A Tutorial on Automated Text Categorisation

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

Abstract

The automated categorisation (or classification) of texts into topical categories has a long history, dating back at least to 1960. Until the late '80s, the dominant approach to the problem involved *knowledge-engineering* automatic categorisers, i.e. manually building a set of rules encoding expert knowledge on how to classify documents. In the '90s, with the booming production and availability of on-line documents, automated text categorisation has witnessed an increased and renewed interest. A newer paradigm based on *machine learning* has superseded the previous approach. Within this paradigm, 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 are a very good effectiveness, a considerable savings in terms of expert manpower, and domain independence. In this tutorial we look at the main approaches that have been taken towards automatic text categorisation within the general machine learning paradigm. Issues of document indexing, classifier construction, and classifier evaluation, will be touched upon.

1 A definition of the text categorisation task

Document *categorisation* (or *classification*) may be seen as the task of determining an *assignment* of a value from {0,1} to each entry of the *decision matrix*

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

where $C = \{c_1, \ldots, c_m\}$ is a set of pre-defined *categories*, and $D = \{d_1, \ldots, d_n\}$ is a set of documents to be categorised (sometimes called "*requests*"). A value of 1 for a_{ij} is interpreted as a decision to file d_j under c_i, while a value of 0 is interpreted as a decision not to file d_j under c_i.

Fundamental to the understanding of this task are two observations:

- the categories are just symbolic labels. No additional knowledge of their "meaning" is available to help in the process of building the categoriser; in particular, this means that the "text" constituting the label (e.g. Sports in a news categorisation task) cannot be used;
- the attribution of documents to categories should, in general, be attributed on the basis of the *content* of the documents, and not on the basis of *metadata* (e.g. publication date,

document type, etc.) that may be available from an external source. This means that the notion of *relevance* of a document to a category is inherently *subjective*[1].

Different constraints may be enforced on the categorisation task, depending on the application: we may want that

1. $\{\leq 1 \mid 1 \mid \geq 1 \mid \dots\}$ elements of C must be assigned to each element of D. When exactly one category is assigned to each document (as e.g. in [28, 43]), this is often referred to as the *non-overlapping categories* case.

2. each element of C must be assigned to $\{\leq 1 \mid 1 \mid \geq 1 \mid \dots\}$ elements of D.

The techniques we will consider here are applicable irrespectively of whether any of these constraints are enforced or not.

1.1 Category- or document-pivoted categorisation

An important distinction is whether we want to fill the matrix one row at a time (*category-pivoted categorisation* – CPC), or fill the it one column at a time (*document-pivoted categorisation* – DPC). This distinction is mostly pragmatic rather than conceptual, but is important in the sense that the sets C of categories and D of documents are not always available in their entirety right from the start.

DPC is thus suitable when documents might become available one at a time over a long span of time, e.g. in the case a user submits one document at a time for categorisation, rather than submitting a whole batch of them all at once. In this case, sometimes the categorisation task takes the form of ranking the categories in decreasing order of their estimated appropriateness for document d; because of this, CPC is sometimes called *category-ranking classification* or *on-line classification* [50].

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

CPC is more commonly used than DPC, as the case in which documents are submitted one at a time is somehow more common than the case in which newer categories dynamically crop up. However, although some specific techniques apply to one and not to the other (e.g. the proportional thresholding method discussed in Section 6, which applies only to CPC), this is more the exception than the rule: most of the techniques we will discuss in this paper allow the construction of classifiers capable of working in either mode.

2 Applications of document categorisation

Automatic text categorisation goes back at least to the early '60, with the seminal work by Maron [29]. Since then, it has been used in a number of different applications. In the following, we briefly review the most important ones.

[1] This is exemplified by the well-known phenomenon of *inter-indexer inconsistency* [5]: when two different humans must take a decision on whether to categorise document d under category c, they may disagree. Note that the above-mentioned notion of relevance of a document to a category is basically the notion of relevance of a document to an information need, as from information retrieval [41].

2.1 Automatic indexing for Boolean information retrieval systems

The first use to which automatic categorisers were put at, and the application that spawned most of the early research in the field, is that of automatic document indexing for use in information retrieval (IR) systems relying on a controlled dictionary. The most prominent example of such IR systems is, of course, that of Boolean systems. In these systems, each document is assigned one or more keywords or keyphrases describing its content, where these keywords and keyphrases belong to a finite set of words, called *controlled dictionary*) and often consisting of a hierarchical thesaurus (e.g. the NASA thesaurus for the aerospace discipline [34], or the MESH thesaurus covering the medical field [35]). Usually, this assignment is performed by trained human indexers, and is thus an extremely costly activity.

If the entries in the thesaurus are viewed as categories, document indexing becomes an instance of the document categorisation task, and may thus be addressed by the automatic techniques described in this paper. Concerning Point 1 in Section 1, note that in this case a typical constraint may be that $l_1 \leq x \leq l_2$ keywords are assigned to each document. Document-pivoted categorisation might typically be the best option, so that documents are categorised on a first come, first served basis.

Various automatic document categorisers explicitly addressed at the document indexing application have been described in the literature; see e.g. [4, 10, 12].

2.2 Document organisation

In general, all issues pertaining to document organisation and filing , be it for purposes of personal organisation or document repository structuring, may be addressed by automatic categorisation techniques. For instance, at the offices of a newspaper, incoming "classified" ads should be, prior to publication, categorised under the categories used in the categorisation scheme adopted by the newspaper; typical categories might be e.g. Personals, Cars for sale, Real estate, ... While most newspapers would handle this application manually, those dealing with a high daily number of classified ads might prefer an automatic categorisation system to choose the most suitable category for a given ad.

Concerning Point 1 in Section 1, note that in this case a typical constraint might be that exactly one category is assigned to each document. Again, a first-come, first-served policy might look the aptest here, which would make one lean for a document-pivoted categorisation style.

2.3 Document filtering

Document filtering (also known as *document routing*) refers to the activity of categorising a *dynamic*, rather than static, collection of documents, in the form of a stream of incoming documents dispatched in an asynchronous way by an information producer to an information consumer [3]. A typical case of this is a newsfeed, whereby the information producer is a news agency (e.g. Reuters or Associated Press) and the information consumer is a newspaper. In this case, the filtering system should discard (i.e. block the delivery to the consumer of) the documents the consumer is not likely to be interested in (e.g. all news not concerning sports, in the case of a sports newspaper).

Filtering can be seen as a special case of categorisation with non-overlapping categories, i.e. the categorisation of incoming documents in two categories, the relevant and the irrelevant. Additionally, a filtering system may also perform a further categorisation into topical categories of the documents deemed relevant to the consumer; in the example above, all articles about sports are deemed relevant, and should be further subcategorised according e.g. to which sport they deal with, so as to allow individual journalists specialised in individual sports to access only documents of high prospective interest for them.

The construction of information filtering systems by means of machine learning techniques is widely discussed in the literature: see e.g. [43].

2.4 Word sense disambiguation

Word sense disambiguation (WSD - see e.g. [20]) refers to the activity of finding, given the occurrence in a text of an ambiguous (i.e. polysemous or homonymous) word, the word sense the word refers to. 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 I got some money on loan from the bank this morning refers to. WSD is very important for a number of applications, including indexing documents by word senses rather than by words for information retrieval or other content-based document management applications.

WSD may be seen as a categorisation task once we view word occurrence contexts as documents and word senses as categories. Quite obviously, this is a case in which exactly one category needs to be assigned to each document, and one in which document-pivoted categorisation is most likely to be the right choice. WSD is viewed as a categorisation task in a number of different works in the literature; see e.g [14, 42].

2.5 Yahoo!-style search space categorisation

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

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

3 The machine learning approach to document categorisation

In the '80s, the main approach used to the construction of automatic document categorisers involved *knowledge-engineering* them, i.e. manually building an expert system capable of taking categorisation decisions. Such an expert system might have typically consisted of a set of manually defined rules (one per category) of type **if** ⟨*DNF Boolean formula*⟩ **then** ⟨*category*⟩, to the effect that if the document satisfied ⟨*DNF Boolean formula*⟩ (*DNF* standing for "disjunctive normal form"), then it was categorised under ⟨*category*⟩. The typical example of this approach is the CONSTRUE system [15], built by Carnegie Group for use at the Reuters news agency.

The drawback of this "manual" approach to the construction of automatic classifiers is the existence of a *knowledge acquisition bottleneck*, similarly to what happens in expert systems. That is, rules must be manually defined by a knowledge engineer with the aid of a domain expert (in this case, an expert in document relevance to the chosen set of categories). If the set of categories is updated, then these two professional figures must intervene again, and if the classifier is ported to a completely different domain (i.e. set of categories), the work has to be repeated anew.

On the other hand, it was suggested that this approach can give very good effectiveness results[2]: Hayes et al. [15] report a .90 "breakeven" result (see Section 7) on a subset of the Reuters-21578 test collection, a figure that outperforms most of the best classifiers built in the late '90s by machine learning techniques. However, no other classifier has been tested on the same dataset as CONSTRUE (see also Table 4), and it is not clear how this dataset was selected from the Reuters-21578 collection (i.e. whether it was a random or a favourable subset of the whole collection). All

[2]See Section 7 for a more formal definition of "effectiveness" in the context of automatic categorisation.

in all, as convincingly argued in [50], the results above do not allow us to confidently say that these effectiveness results may be obtained in the general case.

Since the early '90s, a new approach to the construction of automatic document classifiers (the *machine learning approach*) has gained prominence and eventually become the dominant one (see [31] for a comprehensive introduction to machine learning). In the *machine learning approach* a general *inductive process* automatically builds a classifier for a category c_i by "observing" the characteristics of a set of documents that have previously been classified manually under c_i by a domain expert; from these characteristics, the inductive process gleans the characteristics that a novel document should have in order to be categorised under c_i. Note that this allows us to view the construction of a classifier for the set of categories $C = \{c_1, \ldots, c_m\}$ as m independent tasks of building a classifier for a single category $c_i \in C$, each of these classifiers being a rule that allows to decide whether document d_j should be categorised under category c_i.

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

In terms of effectiveness, categorisers build by means of machine learning techniques nowadays achieve impressive levels of performance (see Section 7), making automatic classification a qualitatively viable alternative to manual classification.

3.1 Training set and test set

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

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

A value of 1 for ca_{ij} is interpreted as an indication from the expert to file d_j under c_i, while a value of 0 for is interpreted as an indication from the expert not to file d_j under c_i. A document d_j is often referred to as a *positive example* of c_i if $ca_{ij} = 1$, a *negative example* of c_i if $ca_{ij} = 0$. For evaluation purposes, in the first stage of classifier construction the initial corpus is typically divided into two sets, not necessarily of equal size:

- a *training set* $Tr = \{\overline{d}_1, \ldots, \overline{d}_g\}$. This is the set of example documents observing the characteristics of which the classifiers for the various categories are induced;
- a *test set* $Te = \{\overline{d}_{g+1}, \ldots, \overline{d}_s\}$. This set will be used for the purpose of testing the effectiveness of the induced classifiers. Each document in Te will be fed to the classifiers, and the classifier decisions compared with the expert decisions; a measure of classification effectiveness will be based on how often the values for the a_{ij}'s obtained by the classifiers match the values for the ca_{ij}'s provided by the experts.

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

it is often the case that in order to optimise a classifier, its internal parameters should be tuned by testing which value of the parameter yields the best effectiveness. In order for this to happen and, at the same time, safeguard scientific standards, the set $\{\overline{d}_1, \ldots, \overline{d}_g\}$ may be further split into a "true" training set $Tr = \{\overline{d}_1, \ldots, \overline{d}_f\}$, from which the classifier is induced, and a *validation test set* $Va = \{\overline{d}_{f+1}, \ldots, \overline{d}_g\}$, on which the repeated tests of the induced classifier aimed at parameter optimization are performed.

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

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

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

3.2 Information retrieval techniques and document classification

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

Information retrieval techniques are used in three phases of the classification task:

1. IR-style *indexing* is always (uniformly, i.e. by means of the same technique) performed on the documents of the initial corpus and on those to be categorised during the operating phase of the classifier;

2. IR-style techniques (such as document-request matching, query expansion, ...) are typically 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 (e.g. [2]) non-standard approaches to Step 1 are also used. Steps 1, 2 and 3 will be the main themes of Sections 4, 5 and 7, respectively.

4 Indexing and dimensionality reduction

In true information retrieval style, each document (either belonging to the initial corpus, or to be categorised in the operating phase of the system) is usually represented by a vector of n weighted *index terms*. Weights usually range between 0 and 1 (only a few authors (e.g. [26, 28, 43]) use binary weights), and with no loss of generality we will assume they always do. This is often referred to as the *bag of words* approach to document representation. Both Apté et al. [1] and Lewis [21] have found that more sophisticated representations yield worse categorisation effectiveness, thereby confirming similar results from information retrieval [39]. In particular, [21] has tried to use *noun phrases*, rather than individual words, as indexing terms, but the experimental results he has found have not been encouraging, irrespectively of whether the notion of "phrase" is motivated

- linguistically; i.e. the phrase is such according to a grammar of the language (*syntactic phrase*; see e.g. [44]);

- statistically; i.e. the phrase is not grammatically such, but is composed of a set/sequence of words that occur contiguously with high frequency in the collection (see e.g. [9]).

Quite convincingly, Lewis argues that the likely reason for the discouraging results is that, although indexing languages based on phrases have superior semantic qualities, they have inferior statistical qualities with respect to indexing languages based on single words. Notwithstanding these discouraging results, investigations on the effectiveness of phrase indexing are still being actively pursued [32, 33].

In general, for determining the weight w_{jk} 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. [39]), defined as

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

where $\#(t_k, d_j)$ denotes the number of times t_k occurs in d_j, $\#(t_k)$ denotes the number of documents in Tr in which t_k occurs at least once (also known as the *document frequency* of term t_k), and g is the cardinality of the training set Tr. This function encodes the intuitions that i) the more often a term occurs in a document, the more it is representative of the content of the document, and ii) the more documents the term occurs in, the less discriminating it is[3]. In order to make weights fall in the [0,1] interval and documents be represented by vectors of equal length, the weights resulting from $tfidf$ are often normalised by cosine normalisation, given by:

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

where T is the set of all terms that occur at least once in Tr. Although $tfidf$ is by far the most popular one, other indexing functions have also been used (e.g. [11]).

Before indexing, the removal of function words is usually performed, while only a few authors (e.g. [36, 43, 46, 50]) perform stemming.

Unlike in information retrieval, in document categorisation the high dimensionality of the term space (i.e. the fact that the number r of terms that occur at least once in the corpus Co is high) may be problematic. In fact, while the typical matching algorithms used in IR (such as cosine matching) scale well to high values of r, the same cannot be said of many among the sophisticated learning algorithms used for classifier induction (e.g. the LLSF algorithm of [51]). Because of this, techniques for *dimensionality reduction* (DR) are often employed whose effect is to reduce the dimensionality of the vector space from r to $r' \ll r$.

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

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

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

[3]Actually, $tfidf$ is, rather than a function, a whole class of functions, which differ from each other in terms of normalisation or other correction factors being applied or not. Formula 2 is then just one of the possible instances of this class; see [39] for variations on this theme.

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

- *local dimensionality reduction*: for each category c_i, $r'_i \ll r$ features are chosen in terms of which the classifier for category c_i will operate (see e.g. [1, 26, 28, 36, 43, 46]). Conceptually, this would mean that each document d_j has a different representation for each category c_i; in practice, though, this means that different subsets of d_j's original representation are used when categorising under the different categories;
- *global dimensionality reduction*: $r' \ll r$ features are chosen in terms of which the classifiers for all categories $C = \{c_1, \ldots, c_m\}$ will operate (see e.g. [46, 50, 53]).

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) either for local or for global DR.

A second, orthogonal distinction may be drawn in terms of what kind of features are chosen:

- *dimensionality reduction by feature selection*: the chosen features are a subset of the original r features;
- *dimensionality reduction by feature extraction*: the chosen features are not a subset of the original r features. Usually, the chosen are not homogeneous with the original features (e.g. if the original r features are words, the chosen features may not be words at all), but are obtained by combinations or transformations of the original ones.

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

4.1 Feature selection

Given a fixed $r' \ll r$, techniques for feature selection (also called *term space reduction* – TSR) purport to select, from the original set of r features, the r' terms that, when used for document indexing, yield the smallest reduction in effectiveness with respect to the effectiveness that would be obtained by using full-blown representations. Results published in the literature [53] have even shown a moderate ($\leq 5\%$) increase in effectiveness after term space reduction has been performed, depending on the classifier, on the *aggressivity* $\frac{r}{r'}$ of the reduction, and on the TSR technique used.

Global TSR is usually tackled by keeping the $r'_i \ll r$ terms that score highest according to a predetermined numerical function that measures the "importance" of the term for the categorisation task.

4.1.1 Document frequency

A simple and surprisingly effective global TSR function is the *document frequency* $\#(t_k)$ of a term t_k, first used in [1] and then systematically studied in [53]. Note that in this section we will interpret the event space as the set of all documents in the training set; in probabilistic terms, document frequency may thus also be written as $P(t_k)$. Yang and Pedersen [53] have shown that, irrespectively of the adopted classifier and of the initial corpus used, by using $\#(t_k)$ as a feature selection technique it is possible to reduce the dimensionality of the term space by a factor of 10 with no loss in effectiveness (a reduction by a factor of 100 brings about just a small loss).

This result seems to state, basically, that the most valuable terms for categorisation are those that occur more frequently in the collection. As such, it would seem at first to contradict a truism of information retrieval, according to which the most informative terms are those with low-to-medium document frequency [39]. But these two results do not contradict each other, since it is well-known (see e.g. [40]) that the overwhelming majority of the words that occur at least once in a given corpus have an extremely low document frequency; this means that by performing a TSR by a factor of 10 using document frequency, only such words are removed, while the words from low-to-medium to high document frequency are preserved.

Finally, note that a slightly more empirical form of feature selection by document frequency is adopted by many authors (e.g. [18, 28, 46]), who remove from consideration all terms that occur

Function	Denoted by	Mathematical form	Used in
Document frequency	$\#(t_k, c_i)$	$P(t_k, c_i)$	[1, 53]
Information gain	$IG(t_k, c_i)$	$P(t_k, c_i) \cdot log\frac{P(t_k, c_i)}{P(c_i) \cdot P(t_k)} + P(\overline{t}_k, c_i) \cdot log\frac{P(\overline{t}_k, c_i)}{P(c_i) \cdot P(\overline{t}_k)}$	[21, 26, 53]
Chi-square	$\chi^2(t_k, c_i)$	$\frac{g \cdot [P(t_k, c_i) \cdot P(\overline{t}_k, \overline{c}_i) - P(t_k, \overline{c}_i) \cdot P(\overline{t}_k, c_i)]^2}{P(t_k) \cdot P(\overline{t}_k) \cdot P(c_i) \cdot P(\overline{c}_i)}$	[43, 53]
Correlation coefficient	$CC(t_k, c_i)$	$\frac{\sqrt{g} \cdot [P(t_k, c_i) \cdot P(\overline{t}_k, \overline{c}_i) - P(t_k, \overline{c}_i) \cdot P(\overline{t}_k, c_i)]}{\sqrt{P(t_k) \cdot P(\overline{t}_k) \cdot P(c_i) \cdot P(\overline{c}_i)}}$	[36]
Relevancy score	$RS(t_k, c_i)$	$log\frac{P(t_k \mid c_i) + d}{P(\overline{t}_k \mid \overline{c}_i) + d}$	[46]

Table 1: Main functions proposed for term space reduction purposes. Information gain is also known as *expected mutual information*; it is used under this name by Lewis [21, page 44]. In the χ^2 and CC formulae, g is as usual the cardinality of the training set. In the $RS(t_k, c_i)$ formula d is a constant damping factor.

in at most x training documents (popular values for x range from 1 to 3), either as the only form of dimensionality reduction [18] or before applying another more sophisticated form [28, 46].

4.1.2 Other information-theoretic TSR functions

Other more sophisticated information-theoretic functions have been used in the literature; the most important of them are summarised in Table 1. Probabilities are interpreted as usual on an event space consisting of the initial corpus (e.g. $P(\overline{t}, c_i)$ thus means the probability that, for a random document x, term t does not occur in x and x does not belong to category c_i), and are estimated by counting occurrences in the training set. Most of these functions try to capture the intuition according to which the most valuable terms for categorisation under c_i are those that are distributed most differently in the sets of positive and negative examples of the category.

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

Unfortunately, the complexity of some of these information-theoretic measures does not always allow one to readily interpret why their results are so good; in other words, the rationale of the use of these measures as TSR functions is not always clear. In this respect, Ng et al. [36] have observed that the use of $\chi^2(t)$ for TSR purposes is contrary to intuitions, as the power of 2 that appears in its formula has the effect of equating those factors that indicate a positive correlation between the term and the category (i.e. $P(t, c_i)$ and $P(\overline{t}, \overline{c}_i)$) with those that indicate a negative correlation (i.e. $P(t, \overline{c}_i)$ and $P(\overline{t}, c_i)$). The "correlation coefficient" $CC(t)$ they propose, being the square root of $\chi^2(t)$, emphasises thus the former and de-emphasises the latter, thus respecting intuitions. The experimental results by Ng et al. [36] show a superiority of $CC(t)$ over $\chi^2(t)$, but it has to be remarked that these results refer to a local, rather than global, term space reduction application.

The use of this function for TSR purposes has been tested in [13] on the Reuters-21578 collection (see Section 7) and on a variety of different classifiers, and its experimental results have outperformed those obtained by means of both $\chi^2(t)$ and $CC(t)$.

It is to be noted, however, that the reported improvements in performance that some TSR functions achieve over others cannot be taken as general statements of the properties of these functions unless the experiments involved have been carried out in thoroughly controlled conditions and on a variety of different situations (e.g. different classifiers, different initial corpora, ...). So far, only the comparative evaluations reported in [53], and partly those reported in [13], seem

conclusive in this respect.

4.2 Feature extraction

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

4.2.1 Term clustering

Term clustering aims at grouping words with a high degree of pairwise semantic relatedness into clusters, so that the clusters (or their centroids) may be used instead of the terms as dimensions of the vector space. Any term clustering method must specify i) a method for grouping words into clusters, and ii) a method for converting the original representation of a document d_j into a new representations for it based on the newly synthesized dimensions. One example of this approach is the work of Li and Jain [28], who view semantic relatedness between terms 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, as noted in Section 5.5, the small size of their experiment hardly allows any hard conclusion to be reached.

4.2.2 Latent semantic indexing

Latent semantic indexing [8] is a technique for dimensionality reduction originally developed in the context of information retrieval 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 other vectors of a lower-dimensional space whose dimensions are obtained as combinations of the original dimensions by looking at their patterns of co-occurrence. The function mapping original vectors into new vectors is obtained by applying a singular value decomposition to the incidence matrix formed by the original document vectors. In the context of text categorisation, this technique is applied by deriving the mapping function from the training set and applying it to each test document so as to produce a representation for it in the lower-dimensional space.

One characteristic of LSI as a dimensionality reduction function is that the newly obtained dimensions are not, unlike the cases of feature selection and term clustering, readily 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. [43, 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 dimensionality reduction had been performed by χ^2-based feature 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 in all evidence the words contained in the document had concurred to generate one or more of the LSI higher-order features that generate the document space of the category. How Schütze et al. 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" [43, page 230]. A drawback of LSI, though, is that if some original term is particularly good in itself at discriminating a category, that discriminatory power may be lost in the new vector space.

Wiener et al. [46] use LSI in two alternative ways: i) for local dimensionality reduction, thus creating several LSI representations specific to individual categories, and ii) for global dimensionality reduction, by creating a single LSI representation for the entire category set. Their experimental results show the former approach to perform better than the latter. Anyway, both LSI-based approaches are shown to perform better than a simple feature selection technique based on the relevancy weighting measure.

5 Building a classifier

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

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

1. the definition of a function $CSV_i : D \rightarrow [0, 1]$ that, given a document d, returns a *categorisation status value* for it, i.e. a number between 0 and 1 that, roughly speaking, represents the evidence for the fact that d should be categorised under c_i. The CSV function takes up different meanings according to the different classifiers: for instance, in the "Naive Bayes" approach discussed in Section 5.1 $CSV(d)$ is a probability, whereas in the "Rocchio" approach discussed in Section 5.2.1 $CSV(d)$ is a distance between vectors in r-dimensional space;

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

Issue 2 will be the subject of Section 6. Let us then concentrate on Issue 1. Lewis et al. [27] distinguish two main ways to build a classifier:

- *parametric*. According to this approach, training data are used to estimate parameters of a probability distribution. We will discuss this approach in detail in Section 5.1.

- *non-parametric*. This approach may be further subdivided in two categories:

  - *profile-based*. In this approach, a *profile* (or *linear classifier*) for the category, in the form of a vector of weighted terms, is extracted from the training documents pre-categorised under c_i. The profile is then used as a query against the documents D to be categorised; the documents with the highest *retrieval status value* (RSV) are categorised under c_i. We will discuss this approach in detail in Section 5.2.

  - *example-based* : According to this approach, the document d to be categorised is used as a query against the training set Tr. The categories under which the training documents with the highest CSV are categorised, are considered as promising candidates for categorising d. We will discuss this approach in detail in Section 5.3.

5.1 Parametric classifiers

The main example of the parametric approach is the probabilistic *Naive Bayes* classifier, used e.g. by Lewis [21], which is based on computing

$$CSV_i(d_j) \stackrel{def}{=} P(c_i|d_j) \stackrel{est}{=} \prod_{y=1}^{r} [P(c_i|t_y) \cdot P(t_y|d_j) + P(c_i|\overline{t}_y) \cdot P(\overline{t}_y|d_j)] \quad (3)$$

In this formula the r terms upon which the product is calculated are the full term set if no term space reduction has been applied, and the reduced set otherwise. However, Li and Jain [28] have found feature selection to be extremely counterproductive for the Naive Bayes approach.

In the learning phase, the four probabilities that intervene in the formula are estimated on the training set. The supposedly "naive" character of this approach derives from the fact that the probability of d_j falling under c_i is viewed as depending on the product of r *independent* factors. In reality, the occurrence of a term t' within a document d is not independent from the occurrence within d of another term t''; assuming otherwise (the *binary independence hypothesis*) is just a matter of convenience, as both computing and making practical use of the stochastic dependence between terms is computationally hard. However restrictive it may seem, the binary independence hypothesis has been shown to work surprisingly well in practice, both in an information retrieval [38] and in a classification context [24].

5.2 Profile-based classifiers

A *profile-based* (or *linear*) *classifier* is basically a classifier which embodies an explicit, or declarative, representation of the category on which it needs to take decisions. The learning phase consists then on the extraction of the profile of the category from the training set.

Profile-extraction may be typically preceded by *local* term space reduction (see Section 4.1), i.e. the r' most important terms for category c_i are selected according to some measure. For this, the "local" variants of the functions illustrated in Table 1 are usually employed[5]. For instance, Apté et al. [1] use document frequency to pick the 100 most frequent words for each category.

Linear classifiers are often partitioned in two broad classes, *incremental* classifiers and *batch* classifiers.

Incremental classifiers build a profile before analysing the whole training set, and refine this profile as they read new training documents. Example incremental classifiers are the *Widrow-Hoff* classifier and a refinement of it, the *exponentiated gradient* classifier, first introduced to the text classification literature in [27].

Batch classifiers instead build a profile by analyzing the training set all at once. Within the text classification literature, one example of a batch linear classifier is the one built by *linear discriminant analysis*, a model of the stochastic dependence between terms that relies the covariance matrices of the various categories [17, 43]. The foremost example of a batch linear classifier, however, is the Rocchio classifier discussed in Section 5.2.1.

5.2.1 Rocchio-style profile extraction

The Rocchio classifier relies on an adaptation to the text categorisation case of Rocchio's formula for relevance feedback in the vector-space model. This adaptation was first used in [18]; since then, Rocchio has been used by many authors, either as the main classifier [37] or as a baseline classifier [6, 11, 13, 27, 43].

Rocchio's classifier is defined by the formula

$$w_{yi} = \beta \cdot \sum_{\{\overline{d}_j \mid ca_{ij}=1\}} \frac{w_{yj}}{|\{\overline{d}_j \mid ca_{ij}=1\}|} + \gamma \cdot \sum_{\{\overline{d}_j \mid ca_{ij}=0\}} \frac{w_{yj}}{|\{\overline{d}_j \mid ca_{ij}=0\}|}$$

[5] By "local variant" of a TSR measure we mean a measure computed with reference only to the subset of the training set consisting of the documents belonging to the category c_i of interest.

where $\beta + \gamma = 1$, $\beta \geq 0$, $\gamma \leq 0$ and w_{yj} is the weight that term t_y has in document $\overline{d}_j$. In this formula, β and γ are control parameters that allow setting the relative importance of positive and negative examples. For instance, if β is set to 1 and γ to 0, this corresponds to viewing the profile of c_i as the *centroid* of the positive training examples of c_j. In general, the Rocchio classifier rewards the closeness of a document to the centroid of the positive training examples, and its distance from the centroid of the negative training examples. Clearly, most of the times the role of negative examples is de-emphasised, by setting β to a high value and γ to a low one; for instance, in [43] the values $\beta = 1$ and $\gamma = 0$ are chosen.

One issue in the application of the Rocchio formula to profile extraction is whether the set of negative training instances $\{\overline{d}_j \in Tr \mid ca_{ij} = 0\}$ should be considered in its entirety, or whether a well-chosen sample of it, such as the set of *near-positives* (defined as "the most positive amongst the negative training examples"), should be selected. When the original Rocchio formula is used for relevance feedback in information retrieval, 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. Most applications of the Rocchio formula to text categorisation (e.g. [18]) use the whole set of negative examples, as it is not easy to individuate near-positives. Near-positives would be more useful for training, since they are the most difficult to tell apart from the relevant documents. Fuhr et al. [11] use near-positives, as the application they work on (Web page categorisation into hierarchical catalogues) does allow such a selection to be performed; as a consequence, the contribution of the $\sum_{\{\overline{d}_j \mid ca_{ij}=0\}} \frac{w_{yj}}{|\{\overline{d}_j \mid ca_{ij}=0\}|}$ factor tends to be more significant.

One of the advantages of the Rocchio method is that it produces "understandable" classifiers, in the sense that the category profile it produces can be readily interpretable of, and thus heuristically tuned, by a human reader. This does not happen for other approaches such as e.g. neural networks.

The Rocchio classifier, as all linear classifiers, has the disadvantage that it basically divides the space of documents in two subspaces; any document falling within the former (in the case of Rocchio, an n-sphere) will be classified under c_i, while all documents falling within the latter will not. This situation is graphically depicted in Figure 1a, 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 clear from the fact that what Rocchio basically does is taking the average (centroid) of all positive examples, and as all averages this is only partly representative of the whole set[6].

5.3 Example-based classifiers

Example-based classifiers do not build an explicit, declarative representation of the category of interest, but "parasite" on the categorisation judgments that the experts have given on the training documents similar to the one to be categorised. These classifiers have thus been called *lazy* learning systems, since they do not involve a true training phase [50].

The first introduction of example-based methods in the text classification literature is due to Masand et al. [7, 30]. Our presentation of the example-based approach will be based instead on the *k-NN* (for "k nearest neighbours") algorithm implemented by Yang in the ExpNet system [49]. For deciding whether d_j should be classified under c_i, k-NN looks at whether the k training documents most similar to d_j have also been classified under c_i; if the answer is positive for a large enough proportion of them, a positive categorisation decision is taken, and a negative decision is taken otherwise.

Mathematically, classifying a document by means of k-NN comes down to computing

$$CSV_i(d_j) = \sum_{\overline{d}_z \in TR_k(d_j)} RSV(d_j, \overline{d}_z) \cdot ca_{iz} \quad (4)$$

[6]The subdivision into different clusters of the positive training examples which is depicted in Figure 1a is quite plausible. This could be the case, for example, of a news categorisation task, and of a set of positive training examples for the category Sports; one of the clusters could consist of articles about basketball and another about skiing, with the former ones having little in common with the latter ones.

where $TR_k(d_j)$ is the set of the k documents $\overline{d}_z$ for which $RSV(d_j, \overline{d}_z)$ is maximum and the ca_{iz} values are from the correct decision matrix of Section 3.1. In turn, $RSV(d_j, \overline{d}_z)$ represents some measure or semantic relatedness, or mutual relevance, between documents d_j and $\overline{d}_z$; any matching function, be it probabilistic or vector-based, from a ranked information retrieval system may be used for this purpose ([49] uses the cosine similarity typical of vector-space retrieval).

The construction of a k-NN classifier also involves determining a threshold k, indicating how many top-ranked training documents have to be considered for computing $CSV_i(d_j)$. This threshold is usually determined experimentally; in her experiments Yang [49, 50] has found $k = 30$ to yield the best effectiveness and values higher than that to yield no significant loss in performance.

Note that k-NN, unlike linear classifiers, does not subdivide the document space in just two subspaces, hence it does not suffer from the problem discussed at the end of Section 5.2.1. This is graphically depicted in Figure 1b, where the more "local" character of k-NN with respect to Rocchio can be appreciated.

Besides its remarkable efficiency, which has been proven through a number of different experiments (see Section 7.3), one of the advantages of k-NN is its efficiency, as the classification of a document in to the m categories of interest can be performed in time linear in the cardinality g of the training set [50].

Various nearest neighbour techniques have been used in the text categorisation literature. Li and Jain [28] use a nearest neighbour technique in the context of a USENET postings classification task with non-overlapping categories. Their results show a lower effectiveness than a Naive Bayes approach (and this is somehow surprising, given the experiments of [50] discussed in Section 7.3), but might be possibly due to the fact that they use the (arguably unpromising) value $k = 1$ without any attempt at optimisation. An interesting variant of the basic k-NN approach is proposed by Galavotti [13], who reinterprets Equation (4) by redefining ca_{iz} as

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

The difference with the original k-NN approach is that a negative weight, instead of 0, is attributed to the negative examples of c_i. In this way, the fact that a document $\overline{d}_z$, similar to the document d_j we want to categorise under c_i, does not belong to c_i is not discarded, but weights *negatively* in the decision to categorise d_j under c_i.

5.4 Combining profile- and example-based classifiers

A combination of profile- and example-based methods is presented in [19]. In this work a k-NN system is fed, in place of training documents, what the authors call *generalised instances* (GIs). This approach may be seen as the result of

- clustering the positive instances of category c_i, thus obtaining a set of clusters $CL_i = \{cl_{i1}, \ldots, cl_{ik_i}\}$;
- extracting a profile $pr(cl_{iz})$ (*generalised instance*) from each cluster cl_{iz}, either with Rocchio, Widrow-Hoff or other algorithm for the extraction of linear classifiers;
- applying k-NN with profiles in place of training documents, i.e. computing

$$CSV_i(d_j) \stackrel{def}{=} \sum_{cl_{iz} \in CL_i} RSV(d_j, pr(cl_{iz})) \cdot \frac{|\ cl_{iz}\ |}{|\ \bigcup_{v=1}^{ik_i} cl_{iv}\ |}$$

This exploits the superior effectiveness (graphically illustrated in Figure 1) of k-NN over linear classifiers while at the same time avoiding the sensitivity of k-NN to noise in the training documents.

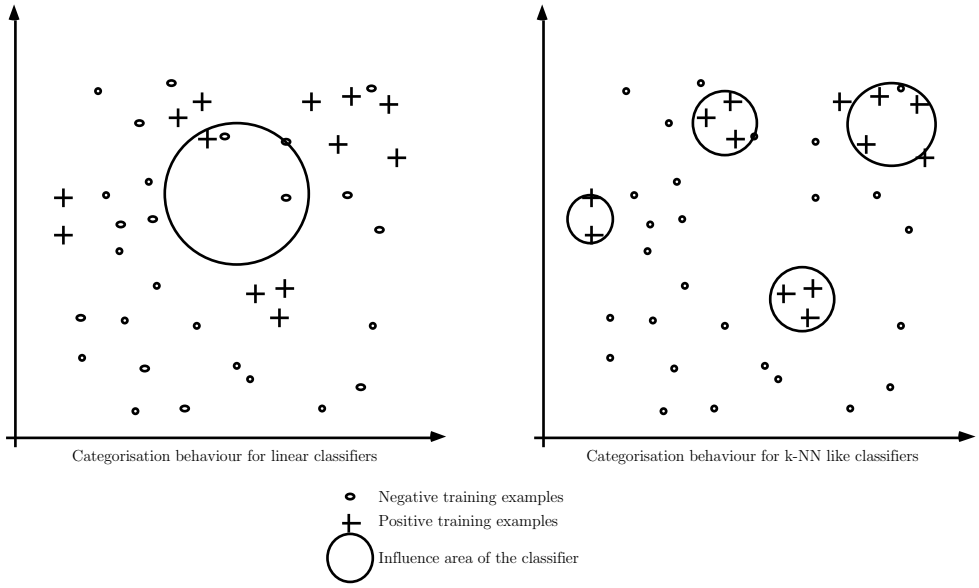

Figure 1: The categorisation behaviour of linear and non-linear classifiers.

5.5 Classifier committees

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

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

Different combination rules have been experimented with in the literature. The simplest possible rule is *majority voting* (MV), whereby the binary classification judgments obtained by the k classifiers are pooled together, and the classification decision that reaches the majority of $\frac{k+1}{2}$ votes is taken (k obviously needs to be an odd number) [28]. This method is particularly suited to the case in which the committee includes classifiers characterised by a binary decision function $CSV_i : D \rightarrow \{0, 1\}$. Another possible policy is *dynamic classifier selection* (DCS), whereby among committee $\{\Phi_1, \ldots, \Phi_k\}$ the classifier Φ_t that yields the best effectiveness on the l validation examples most similar to d_j is selected, and his judgment adopted by the committee [28]. A still different policy, somehow intermediate between MV and DCS, is *adaptive classifier combination* (ACC), whereby the judgments of *all* the classifiers in the committee are summed together, but their individual contribution is weighted by the effectiveness that they have shown on the l validation examples most similar to d_j [28].

Classifier committees have had mixed results in text categorisation so far. Li and Jain [28] have experimented with a committee formed of (various combinations of) a Naive 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 Naive 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). It has to be remarked, however, 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 Determining thresholds

There are various possible policies for determining the threshold τ_i discussed at the beginning of Section 5, also depending on the constraints imposed by the application.

One possible policy is *CSV thresholding* (also called *probability thresholding* in the case of probabilistic classifiers [21], or *Scut* [50]) [46]. In this case the threshold τ_i is a value of the CSV_i function. Lewis [21] considered using a fixed threshold τ equal for all c_i's, but noted that this might result in assigning *all* the test docs to category c_i while not even assigning a single test document to category c_j. He then considered using different thresholds τ_i for different categories c_i established by normalising probability estimates (following a suggestion from [29]). His experimental results did not show, however, a considerable difference in effectiveness between the two variants. Yang [50] uses different thresholds τ_i for the different categories c_i. Each threshold is optimised by testing different values for it on the validation set and choosing the value which yields the best value of the chosen effectiveness function.

A second, popular policy is *proportional thresholding* [21, 46] (also called *Pcut* in [50]). The aim of this policy is to set the threshold τ_i so that the test set generality $g_{Te}(c_i)$ of a category c_i is as close as possible to its training set generality $g_{Tr}(c_i)$. This idea encodes the quite sensible principle according to which both in training set and test set the same percentage of documents of the original set should be classified under c_i. One drawback of this thresholding policy is that, for obvious reasons, it does not lend itself to document-pivoted categorisation. Yang [50] proposes a still more refined version of this policy, i.e. one in which a factor x (equal for all c_i's) is multiplied to $g_{Tr}(c_i)$ to actually obtain $g_{Te}(c_i)$. Yang claims that this factor, whose value is to be empirically determined by experimentation on a validation set, allows a smoother trade-off between recall and precision to be obtained (see Section 7.1.1). For both k-NN and LLSF she found that optimal values lie in the [1.2, 1.3] range.

Sometimes, depending on the application, a *fixed thresholding* policy (also known as "k-per-doc" thresholding [21] or *Rcut* [50]) 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. Strictly speaking, however, this is not a thresholding policy in the sense defined at the beginning of Section 5, as it might happen that d' is categorised under c_i, d'' is not, and $CSV_i(d') < CSV_i(d'')$. Quite clearly, this policy is mostly at home with document-pivoted categorisation. It suffers, however, from a certain coarseness, as the fact that k is equal for all documents (nor could this be otherwise) does not allow system fine-tuning.

In terms of experimental results, Lewis [21] found the proportional policy to be definitely superior to CSV thresholding when microaveraged effectiveness was tested but slightly inferior when using macroaveraging (see Section 7.1.1). Yang [50] found instead CSV thresholding to be superior to proportional thresholding (possibly due to her category-specific optimisation on a validation set), and found fixed thresholding to be consistently inferior to the other two policies. Of course, the fact that these results have been obtained across different classifiers no doubt reinforce them. In general, aside from the considerations above, the choice of the thresholding policy may also be influenced by the application; for instance, in applying a text classifier to document indexing for Boolean systems, a fixed thresholding policy might be chosen, while a proportional or CSV thresholding method might be chosen for Web page classification under Yahoo!-like catalogues.

7 Evaluation issues for document categorisation

As in the case of information retrieval systems, the evaluation of document classifiers is typically conducted *experimentally*, rather than analytically. The reason for this tendency is that, in order to evaluate a system analytically (e.g. proving that the system is correct and complete) we always 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 document classification (namely, that of relevance of a document to a category) is, due to its subjective character, inherently non-formalisable.

The experimental evaluation of classifiers, rather than concentrating on issues of efficiency, usually tries to evaluate the *effectiveness* of a classifier, i.e. its capability of taking the *right* categorisation decisions. The main reasons for this bias are that:

- efficiency is a notion dependent on the hw/sw technology used. Once this technology evolves, the results of experiments aimed at establishing efficiency are no longer valid. This does not happen for effectiveness, as any experiment aimed at measuring effectiveness can be replicated, with identical results, on any different or future hw/sw platform;
- effectiveness is really a measure of how the system is good at tackling the central notion of classification, that of relevance of a document to a category.

7.1 Measures of categorisation effectiveness

7.1.1 Precision and recall

Classification effectiveness is measured in terms of the classic IR notions of precision (Pr) and recall (Re), adapted to the case of document categorisation. *Precision wrt* c_i (Pr_i) is defined as the conditional probability $P(ca_{ix} = 1 \mid a_{ix} = 1)$, i.e. as the probability that if a random document d_x is categorised under c_i, this decision is correct. Analogously, *recall wrt* c_i (Re_i) is defined as the conditional probability $P(a_{ix} = 1 \mid ca_{ix} = 1)$, i.e. as the probability that, if a random document d_x should be categorised under c_i, this decision is taken. These category-relative values may be averaged, in a way to be discussed shortly, to obtain Pr and Re, i.e. values global to the whole category set. Borrowing terminology from logic, Pr may be viewed as the "degree of soundness" of the classifier wrt the given category set C, while Re may be viewed as its "degree of completeness" wrt C.

As they are defined here, Pr_i and Re_i (and consequently Pr and Re) are to be understood, in the line of [48], 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) 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) are defined accordingly. Precision wrt c_i and recall wrt c_i may thus be estimated as

$$Pr_i \stackrel{est}{=} \frac{TP_i}{TP_i + FP_i} \tag{5}$$

$$Re_i \stackrel{est}{=} \frac{TP_i}{TP_i + FN_i} \tag{6}$$

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

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

$$Pr^{\mu} \stackrel{est}{=} \frac{TP}{TP + FP} \tag{7}$$

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

Table 2: The contingency table for category c_i.

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

Table 3: The global contingency table.

$$Re^{\mu} \stackrel{est}{=} \frac{TP}{TP + FN} = \frac{\sum_{i=1}^{m} TP_i}{\sum_{i=1}^{m} (TP_i + FN_i)} \tag{8}$$

where the "μ" superscript stands for microaveraging. For this, the "global" contingency table of Table 3, obtained by summing over all category-specific contingency tables, is needed.

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

$$Pr^{M} \stackrel{est}{=} \frac{\sum_{i=1}^{m} Pr_i}{m} \tag{9}$$

$$Re^{M} \stackrel{est}{=} \frac{\sum_{i=1}^{m} Re_i}{m} \tag{10}$$

where the "M" superscript stands for macroaveraging.

It is important to recognise that these two methods may give quite different results, especially if the different categories are unevenly populated: for instance, if the classifier performs well on categories with a small number of positive test instances, its effectiveness will probably be better according to macroaveraging than according to microaveraging. There is no 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" [46, page 327] and thus favour macroaveraging, while others believe that topics should indeed count proportionally to their frequence, and thus lean towards microaveraging. From now on, we will assume that microaveraging is used, and will thus drop the "i" subscript from Pr, Re and other symbols; it should be clear, however, that everything we will say in the rest of Section 7 may be adapted to the case of macroaveraging in the obvious way.

7.1.2 Combined measures

Neither precision nor recall make sense in isolation of the other. In fact, in order to obtain a classifier with 100% recall, one would only need to set every threshold τ_i to 0, thereby obtaining the *trivial acceptor* (i.e. the classifier that sets $a_{ij} = 1$ for all $1 \leq i \leq m, 1 \leq j \leq n$, i.e. that classifies all documents under all categories). Quite obviously, in this case precision would usually be very low (more precisely, equal to *apc*, the average percentage of categories per test document).

Conversely, it is well-known from everyday information retrieval practice that higher levels of precision may be obtained at the price of a low recall.

In practice, by tuning thresholds τ_i a classification algorithm is tuned so as to improve Pr to the detriment of Re or viceversa. A classifier should thus be measured by means of a "combined" effectiveness measure which both Pr and Re concur to determine. Various such measures have been proposed, among which the following are most frequent:

- effectiveness is computed as *(interpolated) 11-point average precision*. That is, each threshold τ_i is successively set to the values for which recall takes up values of 0.0, 0.1, ..., 0.9, 1.0; for these 11 different thresholds precision is computed and averaged over the 11 resulting values. This methodology is completely analogous to the standard evaluation methodology for IR systems, and may be used
  - with categories being used in place of queries. This is most frequently used in the case of document-pivoted categorisation (see e.g [43, 49, 50, 53]);
  - with documents being used in place of queries and categories in place of documents. This is most frequently used in the case of category-pivoted categorisation (see e.g. [46, Section 6.1]. Note that in this case if macroaveraging is used, it needs to be redefined on a per-document, rather than per-category, basis.
- effectiveness is computed as the *breakeven* point, i.e. the value at which Pr equals Re (e.g. [1, 21, 26]). In all reasonable classifiers a value for each τ_i for which Pr and Re are (almost) equal does exist, since by increasing τ_i from 0 to 1 Pr increases monotonically and Re decreases monotonically. If for no value of τ_i Pr and Re are exactly equal, τ_i is set to the value for which Pr and Re are closest, and an *interpolated breakeven* is computed as the average of the the values of Pr and Re. As noted in [50], when for no value of τ_i Pr and Re are close enough, interpolated breakeven may not be a reliable indicator of the effectiveness of the classifier;
- effectiveness is computed as the value of the F_α *function*, for some $0 \leq \alpha \leq 1$ (e.g. [23]), i.e.

$$F_\alpha = \frac{1}{\alpha\frac{1}{Pr} + (1-\alpha)\frac{1}{Re}}$$

  In this formula α may be seen as the relative degree of importance attributed to Pr and Re: if $\alpha = 1$, then F_α coincides with Pr, if $\alpha = 0$ then F_α coincides with Re. Usually, a value of $\alpha = 0.5$ is used, which attributes equal importance to Pr and Re; for reasons we do not want to enter here, rather than $F_{0.5}$ this is usually called called F_1 (see [23] for details). As shown in [50], for a given classifier Φ, its breakeven value is always less or equal than its F_1 value.

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

7.2 Test collections

For experimentation purposes, standard *test collections* (playing a role akin to the test collections of IR) are available in the public domain. Typical examples include

- the REUTERS-21578 corpus, consisting of a revised version of an older corpus known as REUTERS-22173 [22]. The documents are newswire stories covering the period between 1987 and 1991;
- the OHSUMED corpus [16]. 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 [35].

These two test collections (especially REUTERS-21578 together with its older version) account for most of the experimentation that has been performed to date in text categorisation. Unfortunately, this does not mean that we have definitive results concerning how well a given classifier compares to another, in the sense that many of these experiments have been carried out in sometimes subtly different experimental conditions. In fact, at least six different versions (including REUTERS-21578) of the REUTERS-22173 collection have been carved out of the original and used for experimentation.

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

1. the same collection (i.e. same documents and same categories) is used for both classifiers;
2. the same choice ("split") of training set and test set is made for both classifiers;
3. the same effectiveness measure is used for both classifiers.

Unfortunately, much of earlier experimentation (at least until 1997) was not performed with this *caveat* in mind, which means that the results reported by these researches are seldom directly comparable. By experimenting three different classifiers on five versions of REUTERS-22173, Yang [52] has experimentally shown that a lack of compliance with these three conditions (and with Condition 1 in particular) may significantly influence the experimental results. In particular, [52] shows that experiments carried out on Version 2 are not directly comparable with those using later versions, since the former includes a significant percentage (58%) of "unlabelled" test documents which, being negative examples of all categories, tend to depress effectiveness. Many of these documents appear to have simply not been considered by the human categorisers, rather than having consciously been deemed not to belong to any category; Version 2 is thus simply not a good testbed for automatic classifiers.

7.3 Which classifier is best?

In what may be regarded as the first large-scale cross-experimental evaluation in the text categorisation literature, Yang [52] has been able to obtain indications on the relative performance of the various methods described in the literature. This was achieved by using either

- *direct comparison*: classifiers C' and C'' are experimented on the same test collection TC by using a common evaluation measure;
- *indirect comparison*:
  1. classifier C' is tested on test collection TC' and classifier C'' is tested on test collection TC'';
  2. one or more "baseline" classifiers $\overline{C}_1, \ldots, \overline{C}_m$ are tested on both TC' and TC''.

  Test 2 can give an indication of the relative "hardness" of the two collections; using both this indication and the results from Test 1 yields an indication on the relative value of the two classifiers.

The results reported in this evaluation are illustrated in Table 4. A number of interesting conclusions can be drawn from them (for more detailed discussions on these results see [50]):

System	Type of Method	Reuters 1	Reuters 2	Reuters 3	Reuters 4
WORD	(non-learning)		.15 (Scut)	.31 (Pcut)	.29 (Pcut)
k-NN	instance-based		.69 (Scut)	**.85 (Scut)**	**.82 (Scut)**
LLSF	regression-based			**.85 (Scut)**	.81 (Scut)
NNETS.PARC	neural networks				**.82 (Scut)**
CLASSI	neural networks			.80	
RIPPER	rule learning		.72 (Scut)	.80 (Scut)	
SWAP-1	rule learning			.79	
IND	decision trees		.67 (Pcut)		
C4.5	decision trees			.79 (F_1)	
CHARADE	rule learning			.78	
SLEEPING EXPERTS	rule learning		**.75 (Scut)**	.76 (Scut)	
ROCCHIO			.66 (Scut)	.75 (Scut)	
NAIVE BAYES	probabilistic		.65 (Pcut)	.71 (???)	
CONSTRUE	(non-learning)	**.90**			

Table 4: Comparative results among different classifiers (the entries indicate the microaveraged breakeven point and the thresholding policy used, with "Scut" denoting CSV thresholding and "Pcut" denoting proportional thresholding; **boldface** indicates the best performer on the collection). The results have been obtained by Yang [50], from both new experiments (first three rows of the table) and previously published results (rows from fourth to last), by direct and indirect comparison.

- The highest performance reported in the literature belongs to CONSTRUE, a manually constructed classifier. Unfortunately, this classifier has not been tested on other more standard benchmarks, and it is not clear [50] whether the (small) test set on which it has been experimented has been chosen randomly. As a consequence, the fact that this figure may be indicative of the performance of CONSTRUE has been convincingly questioned [50];
- Aside from CONSTRUE, the best performing methods appear to be nearest-neighbours [49], regression [51], and neural networks [36, 46].
- Rocchio is not a good performer, scoring next to last. These results somehow confirm earlier results by Schutze et al. [43], who had found classifiers based on linear discriminant analysis, linear regression, and neural networks, to perform about 15% better than Rocchio.
- The worst classifier appears to be Naive Bayes [25]. These results are (with the exception of the neural networks approaches) confirmed by experimentation on at least two versions of the collection and, although clearly not definitive, may be considered at least indicative of the relative value of these approaches. SLEEPING EXPERTS is the best performer on Version 2, but performs worse than most other classifiers on Version 3 (a more "reliable" collection than Version 2); this is surprising, to the point that the plausibility of these results has been questioned [50].
- By far the lowest performance is displayed by WORD, a "mock" classifier implemented by Yang [50] and not including any learning component[7]. This shows that machine learning techniques are definitely the way to go for automatic text categorisation.

It is however important to bear in mind that these values are not absolute and final judgments (if there may be any) on the comparative effectiveness of these classifiers. The main reason is

[7] WORD is based on the comparison between documents and category names, each treated as a vector of weighted terms in the vector space model. WORD was implemented by Yang with the only purpose of determining the difference in effectiveness that adding a learning component to a classifier brings about. WORD is actually called STR in [49].

that comparisons tend to be pretty reliable when they concern new experiments, performed by one author under carefully controlled conditions. However, they are more problematic when they involve different classifiers tested as part of different experiments, which is very much the case when comparing results published by different people. As pointed out in [50], in this case various factors, 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.) and indexing, or 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 detailed in published papers. One can only subscribe to the common belief that more experimentation on truly common benchmarks needed in order to arrive at more definitive conclusions.

References

[1] C. Apté, F. Damerau, and S. Weiss. Automated learning of decision rules for text categorization. *ACM Transactions on Information Systems*, 12(3):233–251, 1994.

[2] G. Attardi, A. Gullí, and F. Sebastiani. Automatic Web page categorization by link and context analysis. In C. Hutchison and G. Lanzarone, editors, *Proceedings of THAI'99, European Symposium on Telematics, Hypermedia and Artificial Intelligence*, pages 105–119, Varese, IT, 1999.

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

[4] P. Biebricher, N. Fuhr, G. Lustig, and M. Schwantner. 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*, pages 333–342, Grenoble, FR, 1988. Also reprinted in [45], pp. 513–517.

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

[6] W. W. Cohen and Y. Singer. Context-sensitive learning for text categorization. In *Proceedings of SIGIR-96, 19th ACM International Conference on Research and Development in Information Retrieval*, pages 307–315, Zürich, CH, 1996.

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

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

[9] J. L. Fagan. *Experiments in automatic phrase indexing for document retrieval: a comparison of syntactic and non-syntactic methods*. PhD thesis, Department of Computer Science, Cornell University, Ithaca, US, 1987.

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

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

[12] N. Fuhr, S. Hartmann, G. Knorz, G. Lustig, M. Schwantner, and K. Tzeras. 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"*, pages 606–623, Barcelona, ES, 1991.

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

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

[15] P. J. Hayes and S. P. Weinstein. CONSTRUE/TIS: a system for content-based indexing of a database of news stories. In *Proceedings of IAAI-90, 2nd Conference on Innovative Applications of Artificial Intelligence*, pages 1–5, 1990.

[16] W. Hersh, C. Buckley, T. Leone, and D. Hickman. 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*, pages 192–201, Dublin, IE, 1994.

[17] D. A. Hull. 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*, pages 282–289, Dublin, IE, 1994.

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

[19] W. Lam and C. Y. Ho. 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*, pages 81–89, Melbourne, AU, 1998.

[20] M. E. Lesk. Automatic word sense disambiguation: how to tell a pine cone from an ice cream cone. In V. DeBuys, editor, *Proceedings of SIGDOC-86, 5th ACM International Conference on Systems Documentation*, pages 24–26, New York, US, 1986.

[21] D. D. Lewis. 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*, pages 37–50, Kobenhavn, DK, 1992.

[22] D. D. Lewis. *Representation and learning in information retrieval*. PhD thesis, Department of Computer Science, University of Massachusetts, Amherst, US, 1992.

[23] D. D. Lewis. Evaluating and optmizing autonomous text classification systems. In *Proceedings of SIGIR-95, 18th ACM International Conference on Research and Development in Information Retrieval*, pages 246–254, Seattle, US, 1995.

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

[25] D. D. Lewis and W. A. Gale. A sequential algorithm for training text classifiers. In *Proceedings of SIGIR-94, 17th ACM International Conference on Research and Development in Information Retrieval*, pages 3–12, Dublin, IE, 1994.

[26] D. D. Lewis and M. Ringuette. A comparison of two learning algorithms for text categorization. In *Proceedings of SDAIR-94, 3rd Annual Symposium on Document Analysis and Information Retrieval*, pages 81–93, Las Vegas, US, 1994.

[27] D. D. Lewis, R. E. Schapire, J. P. Callan, and R. Papka. Training algorithms for linear text classifiers. In *Proceedings of SIGIR-96, 19th ACM International Conference on Research and Development in Information Retrieval*, pages 298–306, Zürich, CH, 1996.

[28] Y. H. Li and A. K. Jain. Classification of text documents. *The Computer Journal*, 41(8):537–546, 1998.

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

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

[31] T. Mitchell. *Machine learning*. McGraw Hill, New York, US, 1996.

[32] D. Mladenić. Turning YAHOO! into an automatic Web page classifier. In *Proceedings of ECAI'98, 13th European Conference on Artificial Intelligence*, pages 473–474, Brighton, UK, 1998.

[33] D. Mladenić and M. Grobelnik. Feature selection for classification based on text hierarchy. In *Proceedings of the Workshop on Learning from Text and the Web*, Pittsburgh, USA, 1998.

[34] National Aeronautics and Space Administration, Washington, US. *NASA thesaurus*, 1994 edition. Accessible at `http://www.sti.nasa.gov/nasa-thesaurus.html`.

[35] National Library of Medicine, Bethesda, US. *Medical Subject Headings (MeSH)*, 1993 edition.

[36] H. T. Ng, W. B. Goh, and K. L. Low. 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*, pages 67–73, Philadelphia, US, 1997.

[37] H. Ragas and C. H. Koster. 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*, pages 369–370, Melbourne, AU, 1998.

[38] S. E. Robertson and K. Sparck Jones. Relevance weighting of search terms. *Journal of the American Society for Information Science*, 27(3):129–146, 1976. Also reprinted in [47], pp. 143–160.

[39] G. Salton and C. Buckley. Term-weighting approaches in automatic text retrieval. *Information Processing and Management*, 24(5):513–523, 1988. Also reprinted in [45], pp. 323–328.

[40] G. Salton, A. Wong, and C. Yang. A vector space model for automatic indexing. *Communications of the ACM*, 18(11):613–620, 1975. Also reprinted in [45], pp. 273–280.

[41] T. Saracevic. 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, 1975. Also reprinted in [45], pp. 143–165.

[42] H. Schütze. Automatic word sense discrimination. *Computational Linguistics*, 24(1):97–124, 1998.

[43] H. Schütze, D. A. Hull, and J. O. Pedersen. 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*, pages 229–237, Seattle, US, 1995.

[44] A. F. Smeaton. Using NLP or NLP resources for information retrieval tasks. In T. Strzalkowski, editor, *Natural language information retrieval*. Kluwer Academic Publishers, Dordrecht, NL, 1997.

[45] K. Sparck Jones and P. Willett, editors. *Readings in information retrieval*. Morgan Kaufmann, San Mateo, US, 1997.

[46] E. Wiener, J. O. Pedersen, and A. S. Weigend. A neural network approache to topic spotting. In *Proceedings of SDAIR-95, 4th Annual Symposium on Document Analysis and Information Retrieval*, pages 317–332, Las Vegas, US, 1995.

[47] P. Willett, editor. *Document retrieval systems*. Taylor Graham, London, UK, 1988.

[48] S. M. Wong and Y. Yao. On modeling information retrieval with probabilistic inference. *ACM Transactions on Information Systems*, 13(1):38–68, 1995.

[49] Y. Yang. 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*, pages 13–22, Dublin, IE, 1994.

[50] Y. Yang. An evaluation of statistical approaches to text categorization. *Information retrieval*, 1-2(1):69–90, 1999.

[51] Y. Yang and C. G. Chute. An example-based mapping method for text categorization and retrieval. *ACM Transactions on Information Systems*, 12(3):252–277, 1994.

[52] Y. Yang and X. Liu. 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. Forthcoming.

[53] Y. Yang and J. O. Pedersen. A comparative study on feature selection in text categorization. In *Proceedings of ICML-97, 14th International Conference on Machine Learning*, pages 412–420, Nashville, US, 1997.