An Evaluation of Statistical Approaches to Text Categorization *

Yiming Yang (yiming@cs.cmu.edu, http://www.cs.cmu.edu/˜yiming)
School of Computer Science, Carnegie Mellon University

Abstract.
This paper focuses on a comparative evaluation of a wide-range of text categorization methods, including previously published results on the Reuters corpus and new results of additional experiments. A controlled study using three classifiers, kNN, LLSF and WORD, was conducted to examine the impact of configuration variations in five versions of Reuters on the observed performance of classifiers. Analysis and empirical evidence suggest that the evaluation results on some versions of Reuters were significantly affected by the inclusion of a large portion of unlabelled documents, mading those results difficult to interpret and leading to considerable confusions in the literature. Using the results evaluated on the other versions of Reuters which exclude the unlabelled documents, the performance of twelve methods are compared directly or indirectly. For indirect compararions, kNN, LLSF and WORD were used as baselines, since they were evaluated on all versions of Reuters that exclude the unlabelled documents. As a global observation, kNN, LLSF and a neural network method had the best performance; except for a Naive Bayes approach, the other learning algorithms also performed relatively well.

Keywords: Text Categorization, Statistical Learning Algorithms, Comparative Study, Evaluation

* This research was supported in part by NIH grant LM-05714 and by NSF grant IRI9314992.

1. Introduction

Text categorization (TC) is the problem of assigning predefined categories to free text documents. A growing number of statistical learning methods have been applied to this problem in recent years, including regression models[5, 26], nearest neighbor classifiers[4, 22], Bayesian probabilistic classifiers [19, 10, 12], decision trees[5, 10, 12], inductive rule learning algorithms[1, 3, 13], neural networks[21, 14] and on-line learning approaches[3, 9]. With more and more methods available, cross-method evaluation becomes increasingly important to identify the state-of-the-art in text categorization. However, without a unified methodology in empirical evaluations, objective comparisons of different methods are difficult.

Ideally, all researchers would like to use a common collection and comparable performance measures to evaluate their systems, or would allow their systems to be evaluated under carefully-controlled conditions in a fashion similar to that of the Text Retrieval Conference (TREC). The reality, however, is far from the ideal. Cross-method comparisons have been attempted in the literature, but often only for two or three methods. The small scale of these comparisons could either lead to overly-general statements based on insufficient observations, or provide little insight into a global comparison between a wide range of approaches. An alternative to these small-scale comparisons would be to integrate the available results of categorization methods into a global evaluation, carefully analyzing the test conditions and evaluation measures used and establishing a common basis for cross-collection and cross-experiment integration. This solution would lead to a TREC-like controlled evaluation for text categorization, as well as contribute useful insights to individual studies. This paper is an effort in that direction.

The most serious problem in TC evaluations is the lack of standard data collections. Even if a common collection is chosen, there are still many ways to introduce inconsistent variations. The commonly used Reuters news story corpus, for example, has at least five different versions, depending on how the training/test sets are divided, which subsets of categories or documents were used or not used for evaluation, and so forth. The number of different configurations of this corpus is still growing. It is often unclear whether or not the reported results on the different versions of Reuters are comparable to each other. In this paper we examine the impact of corpus configuration variations on the performance of classifiers, using a carefully-controlled experiments of several categorization systems on five different versions of Reuters. As will be shown in Section 5.2, variations between certain versions of Reuters *do* have a strong impact, while the variations between other versions do not. The underlying reason for this will be analyzed.

Another important issue in cross-experiment evaluation is the comparability between different performance measures used in individual experiments. Many measures have been used, including recall and precision, accuracy or error, *break-even point* or *F-measure*, *micro-average* and *macro-average* for binary categorization, *11-point average precision* for category ranking, and so forth (see Section 3 for definitions). Each of these measures is designed to evaluate some aspect of the categorization performance of a system; however, none of them convey identical information. Which of these measures are more suitable for text categorization? How can published results of text categorization methods be best compared when they were evaluated using different performance measures? These questions are addressed in this paper by applying a variety of performance measures to several classifiers, including the measures for category ranking evaluation or the measures for binary category assignment. We will show that both types of measures are informative and complementary to each other. We will also show that with carefully chosen performance measures and a baseline classifier, one can reasonably (indirectly) compare the relevant performance among classifiers across experiments based on their relevant performance with respect to the baseline classifier.

This paper is divided into six sections in addition to the introduction. Section 2 describes the classifiers and the Reuters corpus we will use in this paper. Section 3 introduces and analyzes performance measures for category ranking evaluation and binary categorization evaluation. Section 4 describes the novel experiments we conducted with WORD, kNN, and LLSF. Section 5 reports the results of our classifiers and evaluates them together with published results of other classifiers. Finally, we summarize our conclusions in Section 6.

2. Classifiers Evaluated on Reuters

2.1. Classifiers

We consider the text categorization systems whose results on the various versions of the Reuters corpus have been published in the literature[6, 10, 1, 21, 13, 3, 27, 14] [1]. In addition to these results, we present new results of three systems. These systems are briefly described below, grouped roughly according to their theoretical foundations or technical characteristics.

CONSTRUE is an expert system developed at the Carnegie Group, and the earliest system evaluated on the Reuters corpus[6]. Impressive results (about 90% in both recall and precision, on average) were reported on a small subset (3%) of this corpus. A major difference between the CONSTRUE approach and the other methods considered in this paper is the use of manually developed domain-specific or application-specific rules in the expert system. Adapting CONSTRUE to other application domains would be costly and labor-intensive.

Decision Tree (DTree) is a well-known machine learning approach to automatic induction of classification trees based on training data[16, 11]. Applied to text categorization, DTree algorithms are used to select informative words based on an information gain criterion, and predict categories of each document according to the occurrence of word combinations in the document. Evaluation result of DTree algorithms on the Reuters text categorization collection were reported by Lewis & Ringuette (using the IND package)[10] and Moulinier (using C4.5)[12], respectively.

Naive Bayes (NaiveBayes) probabilistic classifiers are also commonly-used in text categorization[11]. The basic idea is to use the joint probabilities of words and categories to estimate the probabilities of categories given a document. The naive part of such a model is the assumption of word independence. The simplicity of this assumption makes the computation of the NaiveBayes classifier far more efficient than the exponential complexity of non-naive Bayes approaches because it does not use word combinations as predictors. Evaluation results of NaiveBayes on Reuters were reported by Lewis & Ringuette[10] and Moulinier[12], respectively.

Inductive rule learning in Disjunctive Normal Form (DNF) was tested in the WASP-1, RIPPER and CHARADE systems[1, 13, 3]. DNF rules are of equal power to DTrees in machine learning theory[11]. Empirical results for the comparison between DNF and DTree approaches, however, are rarely available in text categorization, except in an indirect comparison by Apte et al.[1]

Neural network (NNet) approaches to text categorization were evaluated on Reuters by Wiener et al. [21] and Ng. et al. [14], respectively. For convenience, the former system (developed at Xerox PARC) is referred to as NNet.PARC in this paper; the latter system is named CLASSI. Both systems use a separate neural network per category, learning a non-linear mapping from input words (or more complex features such as singular vectors of a document space) to a category. The PARC group tried a perceptron approach and three-layered neural networks in addition. The results of the three-layered neural networks, however, are available only on a subset of the Reuters categories which are common in evaluations of other systems. The CLASSI system only uses perceptrons.

Rocchio is a classic vector-space-model method for document routing or filtering in information retrieval. Applying it to text categorization, the basic idea is to construct a prototype vector per category using a

[1] This paper focuses on the pre-1997 versions of the Reuters collection which have been commonly used in the published evaluations, but not on the newly refined version (Reuters-21578) on which evaluation results are not widely available yet.

training set of documents. Given a category, the vectors of documents belonging to this category are given a positive weight, and the vectors of remaining documents are given a negative weight. By summing up these positively and negatively weighted vectors, the prototype vector of this category is obtained. This method is easy to implement and efficient in computation, and has been used as a baseline in several evaluations[9, 3]. A potential weakness of this method is the assumption of one centroid per category, and consequently, Rocchio does not perform well when the documents belonging to a category naturally form separate clusters.

LLSF stands for Linear Least Squares Fit, a mapping approach developed by Yang[25]. A multivariate regression model is automatically learned from a training set of documents and their categories. The training data are represented in the form of input/output vector pairs where the input vector is a document in the conventional vector space model (consisting of words with weights), and output vector consists of categories (with binary weights) of the corresponding document. By solving a linear least-squares fit on the training pairs of vectors, one can obtain a matrix of word-category regression coefficients. The matrix defines a mapping from an arbitrary document to a vector of weighted categories. By sorting these category weights, a ranked list of categories is obtained for the input document.

Sleeping Experts (EXPERTS) are on-line learning algorithms recently applied to text categorization[3]. On-line learning aims to reduce the computation complexity of the training phase for large applications. EXPERTS updates the weights of n-gram phrases incrementally.

kNN stands for k-nearest neighbor classification. Given an arbitrary input document, the system ranks its nearest neighbors among the training documents, and uses the categories of the k top-ranking neighbors to predict the categories of the input document. The similarity score of each neighbor document to the new document being classified is used as the weight of each of its categories, and the sum of category weights over the k nearest neighbors are used for category ranking.

WORD is a simple, non-learning algorithm which ranks categories for a document based on word matching between the document and category names. The purpose of testing such a simple method is to quantitatively measure how much an improvement is obtained by using statistical learning compared to a non-learning approach. The conventional vector space model is used for representing documents and category names (each name is treated as a bag of words), and the SMART system [17] is used as the search engine.

These classifiers can be divided into two types: independent binary classifiers or m-ary ($m > 2$) classifiers. Given a document, an independent binary classifier makes a YES/NO decision for each category, independently from its decisions on other categories. Of the classifiers listed above, CONSTRUE, DTree, NaiveBayes, DNF, NNet.PARC, CLASSI, Rocchio, and EXPERTS are independent binary classifiers. An m-ary classifier, on the other hand, typically uses a shared classifier for all categories, producing a ranked list of candidate categories for each test document, with a confidence score for each candidate; per-category binary decisions can be obtained (if desired) by thresholding on the ranks or scores of candidate categories. Of the classifiers listed above, kNN, LLSF and WORD are primarily m-ary classifiers.

There are several algorithmic solutions for converting m-ary classification output to per-category binary decisions, which will be described in Section 4. It is also possible, in principle, to use independent binary classifiers to produce category ranking if the binary decisions have confidence scores, and if the confidence scores can be meaningfully compared for category ranking. How to effectively combine the output of binary classifiers, however, is not well-understood at this stage of research.

Table I. Examination on kNN, LLSF and WORD on different versions of Reuters

Version	(prepared by)	UniqCate	TrainDocs	TestDocs	(Labelled TestDocs)
Version 1	(CGI)	182	21,450	723	(80%)
Version 2	(Lewis)	113	14,704	6,746	(42%)
Version 2.2	(Yang)	113	7,789	3,309	(100%)
Version 3	(Apte)	93	7,789	3,309	(100%)
Version 4	(PARC)	93	9,610	3,662	(100%)

2.2. The Reuters Corpus

We use the Reuters collection for our cross-method comparisons because it is the most commonly-used collection for text categorization evaluation in the literature. The Reuters corpus consists of over 20,000 Reuters newswire stories in the period between 1987 to 1991. The original corpus (Reuters-22173) was provided by the Carnegie Group, Inc. (CGI) and used to evaluate their CONSTRUE system in 1990[6]. Several versions have been derived from this corpus since them by varying the documents in the corpus, the division between the training and test set, and the categories used for evaluation. Table I summarizes these versions.

Reuters version 2 (also called Reuters-21450), prepared by Lewis et al.[10], contains all of the documents in the original corpus (version 1) except the 723 test documents. The documents are split into two chronologically contiguous chunks; the early one is used for training, and the later one for testing. A subset of 113 categories were chosen for evaluation. One peculiarity of Reuters-22450 is the inclusion of a large portion of unlabeled documents in both the training (47%) and test (58%) test sets. It is observed by this author that on randomly tested documents, in many cases, the documents do belong to one of those 113 categories but happen to be unlabelled[2]. However, it is not known exactly how many of the unlabeled documents should be labelled with a category.

To facilitate an evaluation of the impact of these unlabeled documents on text categorization evaluation, we created a new corpus from Reuters 2, called Reuters 2.2. Reuters 2.2 is the same as Reuters 2, except that all of the unlabeled documents have been removed.

Reuters version 3 was constructed by Apte et al. for their evaluation of the SWAP-1 by removing all of the unlabeled documents from the training and test sets and restricting the categories to have a training-set frequency of at least two[1][3].

Reuters version 4 was constructed by the research group at Xerox PARC, and was used for their evaluation of their neural network approaches[21]. This version was drawn from Reuters version 1 by eliminating the unlabeled documents and some rare categories. Instead of taking continuous chunks of documents for training and testing, it slices the collection into many small chunks that do not overlap temporally. These subsets are numbered, and the odd-numbered chunks are used for training and the even subsets are used for testing[4].

[2] A personal contact in the Carnegie Group confirmed that Reuters does not always categorize all of their news stories.

[3] A formatted version of this collection was prepared by Y. Yang and colleagues, and is currently available at Carnegie Mellon University's web site through http://moscow.mt.cs.cmu.edu:8081/reuters_21450/apte.

[4] A formatted version of this collection was prepared by Y. Yang and colleagues, and is electronically available at http://moscow.mt.cs.cmu.edu:8081/reuters_21450/parc/.

3. Evaluation Measures

As discussed in Section 2, classifiers can be used for either category ranking or automated category assignments. What type of output is more preferable depends on applications. For user interaction, a category ranking system would be more useful, while for fully automated categorization tasks, either type would be fine. Without assuming a particular application in mind, the ranking performance and binary classification performance would be informative for evaluating classifiers. In this paper, we evaluate the performance of kNN, LLSF and WORD in both types of usage; however, only the binary classification results are used in the comparison with other classifiers because those classifiers do not offer category ranking.

3.1. Performance Measures for Category Ranking

Category ranking can be evaluated using measures similar to the conventional measures for evaluating ranking-based document retrieval systems: recall, precision, and 11-point average precision. Given a classifier whose input is a document, and whose output is a ranked list of categories assigned to that document, the recall and precision can be computed at any threshold on this ranked list:

$$\text{recall} = \frac{\text{categories found and correct}}{\text{total categories correct}}$$

$$\text{precision} = \frac{\text{categories found and correct}}{\text{total categories found}}$$

where "categories found" means categories above the decision threshold. For the global evaluation of a classifier on a collection of test documents, we adapt the procedure for the conventional *interpolated* 11-point average precision[18], as described below:

- For each document, compute the recall and precision at each position in the ranked list where a correct category is found.
- For each interval between recall thresholds of 0%, 10%, 20%, ..., 100%, use the highest precision value in that interval as the "representative" precision value at the left boundary of this interval.
- For the recall threshold of 100%, the "representative" precision is either the exact precision value if such a data point exists, or the precision value at the closest point in terms of recall. If the interval is empty, use the default precision value of zero.
- *Interpolation*: At each of the above recall thresholds, replace the representative precision using the highest score among the representative precision values at this threshold and the higher thresholds.
- *Per-interval Averaging*: Average per-document data points over all the test documents, at each of the above recall thresholds respectively. This step results in 11 per-interval average precision scores.
- *Global Averaging*: Average of the per-interval average precision scores to obtain a single-numbered performance average ("11-pt AVGP").

3.2. Performance Measures for Binary Classifiers

The category assignments of a binary classifier can be evaluated using a two-way contingency table (Table II) for each category, which has four cells:
where

Table II. A contingency table

	YES is correct	No is correct
Assigned YES	a	b
Assigned NO	c	d

– cell a counts the documents correctly assigned to this category;

– cell b counts the documents incorrectly assigned to this category;

– cell c counts the documents incorrectly rejected from this category;

– cell d counts the documents correctly rejected from this category.

Conventional performance measures are defined and computed from these contingency tables. These measures are recall (r), precision (p), fallout (f), accuracy (Acc) and error (Err):

- $r = a/(a+c)$ if $a+c > 0$, otherwise undefined;
- $p = a/(a+b)$ if $a+b > 0$, otherwise undefined;
- $f = b/(b+d)$ if $b+d > 0$, otherwise undefined;
- $Acc = (a+d)/n$ where $n = a+b+c+d > 0$;
- $Err = (b+c)/n$ where $n = a+b+c+d > 0$.

For evaluating performance average across categories, there are two conventional methods, namely *macro-averaging* and *micro-averaging*. Macro-averaged performance scores are computed by first computing the scores for the per-category contingency tables and then averaging these per-category scores to compute the global means. Micro-averaged performance scores are computed by first creating a global contingency table whose cell values are the sums of the corresponding cells in the per-category contingency tables, and then use this global contingency table to compute the micro-averaged performance scores. There is an important distinction between macro-averaging and micro-averaging. Micro-average performance scores gives equal weight to every document, and is therefore considered a per-document average (more precisely, an average over all the document/category pairs). Likewise, macro-average performance scores give equal weight to every category, regardless of its frequency, and is therefore a per-category average.

3.3. Analysis of BEP and F-measure

Some of the performance measures may be misleading when examined alone. For example, a trivial algorithm that says YES to every category for any document will have a perfect recall of 100%, but an unacceptably low score in precision. Conversely, if a system rejects every document for every category, it will have a perfect score in precision and fall-out, but will sacrifice recall to the extreme. Usually, a classifier exhibits a trade-off between recall and precision when the internal parameters or decision threshold in the classifier are adjusted; to obtain a high recall usually means sacrificing precision and vice-versa. If the recall and precision of a classifier can be tuned to have an equal value, then this value is called the *break-even point* (BEP) of the system[10]. BEP has been commonly used in text categorization evaluations. If the recall and precision values cannot be made exactly equal, the average of the nearest recall and precision values is used as the *interpolated* BEP. A problem with the interpolation is that when the *nearest* recall and precision values are far apart, the BEP may not reflect the true behavior of the system.

The F_1 measure, defined by C.J. van Rijsbergen[20], is another common choice for a single-numbered performance measure:

$$F_1 = 2rp/(r+p).$$

It balances recall and precision in a way that gives them equal weight. A more general form of the F-measure is defined as:

$$F_\beta(r,p) = \frac{(\beta^2+1)pr}{\beta^2 p + r}$$

where β is the parameter allowing differential weighting of precision (p) and recall (r).

The F_1 measure is often used as an optimization criterion in threshold tuning for binary decisions. Its score is maximized when the values of recall and precision are equal or close; otherwise, the smaller of recall and precision dominates the value of F_1. It should be noticed that BEP is just a specific value of the F_1 variable. That is, when $r = p$ (at BEP by definition), $F_1 = 2r^2/2r = r = p$. This also means that the BEP score of a system is always equal or less than the optimal value of F_1 of that system. This restricts the kinds of comparisons one would make between a system whose performance is measured via F_1 and one whose performance is measured via the BEP. Specifically, a system whose BEP value is higher than another system's optimal F_1 value is definitely a better performer (when the balanced recall and precision is the main consideration), but the converse is not necessarily true.

3.4. Analysis of Accuracy and Error

Although accuracy and error are common performance measures in the machine learning literature and have been used in some evaluations of text categorizations systems, there is a potential pitfall in using them to train or evaluate a binary classifier. The following example illustrates this pitfall.

The Reuters collection version 3 has 93 categories, with each document having 1.2 categories assigned to it on average. This means that the average probability of a given document belonging to a given category is $1.2/93 = 1.3\%$. Consequently, a trivial algorithm that rejects every document for every category will have a global (micro or macro, they are equal) average error rate of 1.3% and a global average accuracy of 98.7%. This is not to suggest that a trivial rejector classifier is good, but that accuracy or error may not be a sensible measure of the effectiveness or usefulness of a classifier in text categorization when the number of categories is large and the average number of categories per document is small. This problem is further illustrated by the OHSUMED collection[7], which is another corpus commonly used in text categorization research. OHSUMED contains 233,445 documents indexed using 14,321 unique categories; there are about 13 categories per document on average. For this collection, the trivial rejector will have a global average error rate of 0.1% and an accuracy of 99.9%. If global average accuracy or error is used as the optimization criterion for training a categorization system, the system will tend to learn to trivially reject all documents for any category, since this will lead to a very high global accuracy or error rate.

Fundamentally, these difficulties in using accuracy and error as performance measures arises from their definitions. Unlike recall and precision, accuracy and error have n, the number of test documents in their divisor. Therefore a small change in the value of a (true positive) or d (true negative) will produce only a small change in the value of accuracy (likewise a small change in b or c will produce only a small change in the value of error). However, for rare categories the maximum value of a or c is small (neither can be larger than the number of documents that belong to the category in question). Consequently, a or c can range from zero to its maximum value without having much effect on the value of accuracy or error, respectively.

3.5. Comparative Analysis

Now consider the value of recall, defined as $a/(a+c)$; the potential values of a and c are both small, and furthermore the quantity $a+c$ is always constant and equal to the number of documents that belong to the category in question. Consequently, any change in the value of a will produce a relatively large change in the value of recall. Even the correct or incorrect classification of one document will produce a change of 1/(a+c), compared to the 1/n change in accuracy or error for the proper or erroneous classification of the same document. In other words, recall is a much more sensitive measure of performance for rare categories than accuracy or error.

For rare categories, precision and the F_1 measure will have a sensitivity in between that of accuracy and recall. Both precision and the F_1 measure have the value of b in the denominator, which can be quite large. However, if the classifier is performing well, then the value of b will be small, and precision and the F_1 measure will be sensitive to misclassification of documents; in fact, the larger these measures are, the more sensitive they are to errors in the classification process.

Fallout suffers from the same problem as accuracy and error; the value of $b+d$ in the denominator is a constant and large for rare categories, so fallout will not be as sensitive as recall, precision, and the F_1 measure for rare categories.

Thus, F_1 is a the most suitable choice among those measures discussed above when the expected number of categories per document is small compared to the number of categories recognized by the classifier; it is sensitive to classification errors, zero for a trivial rejection algorithm, and very small for a trivial acceptance algorithm. BEP, similar to F_1, would also be a good choice if it is the true BEP without interpolation, or if the interpolated BEP is sufficiently close to the actual recall and precision values. Accuracy and error, on the other hand, are the least suitable measures because they are insensitive to performance variances, and can return near-optimal values for the trivial rejector algorithm.

Finally, just because a particular measure may not be as good as another when considered by itself does not mean that measure is completely useless in evaluation. Accuracy and error can be insightful measures if coupled with one of the more sensitive performance scores, such as BEP or F_1. In general, evaluations should provide a variety of scores when measuring the performance of an algorithm instead of compressing performance into a single score.

4. Experimental Design

We have conducted novel experiments with the kNN, LLSF, and WORD classifiers to explore the cross-method evaluation problem from the following aspects:

– the effect of collection variability on the performance of classifiers;

– the effect of thresholding methods in converting category ranking to binary decisions;

– the sensitivity of performance measures in reflecting classifiers behavior; and

– the scalability of these classifiers in a larger/harder application, i.e. on the OHSUMED collection.

The kNN and WORD classifiers were applied to the five versions of the Reuters collection mentioned in Section 2. The LLSF method was only evaluated on three of the five versions because its computation is more costly than those in kNN and WORD.

4.1. Preprocessing and Feature Selection

The bench-marking retrieval system, SMART[17], is used as a unified preprocessor for the kNN, LLSF and WORD systems. It is used for removing stop words, stemming, and term weighting, where a *term* is a word after stemming. A phrasing option is available in SMART but has not been used in these experiments. The term-weighting schemes combine the within-document term frequency (TF) and the Inverted Document Frequency (IDF) in a variety of ways. Several typical term-weighting options were tested, including "ltc", "atc", "ntc", etc. in SMART's notation. The term-weighting which produced the best results ("ltc" in most cases) is used for cross-method comparisons.

Feature selection is the next step after stop words are removed from documents by SMART. Feature selection attempts to remove non-informative words from documents in order to improve categorization effectiveness and reduce computational complexity. The efficiency improvement from feature selection is especially important for LLSF, which would otherwise be computationally intractable (in the training phase) if applied to large collections. Tractability is not a crucial issue for kNN, but the improved performance from noise reduction by feature selection is still desirable.

Five feature selection criteria were tested with kNN and LLSF, including information gain, mutual exclusion, a χ^2 statistic, document frequency and term strength; a thorough evaluation of these feature selection methods was reported elsewhere[27]. The χ^2 statistic was found most effective, and information gain and document frequency yielded similar results. On the Reuters collection version 3, for example, the χ^2-based feature selection reduced the training set vocabulary from 24,858 unique words to 2,485. Correspondingly, the category ranking performance of kNN improved from 90% to 93% in 11-point average precision, and the category assignments by kNN improved from 0.81 to 0.85 in the F_1-measure. The best results of kNN and LLSF (with χ^2-based feature selection) are reported in the Results section. Aggressive vocabulary reduction was not applied to WORD because it would reduce the chance of matching words between category names and documents.

4.2. Thresholding Strategies for Binary Categorization

The kNN, LLSF, and WORD classifiers are primarily ranking systems. To obtain binary assignments of categories to documents, several thresholding strategies were examined, named the *Rcut, Pcut* and *Scut*.

Rcut stands for rank-based thresholding. Given a target space of m categories, the ranking system (kNN, LLSF or WORD) produces a ranking list of the m categories for each document. YES/NO decisions are obtained by thresholding on the ranks of candidate categories. The threshold is predetermined empirically, chosen from the integers of $1, 2, \ldots, m$. Performance scores (recall, precision, F_1, etc.) vary according to different thresholds. While simple and easily applied to on-line category assignment for a particular document, Rcut suffers from the inability to smoothly adjust the trade-off between recall and precision because of the discrete rank values. That is, given a test set of n documents, there are always n category candidates with the same rank, so they will be be all assigned "YES" or "NO". As a result, the optimization of the trade-off, or the optimal value of F_1, is difficult to obtain.

Pcut is an abbreviation of *proportional assignment*, a method which has been used in previous text categorization research[10, 21]. Before any binary decision is made, the classifier produces confidence scores for all the $n \times m$ document-category decision pairs, where n is the number of test documents and m is the number of training-set categories. Then the decision pairs are sorted by category, resulting in a ranked list of decisions for each category. Binary decision are obtained by assigning "YES" to each of the $x \times n \times P_i$ top-ranking decision pairs in a list, and "NO" to the remaining pairs, where n is the number of documents in the test set, P_i is the training-set probability of the ith category (i in $1 \ldots m$), and x is an empirically chosen parameter. By varying the value of x, a smooth trade-off between recall and precision can be obtained in general. Pcut can effectively adjust the system's bias if its assignments are far out of proportion, i.e. too many YES assignments for some categories but too little YES assignments for others. However, the decision thresholds are only based on the training-set probabilities of categories; they are not learned or optimized using the ranking or confidence scores of the system. In addition, Pcut suffers from the inability to perform on-line category assignment to a document; that is, classifiers using Pcut cannot provide on-line assistance to users in computer-aided text categorization.

Scut stands for optimal thresholding on the confidence scores of category candidates. This is done by splitting the original training set into two portions, and using one portion for training and another portion ("the validation test set") for learning the optimal threshold for each category. Given a category, the optimal threshold is the score which optimizes the F_1 value of the system on the validation set of documents. Optimal thresholds can be learned off-line; once the optimal thresholds are learned, they can be used for on-line category assignment.

All three thresholding methods were tested with kNN, LLSF and WORD on Reuters version 3; the results will be presented in the Results section.

4.3. Parameter Optimization

Parameters empirically determined in LLSF, kNN and WORD include:

- TW, the term weighting scheme which is a choice between atc, ltc, ntc etc.;
- FT, the number of features selected from the vocabulary of the original of training documents;
- TR, the maximum length of the ranked list (i.e., any category ranked beyond the value of TR will have a confidence score of zero);
- *rank*, the threshold for making YES/NO decisions in Rcut;
- x, the average number of system-assigned YES decisions per document Pcut;
- the k value of kNN, which is the number of the nearest neighbors used for category prediction given a test document;
- the p value in LLSF, i.e., the number of singular vectors to use in computing an approximated LLSF regression model.

The last two, k and p, are classifier-specific parameters. Thorough investigations on suitable choices of these parameter values were reported in previous papers where the main observations were that the performance of kNN is relatively stable for a large range of k values[22], and that satisfactory performance of LLSF depends on whether p is sufficiently large[23].

Given the large number of possible combinations of parameter values, exhaustive testing of all the combinations is neither practical nor necessary. We take a greedy-search strategy for parameter tuning. That is, we first subjectively decide the order of parameters to be tuned, and then empirically find the "best" choice for each parameter one at a time, starting with the first parameter in the order and ending with the last. A typical or "promising" value is subjectively chosen as the default for each parameter as the starting point of the process. By varying the value of one parameter and fixing the values of other parameters, we identify the "best" value for that parameter; this value is then used as the fixed value of that parameter in the continued tuning process for other parameters.

5. Evaluation

5.1. Effects of Thresholding Strategies and Other Parameters

The performance variation of WORD, kNN and LLSF with respect to the choices of thresholding strategies and other parameter values were tested on Reuters version 3; the results are summarized in Table III. The original training set was split into two portions; one portion was used for training and the other portion was used for learning the optimal value of parameters. Most parameters were determined via the experiments with WORD and kNN because these systems run relatively fast. Only the p parameter was tuned using LLSF; other parameters in LLSF were set to the same values as those chosen for kNN. If there are a few best choices which produces almost equally good performance scores, we then present all of these choices. Several observations can be obtained from this table:

- TR (the length of ranked list) has little impact to the performance scores as long as it is larger than 5 in the experiments on WORD. This is not too surprising given that the number of categories per document is about 1.2 on average in the Reuters corpus. We then chose TR=10 in the remaining experiments for parameter tuning.
- The choices between TW = *ltc, atc, ntc* were almost equally good in both WORD and kNN. We chose *ltc* in the remaining experiments.
- FT in 2000-5000 appears to be the range of optimal choices when using the χ^2 feature selection criterion. We chose FT = 4113 (which is 15% of the training-set vocabulary) in the the remaining experiments.
- For the parameter k in kNN, the values of 30, 45 and 65 were tested; the resulting difference in the F_1 scores of of kNN are almost negligible.
- For the parameter p in LLSF, the values of 100, 200, 500 and 800 were tested; the performance of LLSF (measured using micro-averaging F_1) was approaching its plateau with $p \geq 500$.
- The effects of thresholding strategies varies in different classifiers. In WORD, using Pcut with the x value between 2.5 to 4.0 yielded much better results than using Rcut. In kNN and LLSF, on the other hand, the performance advantage of using Pcut over Rcut is much less. In all of these classifiers, Scut yielded the best results.

Except for thresholding strategy, most of the other parameters in Table III has a range (or a set) of values with which the classifiers had a relatively stabilized optimal performance. Choosing a value in the best range of each parameter, we obtained the evaluation results of WORD, kNN and LLSF in Table IV.

5.2. Effects of Collection Variability

The effects of collection variability on classifier performance were evaluated using our results of the WORD, kNN, and LLSF classifiers on the five versions of Reuters versions shown in Table V. Since WORD, kNN and LLSF are primarily ranking classifiers, the conventional 11-point average precision scores are presented. The scores on Reuters Version 3 are slightly higher than those shown in Table IV because the complete ranking of categories were used (by setting parameter TR = inf) in the experiments of Table V.

We believe that if two collections are statistically homogeneous, then the performance of a classifier should not vary appreciably between them; on the other hand, if the performance of a classifier changes dramatically when switching from a collection to a supposedly similar one, a careful analysis of the differences between the collections is called for. Since the results of one classifier may be biased, we chose three

Table III. Parameter tuning in WORD, kNN and LLSF on Reuters version 3, validation test set

Fixed Parameters	Tested Values	Best Choice(s)	Best 11-ptAvgp	Best microavgF_1
WORD				
TW=ltc	TR=5,10,15,20	TR= 5-20	.231-.233	-
TR=10	TW=atc,ltc,ntc,...	TW=ltc,atc,ntc	.237-.243	-
TR=10, TW=ltc	Scut: optimal F_1	category-specific	.242	.320
TR=10, TW=ltc	Pcut: $1.0 \leq x \leq 4.0$	x = 2.5-4.0	.242	.314-.320
TR=10, TW=ltc	Rcut: rank = 1, 2, 3	rank = 2	.242	.277
kNN (TR = 10)				
TW=ltc, Scut, k=30	FT=519-20595	2000-5000	.930-.935	.8454-.8456
FT=4113, Scut, k=30	TW=atc,ltc,ntc	ltc,atc,ntc	.925-.935	.838-.846
FT=4113, TW=ltc, Scut	k =30,45,65	30-65	.928-.935	.844-.851
FT=4113, TW=ltc, k=45	Scut: optimal F_1	category-specific	.933	.844
FT=4113, TW=ltc, k=45	Pcut: $1.0 \leq x \leq 2.0$	1.2-1.3	.933	.838-.839
FT=4113, TW=ltc, k=45	Rcut: rank = 1,2,3	rank = 1	.933	.809
LLSF (TR = 10)				
FT=4113, TW=ltc, Scut	p =100,200,500,800	500-800	.911-.912	.854-.858
FT=4113, TW=ltc, p=800	Pcut: $1.0 \leq x \leq 2.0$	x =1.2-1.3	.912	.811-.815
FT=4113, TW=ltc, p=800	Rcut: rank = 1, 2	rank = 1	.912	.810

Table IV. Results of WORD, kNN and LLSF on Reuters version 3, test set (3309 docs)

System	Parameter Settings	Ranking 11-ptAvgp	Scut microavgF_1	Pcut microavgF_1	Rcut microavgF_1
WORD	TR=10, TW=ltc	.221	.289	.311 (x=3.0)	.256 (rank = 2)
kNN	TR=10, TW=ltc, FT=3703, k=45	.924	.852	.834 (x=1.3)	.803 (rank = 1)
LLSF	TR=10, TW=ltc, FT=3703, p=800	.901	.855	.814 (x=1.3)	.792 (rank = 1)

different classifiers WORD, kNN and LLSF to evaluate the different versions of the Reuters collection. We chose these classifiers because they have fundamentally different classification algorithms, and we could closely control the conditions under which they were run. We did not use any of the published results of other classifiers because we could not carefully control the input data or conditions under which these classifiers were tested.

Looking at this table, one can see that versions 2.2, 3 and 4 appear to be relatively homogeneous, since the performance of the three classifiers varied only slightly between these different versions. These results suggest that the inclusion (as in version 2.2) or exclusion (as in versions 3 or 4) of categories with a 0-1 training-set frequency does not appear to have a significant impact on the performance of a classifier. They also suggest that different time-slicing in the training and test sets (version 3 vs. version 4) has only a minor effect on classifier performance. However, the differences in performance of these classifiers between version 2 and versions 2.2, 3 and 4 are significant. Given that the only difference between versions 2 and 2.2 is the inclusion (version 2) or exclusion (version 2.2) of the unlabeled documents, the cause for the performance variations of the classifiers is clear. An examination on randomly-selected test documents

Table V. Examination on kNN, LLSF and WORD on different versions of Reuters

Version	UniqCate	TrainDocs	TestDocs	(labelled)	kNN	LLSF	WORD
Version 1 (CGI)	182	21,450	723	(80%)	.80	-	.28
Version 2 (Lewis)	113	14,704	6,746	(42%)	.84	-	.10
Version 2.2 (Yang)	113	7,789	3,309	(100%)	.93	.92	.22
Version 3 (Apte)	93	7,789	3,309	(100%)	.93	.92	.22
Version 4 (PARC)	93	9,610	3,662	(100%)	.91	.91	.22

from version 2 shows that the documents appear to be classified correctly by kNN in many cases, but were counted as incorrectly classified in evaluation because the documents were unlabeled. Thus, the unlabeled documents of version 2 appear to cause a problem not with the classifier itself, but with the validity of the evaluation.

To further analyze the problem, let us assume that all of the unlabeled documents (58%) in Reuters version 2 test set belong to one or more categories. And now consider a perfect classifier and a trivial rejector classifier evaluated on this test set. The perfect classifier will have an assessed error rate of 58%, while the trivial classifier will have an assessed error rate of 42%. Neither score is an accurate reflection of the performance of the corresponding classifiers, and conclusions drawn from these scores will be erroneous. Of course, we do not know how many documents in Reuters version 2 should be labelled with categories and how many are unlabeled because they do not belong to any of the categories in the test or training sets, so this argument is only indicative of the problem. However, the sharp increase in performance of the two fundamentally different classifiers kNN (84% to 93%) and WORD (10% to 21%) going from version 2 to 2.2 strongly suggests that a large portion of the documents in Reuters version 2 should be labelled, but are not.

Reuters version 1 is something of an outlier collection compared to the other versions of the Reuters collection. It only contains 723 documents in the test set, of which 20% are unlabeled; it is not certain, again, whether the unlabeled documents should be labelled or not. Furthermore, the change in performance of WORD and kNN when switching from version 3 to version 1 calls into questions whether this test set is a random sample from the Reuters corpus. The statistical-learning algorithm of kNN declined in performance from .93 to .80 while the simple word-matching algorithm of WORD actually increased from 0.22 to 0.28. Taken in isolation, these performance changes clearly favor word-matching over statistical learning, even though this is well known not to be the case. Although we do not know what criteria were used to select the test documents, these results suggest that evaluations on Reuters version 1 would be difficult to interpret, or at least inconsistent with evaluations on other collections.

5.3. Cross-Method Comparison

This observations in the section above imply the importance of understanding the nature of test collections. Using a flawed collection (i.e., one with a large portion of documents incorrectly assumed to have no category labels) would lead to conclusions about classifier performance that have little to do with the true behavior of a classifier. Likewise, comparisons across collections without analyzing how performance of classifiers can be affected by collection differences would be equally misleading. These are the fundamental considerations in the discussions below regarding cross-method comparison.

Table VI summarizes previously-published results on Reuters versions 1-4, and the results of our new experiments using WORD, kNN, and LLSF. The micro-average BEP scores are used as the performance measure because it has been the most widely reported score for the classifiers evaluated on Reuters. The

Table VI. Results summary of TC systems on Reuters versions 1-4

System	Reuters version 1	Reuters version 2	Reuters version 3	Reuters version 4
WORD	-	.15 (Scut)	.31 (Pcut)	.29 (Pcut)
kNN	-	.69 (Scut)	.85 (Scut)	.82 (Scut)
LLSF	-	-	.85 (Scut)	.81 (Scut)
NNets.PARC (perceptron)	-	-	-	.82 (Pcut)
CLASSI (perceptron)	-	-	.80	-
RIPPER (DNF)	-	.72 (Scut)	.80 (Scut)	-
SWAP-1 (DNF)	-	-	.79	-
DTree IND	-	.67 (Pcut)	-	-
DTree C4.5	-	-	.79 (F_1)	-
CHARADE (DNF)	-	-	.78	-
EXPERTS (n-gram)	-	.75 (Scut)	.76 (Scut)	-
Rocchio	-	.66 (Scut)	.75 (Scut)	-
NaiveBayes	-	.65 (Pcut)	.71	-
CONSTRUE (Exp. Sys.)	.90	-	-	-

scores of WORD, kNN and LLSF are the interpolated BEP values when using the optimal parameters described in Section 5.1.

In this table, Reuters versions 2 and 3 have the densest columns. Several claims were made in published evaluations using the results on these collections. Cohen concluded that EXPERTS was the best performer ever on the Reuters version 2 collection[3], and Table VI agrees with this. Interestingly, EXPERTS was relatively insensitive to the removal of unlabeled documents in Reuters version 3, going from 0.75 to 0.76, while every other classifier evaluated on Reuters 2 and Reuters 3 showed a significant improvement (at least 6%) going from Reuters 2 to Reuters 3. Furthermore, while EXPERTS is the highest performing classifier on Reuters 2, it is among the lowest on Reuters 3. This is counter-intuitive because, given that classifiers' performance scores should improve when the unlabeled documents are removed from the test set; EXPERTS should continue to be the best performer or nearly the best on Reuters 3.

Apte et. al. [1] compared the results of SWAP-1 on Reuters version 3 to the results of NaiveBayes and DTree by Lewis on Reuters version 2[10], and concluded from its significantly better performance that rule-learning methods were superior to decision trees for text categorization. However, to see the perils of this conclusion, kNN has a performance of 0.69 on version 2, but a performance of 0.85 on version 3. The kNN algorithm's score on version 2 is lower than SWAP-1's, but higher on version 3. Should we conclude that SWAP-1 is better than kNN or the opposite? Even more interestingly, Moulinier recently reported the result of a DTree algorithm as 0.79 in F_1, which is close to SWAP-1's score in BEP on Reuters 3. However, since BEP is a lower bound on F_1, SWAP-1 may yet be a better performer than DTree.

In any case, the claim by Apte et al. about the advantage of rule-learning algorithms over non-rule learning algorithms is not convincing. Similarly, the claim by Cohen & Yoram[3] about the advantage of the context-sensitive classifiers (RIPPER and EXPERTS) over linear classifiers is not necessarily supported by the empirical results here. The rule-induction algorithms (SWAP-1, RIPPER and CHARADE) have similar performance on Reuters 3, which is very close to the result of DTree but not as good as kNN or LLSF on the same collection. Their performance is also below that of NNets based on an indirect comparison using kNN as the baseline on Reuters 3 and 4. While kNN and LLSF do not specify explicit term combinations, they use context implicitly. The classification function of LLSF, for example, is sensitive to weighted linear combinations of words that co-occur in training documents. While this sensitivity does not make LLSF equivalent to a non-linear classifier, it does make LLSF fundamentally different from methods that assume that terms are independent (e.g. NaiveBayes). The ability to acquire context-sensitivity implicitly in the classification functions may be one reason for the high performance of kNN and LLSF.

5.4. Global Observations

According to the scores on the clean versions of Reuters (version 3 and 4), kNN, NNet.PARC and LLSF classifiers exhibited the best performance; CLASSI, DTree and the rule-learning approaches (RIPPER, SWAP-1, and CHARADE) also performed reasonably well, with a level not too far behind the best-performing classifiers. NaiveBayes, surprisingly, had the worst performance on version 3, although significantly outperforming WORD, the non-learning method. The Rocchio method also showed a relatively poor performance. This suggests that Rocchio, while commonly used, is rather a weak baseline for comparisons of learning methods; kNN would be a more challenging alternative.

While global observations on the performance of categorization systems is informative, solid conclusions about specific learning algorithms are still difficult because of the lack of complete information about the performance evaluations of reported classifiers. For example, we have the results of kNN, LLSF and WORD with using different thresholding methods (Scut, Pcut and Rcut) for obtaining binary decisions. However, for all the other methods, the only results reported were those with using one thresholding method. RIPPER and EXPERTS had the Scut results only, while NNet.PARC were evaluated with Pcut only. We observed that Scut was better than Pcut for kNN on Reuters, but we do not know if this is generalizable to NNet.PARC. Therefore, we cannot simply conclude that kNN is equally good as NNet.PARC without knowing if Pcut is the best possible thresholding strategy for that method. Another missing piece of information here is that we do not know if the Pcut threshold (the value of parameter x described in Section 4.3) in NNet.PARC was tuned using test data; if it was, the reported performance of this system may (or may not) be higher than it would be if this parameter were tuned using training data solely.

Beyond thresholds for binary decisions, there are other experimental parameters which might contribute to the performance variations of classifiers, such as the choices made in stemming, term selection, term weighting, sampling strategies for training data, etc. It is not clear whether or not those systems would have worse results than the reported ones if parameter were tuned without test data. Without detailed information, we cannot be sure that a one or two percent difference in the scores of break-even point or F-measure is an indication of the theoretical strength or weakness of a learning method. It is also unclear how a significance test should be designed, given that the performance of a method is compressed into a single number, e.g., the break-even point or the optimized F_1. A variance analysis would be difficult given that the necessary information about performance variation with respect to different parameter settings is not generally published.

Finally, the point of the above analysis is not to show that a global, cross-method and cross-experiment evaluation is impossible or futile. On the contrary, missing detailed information should not prohibit the good use of available information for at least a partial comparison, and certainly for an indication of useful additional experiments. Moreover, by carefully analyzing test conditions in different experiments, as shown in this study, the factors underlying performance variations of classifiers will become much more transparent, leading to better understanding of evaluation problems and improved methodology for the future. Clearly, a carefully conducted comparative study across methods and experiments is useful, at least for observation on significant performance variations due to different choices in approaches, parameter settings, domains and tasks.

5.5. Efficiency of WORD, LLSF and kNN

WORD, the non-learning method, is simply a word-matching algorithm utilizing the inverted-file indexing of category names. The category ranking for a document is very fast, proportional to the number of unique words in the document. We observed a total time of 13 CPU seconds for the categorization of the 3309 test documents of Reuters version 3, which yields an on-line response of .004 CPU second per test document on average.

LLSF is an eager learning method, and has a off-line training phase and an on-line testing phase. The training phase has a quadratic time complexity, $O(pn')$ where p is the number of singular vectors used for computing an approximated LLSF solution[23], and $n' = \max\{m, n\}$ is the larger number between n, the number of training documents, and m, the number of unique terms in the training documents. This quadratic complexity is the computational bottleneck for scaling this method to large applications. Once the training is done, the on-line document categorization is very fast. In an experiment on the Reuters version 3 where $n = 7789$ and $p = 800$, we observed a training time of 3.3 CPU hours (on a SPARCstation Ultra-2); the on-line response of .004 CPU second per document in the testing phase.

kNN is a *lazy learning* instance-based method, and does not have a off-line training phase. The main computation is the on-line scoring of training documents given a test document, in order to find the k nearest neighbors. Using the inverted-file indexing of training documents, the time complexity is $O(l^2 n/m)$[22] where l is the number of unique words in the document, n is the number of training documents, and m is the number of unique terms in the training collection. In the experiments of kNN on Reuters version 3, we observed an on-line response of 0.3 CPU second per test document on average. In the further scaling test of kNN on OHSUMED, we observed 11.5 CPU minutes for the indexing of 183,229 training documents, and an on-line response of 1.0 CPU second per test document on average, with a micro-average F_1 value of .49[24]. This is the only text categorization method which has been examined on the full domain of OHSUMED.

In more general terms, the scaling problem in kNN can be reduced to the scaling problem in on-line document ranking, for which a number of techniques have been studied in the literature, including partial indexing and ranking[15, 2], document clustering[8], dimensionality reduction[27] and parallel computing[4].

6. Conclusions

The following conclusions are reached from this study:

- *Comparative evaluation across methods and experiments* is important for understanding the state-of-the-art in text categorization. As illustrated in this paper, by carefully analyzing not only empirical evidence but also the test conditions under which classifiers were evaluated, the factors underlying performance variations will become much more transparent and better understood, leading to improved evaluation methodology for the future.

- *Impact of collection variability* on classifier performance can be serious. In particular, including a large portion of unlabeled documents in the training or test set, and treating them as negative instances of all categories without knowing the ground truth has caused considerable confusion in text categorization research and has lead to inconsistent observations. The status of the unlabelled documents need to be clarified before results reported on Reuters version 2 can be fully understood.

- *Category ranking evaluation and binary classification evaluation* are both informative. The former reflects the usefulness of classifiers in interactive applications; the latter emphasizes their use in a batch mode. Providing both types of performance results for ranking classifiers makes the effects of thresholding strategies explicitly observable.

- A global observation of the performance of twelve classifiers on versions 3 & 4 of Reuters suggests that most of the learning methods performed reasonably well; all them significantly outperformed WORD, the non-learning method. At a finer level of distinction, kNN, NNets and LLSF are the top performers, followed by a group of classifiers (CLASSI, SWAP-1, RIPPER, DTree and CHARADE) whose performance scores are very close to each other. NaiveBayes, surprisingly, had the worst performance; Rocchio also had relatively poor performance.

- Evaluation of the scalability of classifiers in very large category spaces is an important but rarely investigated area. The kNN classifier is the only learning method evaluated on the full set of the OHSUMED categories, demonstrating the tractability of this method in a target space which is several-orders-of-magnitude larger than the category set of Reuters.

Acknowledgements

I would like to thank Jan Pedersen at InfoSeek, David Lewis and William Cohen at AT&T, and Isabelle Moulinier at University of Paris VI for providing information on their experiments. I would also like to thank Jaime Carbonell at Carnegie Mellon University for suggesting an improvement in binary decision making, Chris Buckley at Cornell for making the SMART system available, and Tom Ault for many valuable suggestions for improving the writing of this paper.

References

1. C. Apte, F. Damerau, and S. Weiss. Towards language independent automated learning of text categorization models. In *Proceedings of the 17th Annual ACM/SIGIR conference*, 1994.
2. Timothy A.H. Bell and Alisair Moffat. The design of a high performance information filtering system. In *Proceedings of the 19th Ann Int ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'96)*, pages 12–20, 1996.
3. William W. Cohen and Yoram Singer. Context-sensitive learning methods for text categorization. In *SIGIR '96: Proceedings of the 19th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 1996. 307-315.
4. R.H. 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. *Comm. ACM*, 35:48–63, 1992.
5. N. Fuhr, S. Hartmanna, G. Lustig, M. Schwantner, and K. Tzeras. Air/x - a rule-based multistage indexing systems for large subject fields. In 606-623, editor, *Proceedings of RIAO'91*, 1991.
6. P.J. Hayes and S. P. Weinstein. Construe/tis: a system for content-based indexing of a database of new stories. In *Second Annual Conference on Innovative Applications of Artificial Intelligence*, 1990.
7. W. Hersh, C. Buckley, T.J. Leone, and D. Hickman. Ohsumed: an interactive retrieval evaluation and new large text collection for research. In *17th Ann Int ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'94)*, pages 192–201, 1994.
8. Makato Iwayama and Takenobu Tokunaga. Cluster-based text categorization: a comparison of category search strategies. In *Proceedings of the 18th Ann Int ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'95)*, pages 273–281, 1995.
9. David D. Lewis, Robert E. Schapire, James P. Callan, and Ron Papka. Training algorithms for linear text classifiers. In *SIGIR '96: Proceedings of the 19th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 1996. 298-306.
10. D.D. Lewis and M. Ringuette. Comparison of two learning algorithms for text categorization. In *Proceedings of the Third Annual Symposium on Document Analysis and Information Retrieval (SDAIR'94)*, 1994.
11. Tom Mitchell. *Machine Learning*. McCraw Hill, 1996.
12. I. Moulinier. Is learning bias an issue on the text categorization problem? In *Technical report, LAFORIA-LIP6, Universite Paris VI*, 1997.

13. I. Moulinier, G. Raskinis, and J. Ganascia. Text categorization: a symbolic approach. In *Proceedings of the Fifth Annual Symposium on Document Analysis and Information Retrieval*, 1996.
14. H.T. Ng, W.B. Goh, and K.L. Low. Feature selection, perceptron learning, and a usability case study for text categorization. In *20th Ann Int ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'97)*, pages 67–73, 1997.
15. Michael Persin. Document filtering for fast ranking. In *Proceedings of the 17th Ann Int ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'94)*, pages 341–348, 1994.
16. J.R. Quinlan. Induction of decision trees. *Machine Learning*, 1(1):81–106, 1986.
17. G. Salton. *Automatic Text Processing: The Transformation, Analysis, and Retrieval of Information by Computer.* Addison-Wesley, Reading, Pennsylvania, 1989.
18. G. Salton and M. J. McGill. *Introduction to Modern Information Retrieval.* McGraw-Hill Computer Science Series. McGraw-Hill, New York, 1983.
19. K. Tzeras and S. Hartman. Automatic indexing based on bayesian inference networks. In *Proc 16th Ann Int ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'93)*, pages 22–34, 1993.
20. C.J. van Rijsbergen. *Information Retrieval.* Butterworths, London, 1979.
21. E. Wiener, J.O. Pedersen, and A.S. Weigend. A neural network approach to topic spotting. In *Proceedings of the Fourth Annual Symposium on Document Analysis and Information Retrieval (SDAIR'95)*, 1995.
22. Y. Yang. Expert network: Effective and efficient learning from human decisions in text categorization and retrieval. In *17th Ann Int ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'94)*, pages 13–22, 1994.
23. Y. Yang. Noise reduction in a statistical approach to text categorization. In *Proceedings of the 18th Ann Int ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'95)*, pages 256–263, 1995.
24. Y. Yang. An evaluation of statistical approach to text categorization. In *Technical Report CMU-CS-97-127, Computer Science Department, Carnegie Mellon University*, 1997.
25. Y. Yang and C.G. Chute. A linear least squares fit mapping method for information retrieval from natural language texts. In *Proceedings of the 14th International Conference on Computational Linguistics (COLING 92)*, pages 447–453, 1992.
26. Y. Yang and C.G. Chute. An example-based mapping method for text categorization and retrieval. *ACM Transaction on Information Systems (TOIS)*, pages 253–277, 1994.
27. Y. Yang and J.P. Pedersen. Feature selection in statistical learning of text categorization. In *The Fourteenth International Conference on Machine Learning*, pages 412–420, 1997.