by Schütze et al. [1995] and Wiener et al. [1995], and they have usually given very good effectiveness results.

A non-linear neural network [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 neural networks to their linear counterparts have been performed, the former have yielded either no improvement [Schütze et al. 1995] or very small improvements [Wiener et al. 1995] over the latter.

6.9 Example-based classifiers

Example-based classifiers do not build an explicit, declarative representation of the category of interest, 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 introduction of example-based methods (aka memory-based reasoning methods) in the TC literature is due to Creecy, Masand and their colleagues [Creecy et al. 1992; Masand et al. 1992]; other 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 implemented by Yang [1994] in the ExpNet system. For deciding whether d_j should be classified under c_i , k-NN looks at whether the k training documents most similar to d_j have also been classified under c_i ; if the answer is positive for a large enough proportion of them, a positive categorization decision is taken, and a negative decision is taken otherwise.

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

$$CSV_i(d_j) = \sum_{d_z \in Tr_k(d_j)} RSV(d_j, d_z) \cdot \llbracket \check{\Phi}(d_z, c_i) \rrbracket$$
 (10)

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

$$[\![\alpha]\!] = \left\{ \begin{array}{l} 1 \text{ if } \alpha = T \\ 0 \text{ if } \alpha = F \end{array} \right.$$

The thresholding methods of Section 6.1 can then be used in order to convert the real-valued CSV_i values into binary categorization decisions. In Equation 10, $RSV(d_j, d_z)$ 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 a threshold k that indicates how many top-ranked training documents have to be considered for computing $CSV_i(d_j)$; k is usually determined experimentally on a validation set. For instance, Larkey and Croft [1996] use k = 20, while Yang [1994, 1999] has found $30 \le k \le 45$ to yield the best effectiveness. Anyhow, various experiments have shown that increasing the value of k does not significantly degrade the performance.

Note that k-NN, unlike linear classifiers, does not subdivide the document space linearly, hence it does not suffer from the problem discussed at the end of Section 6.7. This is graphically depicted in Figure 4b, where the more "local" character of k-NN with respect to Rocchio can be appreciated.

It is quite evident that k-NN is naturally geared towards document-pivoted categorization, since ranking the training documents in terms of their similarity with the test document can be done once for all categories. For category-pivoted categorization 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 characteristic 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 nearest neighbour techniques have been used in the TC literature.

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

$$CSV_i(d_j) = 1 - \prod_{d_z \in Tr_k(d_j)} (1 - RSV(d_j, d_z))^{\llbracket \check{\Phi}(d_z, c_i) \rrbracket}$$

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

A variant of the basic k-NN approach is proposed by Galavotti et al. [2000], who reinterpret Equation 10 by redefining $[\![\alpha]\!]$ as

$$\llbracket \alpha \rrbracket = \left\{ \begin{array}{c} 1 \text{ if } \alpha = T \\ -1 \text{ if } \alpha = F \end{array} \right.$$

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

$$CSV_{i}(d_{j}) \stackrel{def}{=} \sum_{k_{iz} \in K_{i}} RSV(d_{j}, G(k_{iz})) \cdot \frac{|\{d_{j} \in k_{iz}| \check{\Phi}(d_{j}, c_{i}) = T\}|}{|\{d_{j} \in k_{iz}\}|} \cdot \frac{|\{d_{j} \in k_{iz}\}|}{|Tr|}$$

$$= \sum_{k_{iz} \in K_{i}} RSV(d_{j}, G(k_{iz})) \cdot \frac{|\{d_{j} \in k_{iz}| \check{\Phi}(d_{j}, c_{i}) = T\}|}{|Tr|}$$
(11)

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

This exploits the superior effectiveness (see Figure 4) of k-NN over linear classifiers while at the same time avoiding the sensitivity of k-NN to the presence of "outliers" (i.e. positive instances of c_i that "lie out" of the region of document space where most other positive instances of c_i are located) in the training 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 |T|-dimensional space that separate the positive from the negative training examples (decision surfaces), the surface 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 5, 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 positive and the negative examples are not linearly separable. Yang and Liu [1999] experimentally compared the linear case (namely, when the assumption is made that the categories are linearly separable) with the non-linear case on a standard benchmark, and obtained slightly better results in the former case.

According to Joachims [1998], 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;

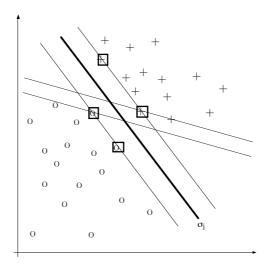


Fig. 5. The induction of support vector classifiers. The small crosses and circles represent positive and negative training examples, respectively, whereas lines represent decision surfaces. Decision surface σ_i (indicated by the thicker line) is, among those shown, the best possible one, as it is the middle element of the widest set of parallel decision surfaces (i.e. its minimum distance to any training example is maximum). Small boxes indicate the support vectors.

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

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

6.11 Classifier committees

The method of classifier committees (aka ensembles) is based on the idea that, given a task that requires expert knowledge to be performed, k experts may be better than one if their individual judgments are appropriately combined. In TC, the idea is to apply k different classifiers Φ_1, \ldots, Φ_k to the same task of deciding whether document d_j should be classified under category c_i , and then combine their outcome appropriately. Such a classifier committee is then 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 be different in terms of the indexing approach used, or in terms of 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]).

Different combination rules have been tested in the literature. The simplest possible rule is $majority\ voting\ (MV)$, whereby the binary classification judgments obtained by the k classifiers are pooled together, and the classification decision

that reaches the majority of $\frac{k+1}{2}$ votes is taken (k obviously needs to be an odd number) [Li and Jain 1998; Liere and Tadepalli 1997]. This method is particularly suited to the case in which the committee includes classifiers characterized by a binary decision function $CSV_i: \mathcal{D} \to \{T, F\}$. A second rule is weighted linear combination (WLC), whereby a weighted sum of the CSV_i 's individually produced by the k classifiers yields the final CSV_i . The weighst w_j are meant to reflect the expected relative effectiveness of classifiers Φ_j , and are such that $\sum_{j=1}^k w_j = 1$. Typically, these weights are optimized on a validation set [Larkey and Croft 1996]. Another possible policy is dynamic classifier selection (DCS), whereby among committee $\{\Phi_1, \ldots, \Phi_k\}$ the classifier Φ_t that yields the best effectiveness on the l validation examples most similar to d_i is selected, and its judgment adopted by the committee [Li and Jain 1998]. A still different policy, somehow intermediate between WLC and DCS, is adaptive classifier combination (ACC), whereby the judgments of all the classifiers in the committee are summed together, but their individual contribution is weighted by the effectiveness that they have shown on the l validation examples most similar to d_i [Li and Jain 1998].

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

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

6.11.1 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 Φ_1, \ldots, Φ_k forming the committee are obtained not by means of k different learning methods, but by the same learning method (here called the weak learner). For instance, if a decision tree classifier is used as the weak learner, the resulting committee will be formed by k decision tree classifiers. The key intuition of boosting is that the k classifiers should be trained not in a conceptually parallel and independent way, as in the classifier committees described above, but sequen-

tially. In this way, the training of classifier Φ_i may take into account how classifiers $\Phi_1, \ldots, \Phi_{i-1}$ perform on the training examples, and concentrate on getting right those examples on which $\Phi_1, \ldots, \Phi_{i-1}$ have performed worst.

Specifically, for the induction of classifier Φ_t each $\langle d_j, c_i \rangle$ pair is attributed an "importance weight" h_{ij}^t (where h_{ij}^1 is set to be equal for all $\langle d_j, c_i \rangle$ pairs¹⁵), meant to represent how hard to get a correct decision for this pair was for classifiers $\Phi_1, \ldots, \Phi_{t-1}$. These weights are exploited in the induction of classifier Φ_t , which will be specially tuned to solve correctly the pairs with higher importance weight. Classifier Φ_t is then applied to the training documents, and as a result weights h_{ij}^t are updated to h_{ij}^{t+1} ; in this update operation, pairs correctly classified by Φ_t will have their importance weight decreased, while pairs misclassified by Φ_t will have their weight increased. After all the k classifiers have been constructed, a weighted linear combination rule is applied to yield the final committee.

In the BOOSTEXTER system [Schapire and Singer 2000], two different boosting algorithms are provided and 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' [1997] PRTFIDF classifier. An algorithm that improves on the effectiveness and (especially) the efficiency of ADABOOST.MH based on a "committee of classifier sub-committees" is presented in [Sebastiani et al. 2000].

An approach similar to boosting is employed by Weiss et al. [1999]. In this work the authors experiment with committees of decision trees each having an average of 16 leaves (hence much more complex than the simple 2-leaves trees used in [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. The authors have found that this approach yields excellent effectiveness gains over the individual decision tree case (and excellent effectiveness tout court), with gains rising briskly until the 10 trees case and reaching a plateau at about 100 trees. Boosting-based approaches have also been employed in [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 inductive approaches proposed in the TC literature, it would be hardly possible to be exhaustive. The recent explosion of this discipline has brought about a fragmentation in terms of the learning approaches adopted, some of which either do not fall squarely under one or the other class of algorithms,

 $^{^{15}}$ 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).

or have remained somehow isolated attempts. Although for reasons of space we will not discuss them in detail, we want to mention the existence of approaches 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 in the case of IR systems, the evaluation of document classifiers is typically conducted *experimentally*, rather than analytically. The reason for this 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 most often measured in terms of the classic IR notions of precision (π) and recall (ρ) , adapted to the case of document categorization. Precision wrt c_i (π_i) is defined as the conditional probability $P(\check{\Phi}(d_x, c_i) = T \mid \Phi(d_x, c_i) = T)$, 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 $P(\Phi(d_x, c_i) = T \mid \check{\Phi}(d_x, c_i) = T)$, 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 the given category set \mathcal{C} , while ρ may be viewed as its "degree of completeness" wrt \mathcal{C} .

As they are defined here, π_i and ρ_i (and consequently π and ρ) are to be understood, in the line of [Wong and Yao 1995], as *subjective* probabilities, i.e. values measuring the expectation of the user that the system will behave correctly when classifying a random document under c_i . These probabilities may be estimated in terms of the *contingency table* for category c_i on a given test set (see Table 2). Here, FP_i (false positives wrt c_i , aka errors of commission) is the number of documents of the test set that have been incorrectly classified under c_i ; TN_i (true negatives wrt c_i), TP_i (true positives wrt c_i) and FN_i (false negatives wrt c_i , 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

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

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

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

Catego	ry	expert judgments		
c_i		YES	NO	
classifier	YES	TP_i	FP_i	
judgments	NO	FN_i	TN_i	

Table 2. The contingency table for category c_i .

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

Table 3. The global contingency table.

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

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

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

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

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

$$\hat{\pi}^{M} = \frac{\sum_{i=1}^{|\mathcal{C}|} \hat{\pi}_{i}}{|\mathcal{C}|}$$

$$\hat{\rho}^{M} = \frac{\sum_{i=1}^{|\mathcal{C}|} \hat{\rho}_{i}}{|\mathcal{C}|}$$

where the "M" superscript stands for macroaveraging.

It is important to recognize that 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 highlighted by macroaveraging and much less so by microaveraging. There is no complete agreement among authors on which is better. Some believe that "microaveraged performance is somewhat misleading (...) because more frequent topics are weighted heavier in the average" [Wiener et al. 1995, page 327] and thus favour macroaveraging. Others (actually, the majority of researchers) believe that topics should indeed count proportionally to their frequence, and thus lean towards microaveraging. We believe

that whether one or the other should be chosen depends on the application requirements. From now on, we will assume that microaveraging is used, and will thus drop the "i" subscript; 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. Other 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] noted, the typically large value that their denominator takes in TC makes them much more insensitive to a variation in the number of correct decisions (TP+TN) than π and ρ are. Besides, [Yang 1999] shows that taking A as the evaluation measure, in the (frequent) case of a low average generality, the trivial rejector (i.e. the classifier Φ such that $\Phi(d_j, c_i) = F$ for all documents d_j and categories c_i) tends to outperform all non-trivial classifiers (see also [Cohen 1995a, Section 2.3]). A consequence of adopting A is also that parameter tuning for a non-trivial classifier on a validation set results 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 precision and recall not on "absolute" values of success $(\Phi(d_j, c_i) = \check{\Phi}(d_j, c_i))$ and failure $(\Phi(d_j, c_i) \neq \check{\Phi}(d_j, c_i))$, but on values of relative success, given by

$$CSV_i(d_j)$$
 if $\check{\Phi}(d_j, c_i) = T$
 $1 - CSV_i(d_j)$ if $\check{\Phi}(d_j, c_i) = F$

This amounts to saying that for a correct (resp. wrong) decision the classifier should be 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 (fully automatic) classification systems. However, it might be appropriate for interactive classifiers of the type used in [Larkey 1999], where the confidence that the classifier has in its own decision influences category ranking and, as a consequence, the overall usefulness of the system.

7.1.3 Measures alternative to effectiveness. In general, criteria different from effectiveness are seldom used in classifier evaluation. For instance, the efficiency criterion, although very important for applicative purposes, is seldom used as the sole yardstick on which to evaluate or compare classifiers, due to the volatility of the parameters on which the evaluation rests. However, efficiency may be a useful criterion for choosing among classifiers with similar effectiveness. On this respect, an interesting evaluation has been carried out by Dumais et al. [1998], who have compared five different induction 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). We refer the interested reader to [Dumais et al. 1998] for details.

One important alternative to effectiveness is *utility*, a class of measures well-known in decision theory that extend effectiveness by economic criteria such as *gain* or *loss*. Utility is based on defining a *utility matrix* such as that of Table 4, where the

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

Table 4. The utility matrix.

numeric values u_{TP} , u_{FP} , u_{FN} and u_{TN} represent the economic gain brought about by a true positive, false positive, false negative and true negative, respectively; both u_{TP} and u_{TN} are greater than both u_{FP} and u_{FN} . "Standard" effectiveness is a particular case of utility, i.e. the one in which $u_{TP} = u_{TN} > u_{FP} = u_{FN}$. Less trivial cases of utility are those in which $u_{TP} \neq u_{TN}$ and/or $u_{FP} \neq u_{FN}$; 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 a legitimate message (FN).

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 on a validation set (as discussed in Section 6.1). Specifically, as Lewis [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 is being more and more used within the text filtering community, and the TREC "filtering track" evaluations have recently adopted it [Lewis 1995c; Hull 1998]. 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 in detail these other measures and their relationships with the more standard ones. This only points to the fact 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 that sets $\Phi(d_j, c_i) = T$ for all documents d_j and categories c_i (the trivial acceptor) has $\rho = 1.0$. 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. Quite obviously, in this case precision would usually be very low

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

Table 5. Trivial cases in text categorization.

(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 precision may be obtained at the price of a low recall.

In practice, by tuning thresholds τ_i a function $CSV_i: \mathcal{D} \to \{T, F\}$ 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 measured by means of a "combined" effectiveness measure which is a function of both π and ρ . Various such measures have been proposed, among which the following are most frequent:

- (1) effectiveness is computed as (interpolated) 11-point average precision. That is, each threshold τ_i is successively set to the values for which ρ_i takes up values of 0.0, 0.1, ..., 0.9, 1.0; for these 11 different thresholds π_i is computed and averaged over the 11 resulting values. This is completely analogous to the standard evaluation methodology for ranked IR systems, and may be used
 - (a) with categories used 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 used in place of IR queries and categories in place of

¹⁶From what we said about the trivial acceptor, one might be tempted to infer by symmetry that the trivial rejector has 100% precision. This is not true, as precision is undefined (the denominator is zero) in the case of the trivial rejector (see Table 5). In fact, it is also clear from its definition $(\pi = \frac{TP}{TP+FP})$ that precision depends only on how the positives (TP+FP) are split between true positives TP and the false positives FP, and does not depend at all on the cardinality of the positives. There is a breakup of "symmetry" between precision and recall here because, from the point of view of classifier judgment (positives vs. negatives; this is the dichotomy of interest in trivial acceptor vs. trivial rejector) the "symmetric" of recall $(\frac{TP}{TP+FN})$ is not precision $(\frac{TP}{TP+FP})$ but c-precision $(\frac{TN}{TP+TN})$, the "contrapositive" of precision. In fact, while recall=1 and c-precision=0 for the trivial acceptor, c-precision=1 and recall=0 for the trivial rejector.

¹⁷While 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]).

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]). Note that in this case if macroaveraging is used it needs to be redefined on a per-document, rather than per-category basis. Note that this measure does not make sense for binary-valued CSV_i functions, since in this case ρ_i may not be varied at will.

- (2) effectiveness is computed as 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 , and breakeven is identified as the value of ρ (or π) for which the plot intersects the $\rho = \pi$ line. This idea relies on the fact that by increasing the τ_i 's from 0 to 1 π usually increases monotonically from $apc \stackrel{def}{=} \sum_{i=1}^{|\mathcal{C}|} g_{Te}(c_i)$ to a value near 1 and ρ always decreases monotonically from 1 to 0. 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^{18} .
- (3) effectiveness is computed as the value of the F_{β} function [van Rijsbergen 1979, Chapter 7], for some $0 \le \beta \le +\infty$ (e.g. [Cohen 1995a; Cohen and Singer 1999; Lewis and Gale 1994; Lewis 1995a; Moulinier et al. 1996; Ruiz and Srinivasan 1999]), where

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

In this formula 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 value of a classifier Φ is always less or equal than its F_1 value.

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

 $^{^{18}}$ 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 make ρ be equal to π is not necessarily desirable or informative, 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 the values of π and ρ are close enough, interpolated breakeven may not be a reliable indicator of the effectiveness of the classifier.

and with different values for parameter p. At the end of the process, the value that has yielded the best effectiveness is chosen for p.

7.2 Benchmarks for text categorization

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

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

Unfortunately, a lot of experimentation, both on Reuters and on other collections, has not been performed with these caveat in mind. By testing three different classifiers on five versions of Reuters, Yang [1999] has shown that a lack of compliance with these three conditions may significantly influence the experimental results. Table 6 lists the results of all experiments known to us that were performed on five major versions of the Reuters benchmark: Reuters-22173 "ModLewis" (column #1), Reuters-22173 "ModApté" (column #2), Reuters-22173 "ModWiener" (column #3), Reuters-21578 "ModApté" (column #4) and Reuters-21578[10] "ModApté" (column #5)¹⁹. Only experiments that have computed either a breakeven or an F_1 result have been listed, since other less popular effectiveness measures do not readily compare with these.

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

¹⁹ The Reuters-21578 collection may be freely downloaded for experimentation purposes from http://www.research.att.com/~lewis/reuters21578.html and is now considered the "standard" variant of Reuters. We have decided not to cover experiments performed on variants of the Reuters benchmark different from the five listed because the small number of authors that have experimented on the same variant makes the reported results difficult to interpret. This includes experiments performed on the original Reuters-22173 "ModHayes" [Hayes et al. 1990] and Reuters-21578 "ModLewis" [Cohen and Singer 1999].

			#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				
Вім	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
Rоссніо	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]		<u> </u>	1	.860	
k-NN	example-based	[Joachims 1998]			l	ĺ	.823
k-NN	example-based	[Lam and Ho 1998]			l	.820	
k-NN	example-based	[Yang 1999]	.690	.852	.820		
k-NN	example-based	[Yang and Liu 1999]				.856	
	SVM	[Dumais et al. 1998]			1	.870	.920
SVMLIGHT	SVM	[Joachims 1998]			l		.864
SVMLIGHT	SVM	[Li and Yamanishi 1999]			l	.841	
SVMLight	SVM	[Yang and Liu 1999]	L		l	.859	
AdaBoost.MH	committee	[Schapire and Singer 2000]		.860	l	ĺ	ĺ
	committee	[Weiss et al. 1999]				.878	
	Bayesian net	[Dumais et al. 1998]			1	.800	.850
	Bayesian net	[Lam et al. 1997]	$.542 \text{ (M}F_1)$				

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

Other collections that have been frequently used for TC evaluation are

- —the OHSUMED collection, set up by Hersh et al. [1994] and used in [Joachims 1998; Lam and Ho 1998; Lam et al. 1999; Lewis et al. 1996; Ruiz and Srinivasan 1999; Yang and Pedersen 1997]²⁰. The documents are titles or title-plus-abstract's from medical journals (OHSUMED actually consists of a subset of the Medline document base); the categories are the postable terms of the MESH thesaurus.
- —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.
- —the AP collection, used in [Cohen 1995a; Cohen 1995b; Cohen and Singer 1999;

²⁰The OHSUMED collection may be freely downloaded for experimentation purposes from ftp://medir.ohsu.edu/pub/ohsumed

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 19, i.e. because in no case a significant enough number of authors have experimented with the same collection and in the same experimental conditions, thus making comparisons difficult.

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

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

- —direct comparison: classifiers Φ' and Φ'' may be compared as they have been tested on the same benchmark collection Ω , typically by the same team of researchers and under the same background conditions. This is the method that yields the more reliable results.
- —indirect comparison: classifiers Φ' and Φ'' may be compared as
 - (1) they have been tested on collections Ω' and Ω'' , respectively, typically by two different teams of researchers and hence under possibly different background conditions.
 - (2) one or more "baseline" classifiers $\overline{\Phi}_1, \dots, \overline{\Phi}_m$ have been tested on both Ω' and Ω'' by the direct comparison method.

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

A number of interesting conclusions can be drawn from Table 6 by using these two methods. Concerning the relative "hardness" of the five collections, if by $\Omega' > \Omega''$ 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, and it looks that efficiency considerations or application-dependent issues might play a role in breaking such a close tie.
- —Neural networks and on-line linear classifiers work very well, although slightly worse than the previously mentioned methods.
- —Batch linear classifiers (Rocchio) and probabilistic Naïve Bayes classifiers definitely look the worst of learning-based classifiers. For Rocchio, these results confirm other earlier results by Schütze et al. [1995], who had found three classifiers based on linear discriminant analysis, linear regression, and neural networks, to perform about 15% better than Rocchio. It should be mentioned, however, that recent results by Schapire et al. [1998] rank Rocchio along the best performers once near-positives are used in training.
- —The data in Table 6 seem hardly sufficient to say anything about decision tree classifiers. However, it should be noted that the recent work by Dumais et al. [1998] in which a version of decision tree classifiers was shown to perform nearly as well as their top performing system (a support vector machine classifier) will probably renew the interest in this type of text classifiers, an interest that had dwindled after previous unimpressive results reported in earlier literature [Cohen and Singer 1999; Joachims 1998; Lewis and Catlett 1994; Lewis and Ringuette 1994].
- —By far the lowest performance is displayed by WORD, a classifier implemented by Yang [1999] and not including any learning component²¹.

Concerning Word and non-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. Unfortunately, this classifier has never been tested on the *standard* variants of Reuters mentioned in Table 6, and it is not clear [Yang 1999] whether the (small) test set of Reuters-22173 "ModHayes" on which the .90 breakeven value was obtained was chosen randomly, as safe scientific practice would demand. As a consequence, the fact that this figure may be indicative of the performance of Construe, and of the manual approach it represents, has been convincingly questioned [Yang 1999].

It is however important to bear in mind that the considerations above are not absolute and final judgments (if there may be any) on the comparative effectiveness of these TC methods. One of the reasons is that a particular applicative context may exhibit very different characteristics from the ones to be found in Reuters, and different classifiers may respond differently to these characteristics. An experimental study by Joachims [1998] involving support vector machines, k-NN, decision trees, Rocchio and Naïve Bayes, showed all these classifiers to have a similar effectiveness on categories with > 300 positive training examples per category. The

²¹Word is based on the comparison between documents and category names, each treated as a vector of weighted terms in the vector space model. Word was implemented by Yang with the only purpose of determining the difference in effectiveness that adding a learning component to a classifier brings about. Word is actually called STR in [Yang 1994; Yang and Chute 1994]. Another non-learning classifier is proposed in [Wong et al. 1996].

fact that this experiment involved the methods which have scored best (support vector machines, k-NN) and worst (Rocchio and Naïve Bayes) according to Table 6 results is indicative of the fact that conditions different from those of Reuters may very well invalidate conclusions drawn on this latter.

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

8. CONCLUSION

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

- —Its domains of application are numerous and important, and given the proliferation of documents in digital form they are bound to increase dramatically in both number and importance.
- —It is indispensable in many applications in which the sheer number of the documents to be classified and the short response time 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 expert judgment [Larkey and Croft 1996], by providing tools that quickly "suggest" plausible decisions.
- —It has reached effectiveness levels comparable to those obtained in manual TC with the involvement of trained professionals. The effectiveness of manual TC is not 100% anyway [Cleverdon 1984] and, perhaps more importantly, it is unlikely to be improved substantially by the progress of research. On the contrary, the levels of effectiveness of automated TC are growing at a steady pace, and even if it is likely that they will eventually reach a plateau well below the 100% level, this plateau will probably be higher that the effectiveness levels of manual TC.

One of the reasons why from the early '90s onwards the effectiveness of text classifiers has dramatically improved, is the entrance 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 theoretically weak methods such as Rocchio. In TC, the field of ML has 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, rather than just being applicable to small problems, 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 applications. Techniques typical of automated TC have already been extended successfully to the automated categorization of documents expressed in slightly different media; for instance:

- —the classification of 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.
- —the classification of 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 current situation is not as bright. Current research on automatic document categorization under thematic categories mostly focuses on the text medium (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 in the automatic indexing of nontextual media is still an open research problem. While there are systems that attempt to detect content e.g. in images by recognising shapes, colour distributions and texture, the general problem of image semantics is still unsolved. The main reason is that natural language, the language of the text medium, admits far fewer variations than the "languages" employed by the other media. For instance, while the concept of a house can be "triggered" by relatively few natural language expressions such as house, houses, home, housing, inhabiting, etc., it can be triggered by far more images: the images of all the different houses that exist, of all possible colours and shapes, viewed from all the possible perspectives, from all the possible distances, etc. If we had solved the multimedia indexing problem in a satisfactory way, the general methodology (i.e. classifier induction, experimental evaluation, etc.) that we have discussed in this paper for text would also apply to automated multimedia 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.
- Blosseville, M., Hebrail, G., Montell, M., and Penot, N. 1992. Automatic document classification: natural langage processing and expert system techniques used together. In *Proceedings of SIGIR-92, 15th ACM International Conference on Research and Development in Information Retrieval* (Kobenhavn, DK, 1992), pp. 51–57.
- Borko, H. and Bernick, M. 1963. Automatic document classification. *Journal of the Association for Computing Machinery* 10, 2, 151–161.
- BRUCKNER, T. 1997. The text categorization system TEKLIS at TREC-6. In *Proceedings* of TREC-6, 6th Text Retrieval Conference (Gaithersburg, US, 1997), pp. 619–621.
- 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. 1998. 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. 1998. 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. 1996. Learning rules that classify e-mail. In *Proceedings of the AAAI Spring Symposium on Machine Learning in Information Access* (Stanford, US, 1996), pp. 18–25.
- 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.
- DE BUENAGA RODRÍGUEZ, M., GÓMEZ-HIDALGO, J. M., AND DÍAZ-AGUDO, B. 1997. Using WordNet to complement training information in text categorization. In *Proceedings of RANLP-97, 2nd International Conference on Recent Advances in Natural Language Processing* (Tzigov Chark, BL, 1997).
- 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.
- DIAO, Y., Lu, H., AND Wu, D. 2000. A comparative study of classification-based personal e-mail filtering. In Proceedings of PAKDD-00, 4th Pacific-Asia Conference on Knowledge Discovery and Data Mining (Heidelberg, DE, 2000), pp. 408–419.
- 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.
- DÖRRE, J., GERSTL, P., AND SEIFFERT, R. 1999. Text mining: finding nuggets in mountains of textual data. In *Proceedings of KDD-99, 5th ACM International Conference on Knowledge Discovery and Data Mining* (San Diego, US, 1999), pp. 398–401.
- 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.
- FIELD, B. 1975. Towards automatic indexing: automatic assignment of controlled-language indexing and classification from free indexing. *Journal of Documentation* 31, 4, 246–265.
- Forsyth, R. S. 1999. New directions in text categorization. In A. Gammerman Ed., Causal models and intelligent data management, pp. 151–185. Heidelberg, DE: Springer.
- FUHR, N. 1985. A probabilistic model of dictionary-based automatic indexing. In Proceedings of RIAO-85, 1st International Conference "Recherche d'Information Assistee par Ordinateur" (Grenoble, FR, 1985), pp. 207–216.

- Fuhr, N. 1989. Models for retrieval with probabilistic indexing. *Information Processing* and Management 25, 1, 55–72.
- Fuhr, N. and Buckley, C. 1991. A probabilistic learning approach for document indexing. $ACM\ Transactions\ on\ Information\ Systems\ 9,\ 3,\ 223-248.$
- Fuhr, N., 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.
- 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.
- GOODMAN, M. 1990. PRISM: a case-based telex classifier. In *Proceedings of IAAI-90, 2nd Conference on Innovative Applications of Artificial Intelligence* (1990), pp. 25–37.
- 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.
- Hamill, K. A. and Zamora, A. 1980. The use of titles for automatic document classification. *Journal of the American Society for Information Science* 33, 6, 396–402.
- HAYES, P. J., ANDERSEN, P. M., NIRENBURG, I. B., AND SCHMANDT, L. M. 1990. TCS: a shell for content-based text categorization. In *Proceedings of CAIA-90*, 6th IEEE Conference on Artificial Intelligence Applications (Santa Barbara, US, 1990), pp. 320–326.
- Heaps, H. 1973. A theory of relevance for automatic document classification. *Information and Control 22*, 3, 268–278.
- HEARST, M. A. 1991. Noun homograph disambiguation using local context in large corpora. In Proceedings of the 7th Annual Conference of the University of Waterloo Centre for the New Oxford English Dictionary (Oxford, UK, 1991), pp. 1–22.
- Hersh, W., Buckley, C., Leone, T., and Hickman, D. 1994. Ohsumed: an interactive retrieval evaluation and new large text collection for research. In *Proceedings of SIGIR-94, 17th ACM International Conference on Research and Development in Information Retrieval* (Dublin, IE, 1994), pp. 192–201.
- HOYLE, W. 1973. Automatic indexing and generation of classification by algorithm. *Information Storage and Retrieval 9*, 4, 233–242.
- HULL, D. A. 1994. Improving text retrieval for the routing problem using latent semantic indexing. In Proceedings of SIGIR-94, 17th ACM International Conference on Research and Development in Information Retrieval (Dublin, IE, 1994), pp. 282–289.
- Hull, D. A. 1998. The TREC-7 filtering track: description and analysis. In *Proceedings of TREC-7*, 7th Text Retrieval Conference (Gaithersburg, US, 1998), pp. 33–56.
- Hull, D. A., Pedersen, J. O., and Schütze, H. 1996. Method combination for document filtering. In *Proceedings of SIGIR-96*, 19th ACM International Conference on Research

- and Development in Information Retrieval (Zürich, CH, 1996), pp. 279-288.
- ITTNER, D. J., LEWIS, D. D., AND AHN, D. D. 1995. Text categorization of low quality images. In Proceedings of SDAIR-95, 4th Annual Symposium on Document Analysis and Information Retrieval (Las Vegas, US, 1995), pp. 301–315.
- IWAYAMA, M. AND TOKUNAGA, T. 1995. Cluster-based text categorization: a comparison of category search strategies. In Proceedings of SIGIR-95, 18th ACM International Conference on Research and Development in Information Retrieval (Seattle, US, 1995), pp. 273-281.
- JOACHIMS, T. 1997. A probabilistic analysis of the Rocchio algorithm with TFIDF for text categorization. In *Proceedings of ICML-97*, 14th International Conference on Machine Learning (Nashville, US, 1997), pp. 143–151.
- JOACHIMS, T. 1998. Text categorization with support vector machines: learning with many relevant features. In *Proceedings of ECML-98*, 10th European Conference on Machine Learning (Chemnitz, DE, 1998), pp. 137–142.
- JOACHIMS, T. 1999. Transductive inference for text classification using support vector machines. In *Proceedings of ICML-99*, 16th International Conference on Machine Learning (Bled, SL, 1999), pp. 200–209.
- JOACHIMS, T. AND SEBASTIANI, F. 2001. Special issue on text categorization: Guest editorial. *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 Recog*nition 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.
- 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).
- 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.
- Lewis, D. D., Stern, D. L., and Singhal, A. 1999. Attics: a software platform for online text classification. In *Proceedings of SIGIR-99, 22nd ACM International Conference on Research and Development in Information Retrieval* (Berkeley, US, 1999), pp. 267–268.
- LI, H. AND YAMANISHI, K. 1999. Text classification using ESC-based stochastic decision lists. In Proceedings of CIKM-99, 8th ACM International Conference on Information and Knowledge Management (Kansas City, US, 1999), pp. 122–130.
- LI, Y. H. AND JAIN, A. K. 1998. Classification of text documents. The Computer Journal 41, 8, 537–546.
- LIDDY, E. D., PAIK, W., AND YU, E. S. 1994. Text categorization for multiple users based on semantic features from a machine-readable dictionary. ACM Transactions on Information Systems 12, 3, 278–295.
- LIERE, R. AND TADEPALLI, P. 1997. Active learning with committees for text categorization. In Proceedings of AAAI-97, 14th Conference of the American Association for Artificial Intelligence (Providence, US, 1997), pp. 591–596.

- LIM, J. H. 1999. Learnable visual keywords for image classification. In Proceedings of DL-99, 4th ACM Conference on Digital Libraries (Berkeley, US, 1999), pp. 139–145.
- 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. 1998. Learning to classify text from labeled and unlabeled documents. In *Proceedings of AAAI-98, 15th Conference of the American Association for Artificial Intelligence* (Madison, US, 1998), pp. 792–799. An extended version appears as [Nigam et al. 2000].
- 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.
- Papka, R. and Allan, J. 1998. Document classification using multiword features. In *Proceedings of CIKM-98*, 7th ACM International Conference on Information and Knowledge Management (Washington, US, 1998), pp. 124–131.
- Pazienza, M. T. Ed. 1997. Information extraction. Number 1299 in Lecture Notes in Computer Science. Springer, Heidelberg, DE.
- RAGAS, H. AND KOSTER, C. H. 1998. Four text classification algorithms compared on a Dutch corpus. In *Proceedings of SIGIR-98*, 21st ACM International Conference on Research and Development in Information Retrieval (Melbourne, AU, 1998), pp. 369–370.

- RAU, L. F. AND JACOBS, P. S. 1991. Creating segmented databases from free text for text retrieval. In *Proceedings of SIGIR-91*, 14th ACM International Conference on Research and Development in Information Retrieval (Chicago, US, 1991), pp. 337–346.
- RILOFF, E. 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.
- ROUSSINOV, D. AND CHEN, H. 1998. A scalable self-organizing map algorithm for textual classification: A neural network approach to thesaurus generation. *Communication and Cognition* 15, 1-2, 81–112.
- 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.
- 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.
- Tong, R., Winkler, A., and Gage, P. 1992. Classification trees for document routing: A report on the TREC experiment. In *Proceedings of TREC-1*, 1st Text Retrieval Conference (Gaithersburg, US, 1992), pp. 209–228.
- 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.
- Wong, S. M. and Yao, Y. 1995. On modeling information retrieval with probabilistic inference. ACM Transactions on Information Systems 13, 1, 38–68.
- Yang, Y. 1994. Expert network: effective and efficient learning from human decisions in text categorisation and retrieval. In *Proceedings of SIGIR-94, 17th ACM International Conference on Research and Development in Information Retrieval* (Dublin, IE, 1994), pp. 13–22.
- Yang, Y. 1995. Noise reduction in a statistical approach to text categorization. In *Proceedings of SIGIR-95*, 18th ACM International Conference on Research and Development in Information Retrieval (Seattle, US, 1995), pp. 256–263.
- Yang, Y. 1999. An evaluation of statistical approaches to text categorization. *Information Retrieval* 1, 1-2, 69–90.
- Yang, Y. and Chute, C. G. 1994. An example-based mapping method for text categorization and retrieval. ACM Transactions on Information Systems 12, 3, 252–277.
- Yang, Y. and Liu, X. 1999. A re-examination of text categorization methods. In *Proceedings of SIGIR-99, 22nd ACM International Conference on Research and Development in Information Retrieval* (Berkeley, US, 1999), pp. 42–49.
- Yang, Y. and Pedersen, J. O. 1997. A comparative study on feature selection in text categorization. In *Proceedings of ICML-97*, 14th International Conference on Machine Learning (Nashville, US, 1997), pp. 412–420.
- 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.