A Probabilistic Analysis of the Rocchio Algorithm with TFIDF for Text Categorization

Thorsten Joachims

March 1996

CMU-CS-96-118

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Abstract

A probabilistic analysis of the Rocchio relevance feedback algorithm, one of the most popular learning methods from information retrieval, is presented in a text categorization framework. The analysis results in a probabilistic version of the Rocchio classifier and offers an explanation for the TFIDF word weighting heuristic. The Rocchio classifier, its probabilistic variant and a standard naive Bayes classifier are compared on three text categorization tasks. The results suggest that the probabilistic algorithms are preferable to the heuristic Rocchio classifier.

This research is sponsored by the Wright Laboratory, Aeronautical Systems Center, Air Force Materiel Command, USAF, and the Advanced Research Projects Agency (ARPA) under grant F33615-93-1-1330. The US Government is authorized to reproduce and distribute reprints for Government purposes, notwithstanding any copyright notation thereon. Views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of Wright Laboratory or the United States Government.

Keywords: text categorization, relevance feedback, naive Bayes classifier, information retrieval, vector space retrieval model, machine learning

1 Introduction

Text categorization is the process of grouping documents into different categories or classes. With the amount of online information growing rapidly, the need for reliable automatic text categorization has increased. Text categorization techniques are used, for example, to build a personalized netnews filter which learns about the news-reading preferences of a user [Lang, 1995]. They are used to classify news stories [Hayes et al., 1988] or guide a user's search on the World Wide Web [Mitchell et al., 1995].

Two kinds of algorithms have frequently been used for text categorization. Both are based on the bag-of-words representation developed in information retrieval. This is an attribute-value representation where documents are represented by the words occurring in it without attention to their ordering.

The first kind of algorithm is based on the relevance feedback method introduced by Rocchio [Rocchio, 1971] for the vector space retrieval model [Salton, 1991]. It is a nearest neighbor learning method with prototype vectors using the TFIDF (term frequency / inverse document frequency) [Salton, 1991] word weighting heuristic. This kind of algorithm will be called a TFIDF classifier in the following.

Naive Bayes classifiers make up the other group of algorithms. Based on the same bag-of-words representation, these algorithms use probabilistic models to estimate the likelihood that a given document is in a class. They use this probability estimate for decision making.

This paper introduces a probabilistic foundation for the TFIDF classifier. I will show how a particular TFIDF classifier can be analyzed in the same framework as used for naive Bayes classifiers. The analysis results in a probabilistic version of the TFIDF algorithm, called PrTFIDF, that suggests how the TFIDF algorithm can be improved. Empirical results on three categorization tasks show that PrTFIDF does not only allow a better theoretical understanding of the TFIDF algorithm, but that it also performs better in practice.

2 Paper Overview

Section 3 gives a short description of the text categorization problem and introduces its working definition used throughout this paper. A TFIDF classifier and a naive Bayes classifier are described in section 4. Section 5 presents the probabilistic analysis of the TFIDF classifier and states its implications for text categorization using TFIDF. Empirical results and the conclusions can be found in sections 6 and 7.

3 Text Categorization

The problem of text categorization is to classify documents into a fixed number of classes. But often the underlying concept of how to assign documents to a class is unknown or is difficult to formulate in a constructive fashion. The problem discussed in this paper is to find approximations to the category definitions automatically, given examples of correct category assignments. This task is a supervised learning problems [Quinlan, 1993] as commonly addressed in pattern recognition and machine learning.

The working definition of text categorization used in this paper assumes that the number of categories is fixed and known and each document is assigned to exactly one of them. To put it more formally, there is a set of classes $\mathcal{C}$ and a set of training documents D. Furthermore, there is a mapping $T(d) \in \mathcal{C}$ which is the true function that assigns documents to a class. $T(d)$ is known for the documents in the training set. The information contained in the training examples can be used by the learning algorithm to find a model or hypothesis $H(d) \in \mathcal{C}$ which identifies or approximates $T(d)$. $H(d)$ is the class which the learned hypothesis assigns document d to and it can be used to classify new documents. The objective is to find a hypothesis which maximizes accuracy, the percentage of times a hypothesis makes a correct prediction.

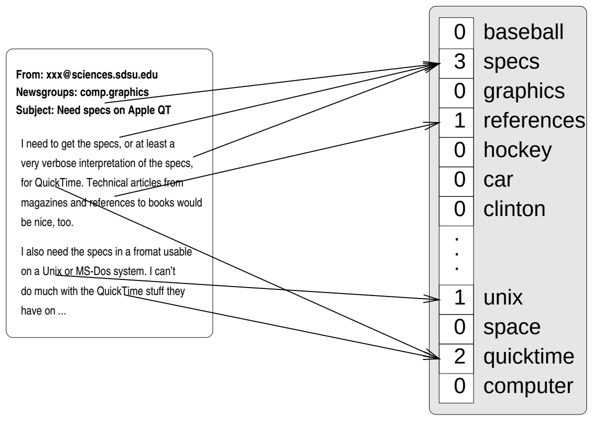
Throughout this paper accuracy will be used as the performance measure. Other researcher have proposed different evaluation measures, like precision recall graphs [Hayes et al., 1988] or the break-even point of recall and precision [Lewis, 1991]. Nevertheless, accuracy is an appropriate measure for many classification tasks, since they are not placed in a retrieval setting, but in a setting where a decision between more than two equally valuable classes has to be made.

4 Learning Methods for Text Categorization

This section describes the general framework for the experiments presented in this paper and defines the particular TFIDF classifier and the naive Bayes classifier used. The TFIDF classifier and the probabilistic model introduced for the naive Bayes classifier will be the basis for the analysis presented in section 5.

4.1 Representation

The representation of a problem has a strong impact on the generalization accuracy of a learning system. For the problem of text categorization a document, which typically is a string of characters, has to be transformed into a representation which is suitable for the learning algorithm and the classification task. As suggested by researchers in the IR community [Lewis, 1992], words seem to



From: xxx@sciences.sdsu.edu
Newsgroups: comp.graphics
Subject: Need specs on Apple QT

I need to get the specs, or at least a very verbose interpretation of the specs, for QuickTime. Technical articles from magazines and references to books would be nice, too.

I also need the specs in a fromat usable on a Unix or MS-Dos system. I can't do much with the QuickTime stuff they have on ...

0	baseball
3	specs
0	graphics
1	references
0	hockey
0	car
0	clinton
⋮	
1	unix
0	space
2	quicktime
0	computer

Figure 1: Bag-of-words representation in an attribute value style.

work well as features for many classification tasks. This simplifies the representation of documents from sequences of characters to sequences of words.

In addition to using words as indexing terms it is usually assumed that the ordering of the words in a document does not matter. This way documents no longer have to be represented as sequences. Instead a document can be represented as a bag of words. This representation is equivalent to an attribute-value representation as used in machine learning. Each distinct word is a feature and the number of times the word occurs in the document is its value. This value is called the term frequency $TF(w,d)$ of word w in document d. Figure 1 shows an example feature vector for a particular document. The terms "feature" and "word" will be used interchangeably in the following.

Representing documents as a bag of words is a common technique in information retrieval. Nevertheless one has to keep in mind that when using this representation, some information about the document is lost.

4.2 Feature Selection

In the field of machine learning many have argued that maximum performance is often not achieved by using all available features, but by using only a "good" subset of those [Caruana, Freitag, 1994]. The problem of finding a "good" subset of features is called *feature selection*. Applied to text classification this means that we want to find a subset of words which helps to discriminate between classes. Having too few features can make it impossible to formulate a good hypothesis. But having features which do not help discriminate between classes adds noise.

In this paper a combination of three feature selection methods is used, which are explained in more detail below:

1. Pruning of infrequent words.
2. Pruning of high frequency words.
3. Choosing words which have high mutual information [Quinlan, 1993] with the target concept.

Pruning of infrequent words means that words are only considered as features, if they occur at least a times in the training data. In the experiments described in this paper words had to occur at least 3 times. This removes most spelling errors and speeds up the following 2 stages of features selection.

In a second step the b most frequently occuring words are removed. This technique is supposed to eliminate non content words like “the”, “and”, or “for”.

In a final step the remaining words are ranked according to their mutual information $I(T,w)$ with the target concept. Mutual information measures the reduction in entropy that is achieved, if one random variable is conditioned on another one. In our case we look at how well the occurrence of a word predicts whether an article is in a certain category or not. Mutual information can be calculated as

$$\begin{aligned} I(T,w) &= E(T) - E(T|w) && (1) \\ &= -\sum_{C\in\mathcal{C}} \Pr(T(d)=C)\cdot\log\Pr(T(d)=C) \\ &\quad +\sum_{C\in\mathcal{C}} \Pr(T(d)=C,w=0)\cdot\log\Pr(T(d)=C|w=0) \\ &\quad +\sum_{C\in\mathcal{C}} \Pr(T(d)=C,w=1)\cdot\log\Pr(T(d)=C|w=1) && (2) \end{aligned}$$

$E(X)$ is the entropy of the random variable X. $\Pr(T(d)=C)$ is the probability that an arbitrary article d is in category $C\in\mathcal{C}$. $\Pr(T(d)=C,w=1)$ and $\Pr(T(d)=C,w=0)$ are the probabilities that article d is in category C and it does or does not contain word w. $\Pr(T(d)=C|w=0)$ and $\Pr(T(d)=C|w=1)$ are defined similarly.

To select a subset of n features, the n words with the highest mutual information are chosen. An example is given in table 1. The resulting feature (word)

wheat	washington	crop
tonnes	department	cts
agriculture	soviet	inc
grain	export	winter
usda	corn	company

Table 1: The 15 words with the highest mutual information for the Reuters category "wheat".

set is denoted by F with $|F| = n$. The following learning algorithms will only consider words which are elements of F.

4.3 Learning Algorithms

4.3.1 TFIDF Classifier

This type of classifier is based on the Rocchio relevance feedback algorithm [Rocchio, 1971] and uses TFIDF word weights. There are a number of algorithms in this family which differ in their selection of word weighting method and similarity measure [Salton, Buckley, 1987]. The variant presented here seems to be the most straightforward application of this type of algorithm to text categorization domains with more than two categories. Similar algorithms have been used in [Buckley et al., 1994] [Mitchell et al., 1995] [Balabanovic et al., 1995] [Schütze et al., 1995].

The basic idea of the algorithm is to represent each document d as a vector $\vec{d} = (d^{(1)}, \ldots, d^{(|F|)})$ in a vector space so that documents with similar content have similar vectors. Each dimension of the vector space represents a word selected by the feature selection process described above. This representation is closely related to the feature vector representation introduced in section 4.1.

The values of the vector elements $d^{(i)}$ for a document d are calculated as a combination of the statistics $TF(w,d)$ and $DF(w)$. The *term frequency* $TF(w,d)$ is the number of times word w occurs in document d. The *document frequency* $DF(w)$ is the number of documents in which the word w occurs at least once. The *inverse document frequency* $IDF(w)$ can be calculated from the document frequency.

$$IDF(w) = log\left(\frac{|D|}{DF(w)}\right) \tag{3}$$

$|D|$ is the total number of documents. The inverse document frequency of a word is low if it occurs in many documents and is highest if the word occurs in only one. The value $d^{(i)}$ of feature w_i for document d is then calculated as the product

$$d^{(i)} = TF(w_i, d) \cdot IDF(w_i) \tag{4}$$

$d^{(i)}$ is called the weight of word w_i in document d. This word weighting heuristic says that a word w_i is an important indexing term for document d if it occurs frequently in it (the term frequency is high). On the other hand, words which occur in many documents are rated less important indexing terms due to their low inverse document frequency. This representation is equivalent with the original feature vector representation from section 4.1 with each dimension being stretched by their $IDF(w)$ value.

The TFIDF algorithm learns a class model by combining document vectors into a prototype vector $\vec{c}$ for every class $C \in \mathcal{C}$. Prototype vectors are generated by adding the document vectors of all documents in the class.

$$\vec{c} = \sum_{d \in C} \vec{d} \tag{5}$$

The resulting set of prototype vectors, one vector for each $C \in \mathcal{C}$, represents the learned model.

This model can be used to classify a new document d'. Again the document is represented as a vector $\vec{d'}$ using the scheme described above. To classify d' the cosine of the prototype vector of each class with $\vec{d'}$ is calculated. The new document is assigned to the class with which its document vector has the highest cosine.

$$H_{TFIDF}(d') \quad = \quad \operatorname*{argmax}_{C \in \mathcal{C}} cos(\vec{d'}, \vec{c}) \tag{6}$$

$H_{TFIDF}(d')$ is the category to which the algorithm assigns document d'. The cosine measures the angle between the vector of the document being classified and the prototype vector of each of the classes. The smaller the angle, the larger the cosine. So d' is assigned to the class which has the smallest angle between its prototype vector and $\vec{d'}$. The algorithm can be summarized in the following decision rule:

$$H_{TFIDF}(d') = \operatorname*{argmax}_{C \in \mathcal{C}} \frac{\vec{d'} \cdot \vec{c}}{||\vec{d'}|| \cdot ||\vec{c}||} \tag{7}$$

$$= \operatorname*{argmax}_{C \in \mathcal{C}} \frac{\sum_{i=1}^{|F|} d'^{(i)} \cdot c^{(i)}}{\sqrt{\sum_{i=1}^{|F|} (d'^{(i)})^2} \cdot \sqrt{\sum_{i=1}^{|F|} (c^{(i)})^2}} \tag{8}$$

$$= \operatorname*{argmax}_{C \in \mathcal{C}} \frac{\sum_{w \in F} (TF(w, d') \cdot IDF(w)) \cdot (TF(w, C) \cdot IDF(w))}{\sqrt{\sum_{w' \in F} (TF(w', C) \cdot IDF(w'))^2}} \tag{9}$$

$TF(w, C)$ is the number of times word w occurs within the documents in class C. In (9) the normalization with the length of the document vector is left out since it does not influence the argmax.

4.3.2 Naive Bayes Classifier

The classifier presented in this section uses a probabilistic model of text. Although this model is a strong simplification of the true process by which text is generated, the hope is that it still captures most of the important characteristics.

Assumption: Documents are generated by drawing words from a probability distribution. Let's assume that we have $|\mathcal{C}|$ probability distributions, one for each category. All documents assigned to a particular class are generated from the probability distribution associated with this class in a number of independent trials. The i-th word of the document is generated by the i-th independent trial. This model is consistent with the bag-of-words assumptions introduced in section 4.1.

Probabilistic classifiers try to estimate $\Pr(C|d')$, the probability that a document d' is in class C. Bayes' rule [James, 1985] says that to achieve the highest classification accuracy, d' should be assigned to the class for which $\Pr(C|d')$ is highest.

$$H_{BAYES}(d') = \operatorname*{argmax}_{C \in \mathcal{C}} \Pr(C|d') \tag{10}$$

Bayes' theorem [James, 1985] can be used to split the estimation of $\Pr(C|d')$ into two parts.

$$\Pr(C|d') = \frac{\Pr(d'|C) \cdot \Pr(C)}{\sum_{C \in \mathcal{C}} \Pr(d'|C) \cdot \Pr(C)} \tag{11}$$

$\Pr(C)$ is the prior probability that a document is in class C. $\Pr(d'|C)$ is the likelihood of observing document d' in a given class.

$\hat{\Pr}(C)$, the estimate of $Pr(C)$, can be calculated from the fraction of the training documents that is assigned to this class.

$$\hat{\Pr}(C) = \frac{|C|}{\sum_{C' \in \mathcal{C}} |C'|} \tag{12}$$

$|C|$ is the number of training documents in a class.

The estimation of $\Pr(d'|C)$ is more difficult. $\Pr(d'|C)$ is the probability of observing a document like d' in class C. Since there is a huge number of different documents it is impossible to collect a sufficiently large number of training examples to estimate this probability without prior knowledge or further assumptions. In our case the estimation becomes possible due to the way documents are generated (assumption). The assumption implies that a word's occurrence is dependent on the class the document comes from, but that it occurs independently of the other words in the document. So $\Pr(d'|C)$ can be written as:

$$\Pr(d'|C) = \prod_{i=1}^{|d'|} \Pr(w_i|C) \tag{13}$$

w_i ranges over the sequence of words in document d' which are element of the feature vector F. Unlike other probabilistic classifiers [Lewis, 1991], this classifier does not use the binary occurrence of words in a document as features, but can take frequency information into account. $|d'|$ is the number of words in document d'. The estimation of $P(d'|C)$ is reduced to estimating each $P(w_i|C)$ independently. A Bayesian estimate is used for $\Pr(w_i|C)$.

$$\hat{\Pr}(w_i|C) = \frac{1 + TF(w_i, C)}{|F| + \sum_{w' \in |F|} TF(w', C)} \tag{14}$$

This estimator, which is often called Laplace estimator, is suggested in [Vapnik, 1982] (pages 54-55). It assumes that the observation of each word is a priori equally likely. I found that this Bayesian estimator works well in practice. Unlike the maximum likelihood estimator [Larson, 1982], it does not falsely estimates probabilities to be zero.

The following is the resulting decision rule if equations (10), (11) and (13) are combined.

$$\begin{aligned} H_{BAYES}(d') &= \operatorname*{argmax}_{C \in \mathcal{C}} \frac{\Pr(C) \cdot \prod_{i=1}^{|d'|} \Pr(w_i|C)}{\sum_{C' \in \mathcal{C}} \Pr(C') \cdot \prod_{i=1}^{|d'|} \Pr(w_i|C')} && (15) \\ &= \operatorname*{argmax}_{C \in \mathcal{C}} \frac{\Pr(C) \cdot \prod_{w \in F} \Pr(w|C)^{TF(w,d)}}{\sum_{C' \in \mathcal{C}} \Pr(C') \cdot \prod_{w \in F} \Pr(w|C')^{TF(w,d)}} && (16) \end{aligned}$$

If $\Pr(C|d')$ is not needed as a measure of confidence, the denominator can be left out since it does not change the argmax.

5 PrTFIDF: A Probabilistic Classifier Derived from TFIDF

In the following I will analyze the TFIDF classifier in a probabilistic framework. I will propose a classifier, called PrTFIDF, based on the probabilistic indexing paradigm [Fuhr, 1989] and then show its equivalence with the TFIDF algorithm under certain assumptions. The analysis will present a different view on classification using the Rocchio algorithm and the TFIDF word weighting heuristic.

Other researchers have already proposed theoretical interpretations of the vector space retrieval model [Bookstein, 1982][Wang et al., 1992] and the TFIDF word weighting scheme [Wong, Yao, 1989][Wu, Salton, 1981]. However, their work analyzes only parts of the TFIDF algorithm and is based on information retrieval instead of text categorization.

5.1 The PrTFIDF Algorithm

The PrTFIDF algorithm makes use of the probabilistic indexing paradigm [Fuhr, 1989], which offers an elegant way to distinguish between a document and the representation of a document. First, a function Θ maps the document to its representation; then, the classifier uses this representation for decision making. Like the classifier proposed in the previous section this algorithm tries to estimate the probability $\Pr(C|d', \Theta)$ that document d' is in class C. But now this probability explicitly includes the method Θ of how documents are represented. As in the previous section, Bayes' rule is used for making a prediction.

$$H_{PrTFIDF}(d') = \operatorname*{argmax}_{C \in \mathcal{C}} \Pr(C|d', \Theta) \tag{17}$$

$\Pr(C|d', \Theta)$ can be written in two parts.

$$\Pr(C|d', \Theta) = \sum_{x} \Pr(C|x) \cdot \Pr(x|d', \Theta) \tag{18}$$

$\Pr(x|d', \Theta)$ maps document d' to its representation x with a certain probability according to Θ. $\Pr(C|x)$ is the probability that a document with representation x is in class C.

The representation mapping Θ is a design choice. Section 5.2 will show that the mapping presented in the following is motivated by the TFIDF classifier. In particular, documents will be represented by single words. So $x = w$ and $\Pr(x|d', \Theta) = \Pr(w|d', \Theta)$. Each word is selected with a certain probability. This way documents do not have one fixed representation, but several ones. Word w

is chosen as a representation for d' with the same probability as if it was picked randomly from the bag of words which makes up d.

$$\hat{\Pr}(x|d', \Theta) = \hat{\Pr}(w|d', \Theta) = \frac{TF(w, d')}{\sum_{w' \in F} TF(w', d')} \tag{19}$$

The remaining part of equation 18, the probability $\Pr(C|x) = \Pr(C|w)$ that a document with representation x is in class C, can again be split into two parts using Bayes' theorem.

$$\Pr(C|w) = \frac{\Pr(w|C) \cdot \Pr(C)}{\sum_{C' \in \mathcal{C}} \Pr(w|C') \cdot \Pr(C')} \tag{20}$$

As in the previous section, $\Pr(C)$ can be estimated from the fraction of the training documents that are assigned to this class.

$$\hat{\Pr}(C) = \frac{|C|}{\sum_{C' \in \mathcal{C}} |C'|} \tag{21}$$

$|C|$ is the number of training documents in a class.

Making the same assumption about how documents are generated as for the Bayes classifier (section 4.3.2), $\Pr(w|C)$, the likelihood of observing w as a representation of a document in class C, can be estimated from the ratio of how often word w occurred in documents in class C, divided by the total number of words in the documents in class C.

$$\hat{\Pr}(w|C) = \frac{TF(w, C)}{\sum_{w' \in F} TF(w', C)} \tag{22}$$

The resulting decision rule for PrTFIDF is

$$H_{PrTFIDF}(d') = \arg\max_{C \in \mathcal{C}} \sum_{w \in F} \frac{\Pr(w|C) \cdot \Pr(C)}{\sum_{C' \in \mathcal{C}} \Pr(w|C') \cdot \Pr(C')} \cdot \Pr(w|d', \Theta) \tag{23}$$

5.2 The Equivalence of TFIDF and PrTFIDF

In the following I will to show the relationship of the PrTFIDF classification rule to the TFIDF algorithm from section 4.3.1. To achieve equivalence, the following assumptions have to be made:

1. Uniform class priors: Each class contains an equal number of documents. This assumption is true for a variety of datasets, for example the 20 Newsgroups experiment described later in this paper.

2. There is a λ so that for all classes:

$$\lambda \cdot \sum_{w' \in F} TF(w', C) = \sqrt{\sum_{w' \in F} (TF(w', C) \cdot IDF(w'))^2} \tag{24}$$

   This assumption states that the Euclidean length of the prototype vectors for each class is a linear function of the number of words in the class. The property is approximately met if each class contains an equivalent number of high and low frequency words. To find out how well the assumption holds in practice, I calculated λ for the 20 Newsgroups data. The assumption does hold approximately. Using a word vector of 15000 words the mean for λ over the 20 categories is 0.107 with a standard deviation of 0.021 and minimum and maximum values of 0.080 and 0.179.

3. Modified definition of $IDF(w)$: The definition of inverse document frequency as stated in section 4.3.1 was

$$IDF(w) = log\left(\frac{|D|}{DF(w)}\right) \tag{25}$$

   I now introduce a slightly different version of $IDF(W)$ which allows the transformation of TFIDF into PrTFIDF.

$$IDF'(w) = sqrt\left(\frac{|D|}{DF'(w)}\right) \tag{26}$$

$$DF'(w) = \sum_{C \in \mathcal{C}} \frac{TF(w, C)}{\sum_{w' \in F} TF(w', C)} \tag{27}$$

   There are two differences between this definition of $IDF(w)$ and the usual one. First, $DF'(w)$ is not the number of documents with an occurrence of word w. Instead it is the sum of the fractions of how much w takes up of the documents in each class. Nevertheless the dynamics of $DF(w)$ and $DF'(w)$ are similar. The more often a word w occurs throughout the corpus, the higher $DF(w)$ and $DF'(w)$. The dynamics are different only in case there is a small fraction of documents in which the word w occurs very frequently. Then $DF'(w)$ will rise faster than $DF(w)$.

The second difference is that not the logarithm is used to dampen the effect of the document frequency, but the square root. Nevertheless, both functions are similar in shape and reduce the impact of high document frequencies.

In the following I will start with the decision rule for PrTFIDF and transform it into the decision rule for TFIDF using the assumptions made above.

$$H_{PrTFIDF}(d') = \arg\max_{C \in \mathcal{C}} \sum_{w \in F} \frac{\Pr(w|C) \cdot \Pr(C)}{\sum_{C' \in \mathcal{C}} \Pr(w|C') \cdot \Pr(C')} \cdot \Pr(w|d', \Theta) \tag{28}$$

The term $\sum_{C' \in \mathcal{C}} \Pr(w|C') \cdot \Pr(C')$ in equation 28 can be re-expressed using the $IDF'(w)$ definition from above.

$$\begin{aligned}
\sum_{C' \in \mathcal{C}} \Pr(w|C') \cdot \Pr(C') &= P(C) \cdot \sum_{C' \in \mathcal{C}} \Pr(w|C') && (29)\\
&= \frac{1}{|\mathcal{C}|} \cdot \sum_{C' \in \mathcal{C}} \frac{TF(w, C')}{\sum_{w' \in F} TF(w', C')} && (30)\\
&= |C| \cdot \frac{\sum_{C' \in \mathcal{C}} \frac{TF(w,C')}{\sum_{w' \in F} TF(w',C')}}{|D|} && (31)\\
&= \frac{|C|}{IDF'(w)^2} && (32)
\end{aligned}$$

The equality in line (29) holds due to the assumption that all classes have equal priors. Replacing probabilities with their estimators in line (30), the expression can be reduced to a function of $IDF'(w)$.

Using this and again substituting probabilities with their estimators, the decision rule can be rewritten as:

$$H_{PrTFIDF}(d') = \text{argmax}_{C \in \mathcal{C}} \sum_{w \in F} \frac{TF(w, C) \cdot IDF'(w)^2 \cdot TF(w, d')}{|C| \cdot |\mathcal{C}| \cdot \sum_{w' \in F} TF(w', d') \cdot \sum_{w' \in F} TF(w', C)} \tag{33}$$

The forms $|C|$, $|\mathcal{C}|$ and $\sum_{w' \in F} TF(w', d')$ in the denominator do not influence the argmax since they are constant over all $C \in \mathcal{C}$; so they can be left out as shown in (34). The same argument applies for introducing the term $\sqrt{\sum_{w' \in F} (TF(w', C) \cdot IDF'(w'))^2}$. It equals the number of words in a class $\sum_{w' \in F} TF(w', C)$ when multiplied with a constant (assumption 2). In the way this assumption is used here it states that normalization with the Euclidean length and normalization with the number of words produce the same ranking. The resulting formula is equivalent with the decision rule (9) of TFIDF.

$$H_{PrTFIDF}(d')=\underset{C\in\mathcal{C}}{\operatorname{argmax}}\frac{\sum_{w\in F}(TF(w,d')\cdot IDF'(w))\cdot(TF(w,C)\cdot IDF'(w))}{\sqrt{\sum_{w'\in F}(TF(w',C)\cdot IDF'(w'))^2}} \quad (34)$$

5.3 Implications of the Analysis

The analysis shows how and under which preconditions the TFIDF classifier fits into a probabilistic framework. The close relationship to the probabilistic PrTFIDF classifier offers a theoretical justification for the vector space model and the TFIDF word weighting heuristic for text categorization. The analysis also suggests that the following changes to the original TFIDF algorithm might lead to an improved classifier. PrTFIDF is an implementation of TFIDF incorporating these changes.

- Use of prior probabilities $P(C)$.
- Use of $IDF'(w)$ (section 5.2) for word weighting instead of $IDF(w)$.
- Use of the number of words for normalization instead of the Euclidean length.

6 Experiments

Experiments were performed to show the performance of PrTFIDF, TFIDF, and the naive Bayes classifier BAYES on real data. The paper includes results on three different classification tasks. One task is set in a Netnews domain, the other two are based on Reuters news stories. For each task I will present results which show how the three classifiers perform with respect to changing numbers of examples and changing numbers of words selected in the feature selection process. Experimental results for additional datasets and additional methods can be found in [Mitchell et al., 1996].

6.1 Data Sets

6.1.1 Newsgroup Data

This data set consist of Usenet articles Lang collected from 20 different newsgroups (table 2). Over a period of time 1000 articles were taken from each of the newsgroups, which makes an overall number of 20000 documents in this collection. Except for a small fraction of the articles, each document belongs to exactly one newsgroup. The task is to learn which newsgroup an article was

comp.graphics	talk.politics.guns
comp.windows.x	talk.politics.mideast
comp.os.ms-windows.misc	talk.politics.misc
comp.sys.mac.hardware	talk.religion.misc
comp.sys.ibm.pc.hardware	soc.religion.christian
sci.electronics	alt.atheism
sci.crypt	rec.autos
sci.space	rec.motorcycles
sci.med	rec.sport.baseball
misc.forsale	rec.sport.hockey

Table 2: Usenet newsgroups used in newsgroup data.

posted to[1].

In addition to the body of an article, the subject line is used for classification. The documents in this dataset have the typical properties of Usenet articles. Many articles contain signature files and parts of other articles are cited.

The results reported on this dataset are averaged over a number of different test/training splits using binomial sign tests for estimating significance. A random subset of 33% of the data considered in an experiment was used for testing. For this dataset the 100 most frequent words are thrown out in the feature selection process.

6.1.2 Reuters Data

The Reuters dataset has been used in a variety of classification experiments [Hayes et al., 1988] [Lewis, 1991] [Apté and Damerau, 1994] [Lewis, 1992]. The data was collected by the Carnegie group from the Reuters newswire in 1987. Lewis compiled the collection into a corpus of 21,450 articles which are classified into 135 topic categories. Each article can have multiple category labels. 31% of the articles have no category label, 57% have exactly one label and the remaining 12% have more than one and up to twelve class labels assigned. The class assignment was done manually.

The distribution of members in each of the classes is very uneven. The 20 most frequently assigned classes are shown in table 3. The category assignment contains many spelling errors so that articles end up in obviously misspelled

[1]About 4% of the articles were cross-posted among two of the newsgroups. In these cases predicting either of the two newsgroups was counted as a correct prediction.

earn	4053	interest	497
acq	2427	wheat	293
cbond	1122	ship	290
bypass	1041	corn	241
money-fx	745	ebond	228
grain	601	dlr	185
crude	585	money-supply	177
corp-news	511	oilseed	176
loan	509	sugar	169
trade	504	coffee	145

Table 3: The 20 most frequently assigned topic categories for the Reuters data with the number of examples assigned to them.

categories.

This paper follows the usual experimental setup on this collection to split the prediction task into binary decision. For each class an independent classifier is trained which predicts whether or not an article has the topic assigned to it. The body of the articles and the subject line are used for classification. The experiments described in this paper are not done for all of the categories. Instead I will present a more detailed analysis of two categories which have certain key properties.

The categories "acq" and "wheat" are good representatives of the different kinds of classes there are in the Reuters data. The category "acq" is the one with the second most documents in it. It contains articles about corporate acquisitions. Most of them are short news stories in plain text. The category "wheat" contains articles about wheat in any context.

The "wheat" category has a very narrow definition. There is a small number of words which are very good clues as to whether a document is in this category or not. A simple classifier which classifies a document according to whether or not it contains the word `wheat` has an accuracy of 99.7%. In the test set, 80 of the 81 documents in the "wheat" category have the word `wheat` in it, but only 21 of the 6665 negative examples do. The category "acq" does not have such an obvious definition. Its concept is more abstract and a successful classifier will have to consider and combine a number of words.

For the experiments described in this paper the following test/training split was used. All articles which appeared on April 7, 1987 or before are in the training set. The articles which appeared on April 8 and later are in the test set. This

	20 Newsgroups	Reuters "acq"	Reuters "wheat"
PrTFIDF	90.3	89.3	95.6
BAYES	88.6	89.3	95.6
TFIDF	82.3	87.9	94.0

Table 4: Maximum accuracy in percent for the best parameter setting.

results in a corpus of 14,704 training examples and 6,746 test examples. To allow a more differentiated analysis of the results, the data was subsampled in such a way that there is an equal number of positive and negative examples. Since for all categories there is a larger number of negative examples than positive ones, a subset of the negative examples is selected randomly on the training set as well as on the test set. The results presented here are averaged over a number of trials and binomial sign tests and t-tests are used to estimate significance.

6.2 Experimental Results

6.2.1 Which Approach is most accurate?

Table 4 shows the maximum accuracy each learning method achieves at the best parameter setting. On the newsgroup data PrTFIDF performs significantly better than BAYES and BAYES is significantly better than TFIDF. There is no difference between the performance of PrTFIDF and BAYES on the both Reuters tasks. For both Reuters tasks the probabilistic methods perform significantly better than TFIDF. Nevertheless, the difference is much smaller than on the newsgroup data.

6.2.2 How does the Number of Training Examples Influence Accuracy?

As expected the accuracy increases with the number of training examples. This holds for all learning methods and categorization tasks. Nevertheless, there are differences in how quickly the accuracy increases. Figure 2 shows accuracy in relation to the number of training examples for the newsgroup experiment. In contrast to BAYES, PrTFIDF does particularly well for small numbers of training examples. The performance of BAYES approaches the one of PrTFIDF for higher numbers. The accuracy of the TFIDF classifier increases less quickly than for the probabilistic methods.

For the Reuters category "acq" BAYES and PrTFIDF show nearly identical curves (table 3). TFIDF is significantly below the two probabilistic methods

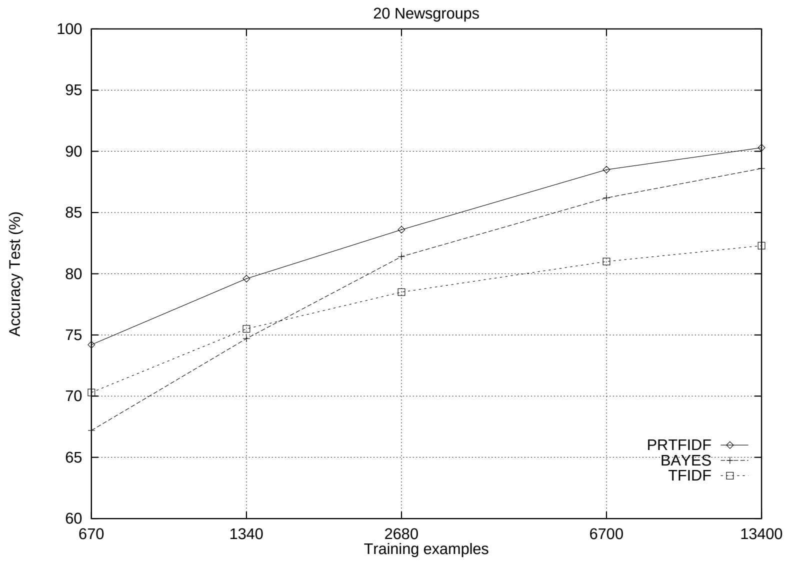


Figure 2: Accuracy versus the number of training examples on the newsgroup data using the maximum number of features.

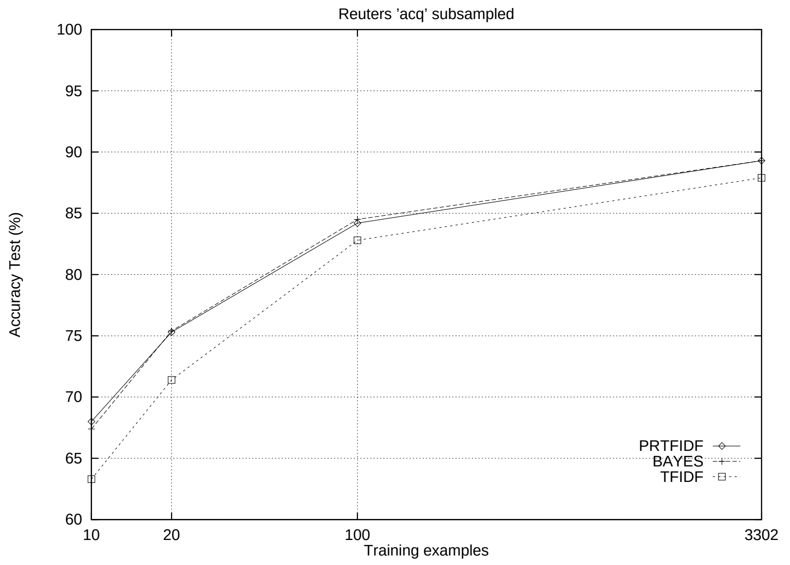


Figure 3: Accuracy versus the number of training examples on the Reuters category “acq” using the maximum number of features.

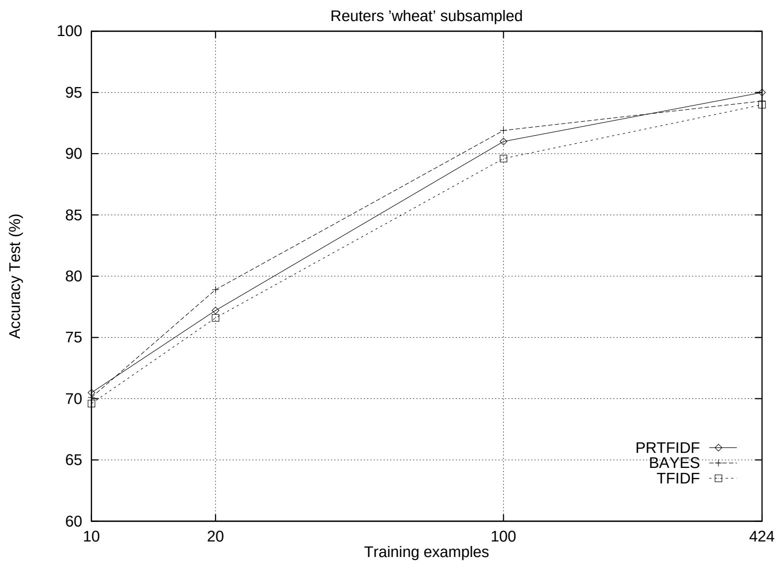


Figure 4: Accuracy versus the number of training examples on the Reuters category "wheat" using the maximum number of features.

over the whole spectrum. There are no big differences between the three methods for the Reuters category "wheat" (figure 4).

6.2.3 What is the Influence of the Number of Features on the Accuracy?

Table 5 shows the influence of the number of words chosen during the feature selection process on the accuracy for the newsgroup data. Keeping the number of training examples at maximum, the performance of the system is higher the more words are used. This stands in contrast to observations other researchers did on other datasets (e. g. [Lewis, 1991]). PrTFIDF and BAYES are significantly above TFIDF. The overall highest performance is achieved using PrTFIDF with the largest feature set.

The findings on the Reuters category "acq" are similar. Table 6 shows increasing performance of all methods with an increasing number of features. Over the whole spectrum PrTFIDF and BAYES perform significantly better than TFIDF. There is no significant difference in accuracy between BAYES and PrTFIDF. The highest performance is achieved with BAYES and PrTFIDF using the maximum number of words.

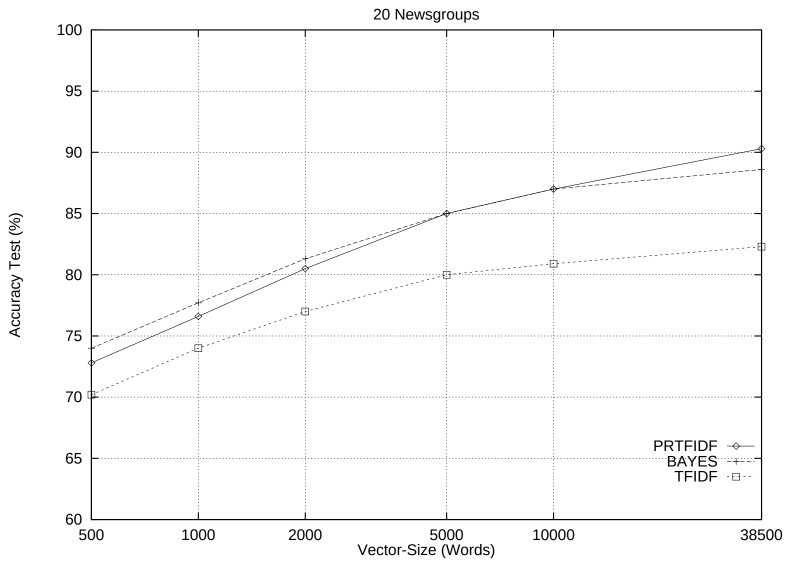


Figure 5: Accuracy versus the number of features on the newsgroup data using the maximum number of training examples.

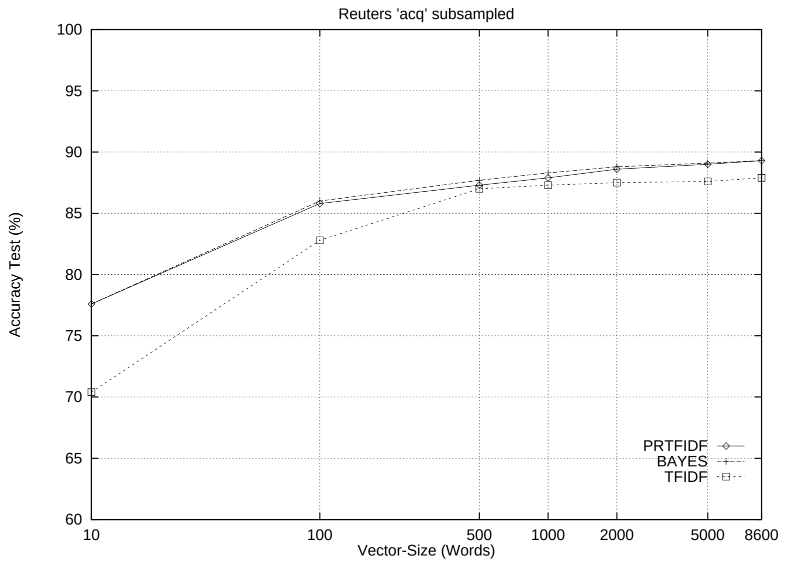


Figure 6: Accuracy versus the number of features on the Reuters category "acq" using the maximum number of training examples.

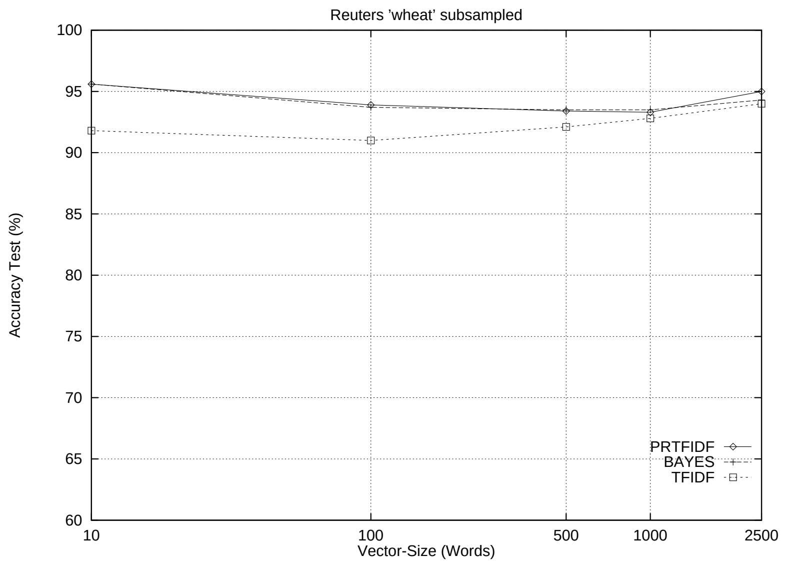


Figure 7: Accuracy versus the number of features on the Reuters category "wheat" using the maximum number of training examples.

The Reuters category "wheat" shows different characteristics from the ones seen on the previous two datasets. Table 7 shows that for the probabilistic methods the accuracy does not rise with the number of words used. The highest performance is achieved by PrTFIDF and BAYES when only the minimum number of 10 words is used.

The findings for the wheat category are probably due to the different properties of this task. As described in the previous section the definition of the category "wheat" is more narrow than the definition of the other ones. The single word `wheat` is a nearly perfect predictor for class membership. This explains why the maximum performance is achieved for small numbers of words. Adding more words adds noise, since these are words with lower predictive power.

6.2.4 Which Method is Most Robust for Small Numbers of Training Examples?

In figure 2, it is interesting to see that with a rising number of training examples, the performance of BAYES approximates that of PrTFIDF. Relative to PrTFIDF, BAYES becomes less accurate for big word-vector sizes (figure

5) on this dataset. Experiments on the newsgroups data have shown that the smaller the size of the word-vector, the fewer training examples are needed for BAYES to achieve or exceed the performance of PrTFIDF. My conjecture is that BAYES is very sensitive to the inaccurate probability estimates which arise with a low number of training examples. Looking at equation 16, each single estimate $P(w|C)$ can alter the outcome of the decision rule due to its multiplicative form. This means that a single bad estimate can completely change the behavior of the classifier. In PrTFIDF probabilities are not multiplied, but they are combined in a sum. Due to this "averaging" over the probabilities, single bad estimates do not have such a big influence on the outcome.

This behavior cannot be observed on the other datasets. For the two Reuters categories the performance of PrTFIDF and BAYES is very similar over the whole spectrum.

7 Conclusions

This paper shows the relationship between text classifiers using TFIDF and probabilistic classifiers. It presents a probabilistic analysis of a particular TFIDF classifier which shows that under certain reasonable assumptions this algorithm can be described using the same basic techniques from statistical pattern recognition that are used in probabilistic classifiers like BAYES. Furthermore, the analysis offers a theoretical explanation for the TFIDF word weighting heuristic in combination with the vector space retrieval model for text categorization. It results in a probabilistic version of the TFIDF classifier, which can be described in a probabilistic indexing framework.

Empirical results on three different classification tasks also support the idea that there is a close relationship between the three algorithms. None of the three methods failed completely on any of the classification tasks, and all three algorithms show similar characteristics for varying numbers of training examples and varying numbers of features. Nevertheless, the two probabilistic methods showed much better performance than the heuristic TFIDF classifier on one of the three categorization tasks and smaller gains on the other two. These empirical results suggest that the probabilistically accurate modelling of the bag-of-words assumptions is preferable to the heuristic TFIDF modelling. The probabilistic methods are preferable from a theoretical viewpoint, too, since a probabilistic framework allows the clear statement and easier understanding of the simplifying assumptions made.

8 Future Work

The two probabilistic classifiers described in this paper - and, as the theoretical analysis shows, the TFIDF classifier - are based on statistical pattern recognition techniques. Classification is done by estimating $P(d|C)$, the likelihood of observing document d in class C. Machine learning research suggests that there is the alternative to learn in a "learning as search" framework using explicit error minimization instead. In this framework the estimation of $P(d|C)$ is not necessary. Nevertheless, the problem of overfitting arises when learning is done by explicit error minimization in a noisy environment. This might be a challenging problem due to the large number of possible features. The methods described in this paper do not overfit since they do not perform search in a hypothesis space using explicit error minimization but rely on probability estimates.

9 Acknowledgements

I would like to thank Tom Mitchell for his inspiring comments on this work. Many thanks also to Sebastian Thrun, Phoebe Sengers, and Sean Slattery for their suggestions regarding this paper, and to Ken Lang for the dataset and parts of the code use in the experiments. This research is supported by ARPA under grant number F33615-93-1-1330.

References

[Apté and Damerau, 1994] C. Apté, F. Damerau, *"Automated Learning of Decision Rules for Text Categorization"*, ACM Transactions on Information Systems, Vol. 12, No. 3, July 1994, pages 233-251.

[Balabanovic et al., 1995] M. Balabanovic, Y. Shoham, *"Learning Information Retrieval Agents: Experiments with Automated Web Browsing"*, AAAI Spring Symposium Series on Information Gathering from Distributed, Heterogeneous Environments, Working Notes, 1995, http://www.isi.edu/sims/knoblock/sss95/balabanovic.ps

[Bookstein, 1982] A. Bookstein, *"Explanation and Generalization of Vector Models in Information Retrieval"*, Research and Development in Information Retrieval, Berlin, 1982.

[Buckley et al., 1994] C. Buckley, G. Salton, J. Allan, *"The Effect of Adding Relevance Information in a Relevance Feedback Environment"*, International ACM SIGIR Conference on Research and Development in Information Retrieval, 1994.

[Caruana, Freitag, 1994] R. Caruana, D. Freitag, *"Greedy Attribute Selection"*, International Conference on Machine Learning, 1994.

[Fuhr, 1989] N. Fuhr, *"Models for Retrieval with Probabilistic Indexing"*, Journal of Information Processing and Management, 25(1), pages 55-72, 1989.

[Hayes et al., 1988] P. Hayes, L. Knecht, M. Cellio, *"A news story categorization system"*, Second Conference on Applied Natural Language Processing, pages 9-17, 1988.

[James, 1985] M. James, *"Classification Algorithms"*, Wiley, 1985.

[Lang, 1995] K. Lang, *"NewsWeeder: Learning to Filter Netnews"*, International Conference on Machine Learning, 1995, http://anther.learning.cs.cmu.edu/ml95.ps

[Larson, 1982] H. Larson, *"Introduction to Probability Theory and Statistical Inference"*, Wiley, New York, 1982.

[Lewis, 1991] D. Lewis, *"Representation and Learning in Information Retrieval"*, Ph. D. Thesis, Department of Computer and Information Science, University of Massachusetts, COINS Technical Report 91-93, 1991.

[Lewis, 1992] D. Lewis, *"An Evaluation of Phrasal and Clustered Representations on a Text Categorization Task"*, International ACM SIGIR Conference on Research and Development in Information Retrieval, 1992.

[Mitchell et al., 1995] R. Armstrong, D. Freitag, T. Joachims, T. Mitchell, *"WebWatcher: A Learning Apprentice for the World Wide Web",* 1995 AAAI Spring Symposium on Information Gathering from Heterogeneous, Distributed Environments, March 1995, http://www.cs.cmu.edu/afs/cs/project/theo-6/web-agent/www/webagent-plus.ps.Z

[Mitchell et al., 1996] D. Freitag, H. Hirsh, T. Joachims, A. Kawamura, E. de Kroon, D. Loewenstern, B. McBarron, T. Mitchell, S. Slattery, *"Experiments in Learning from Text"*, to be published as a technical report, School of Computer Science, Carnegie Mellon University, 1996.

[Quinlan, 1993] J.R. Quinlan, *"C4.5: Programs for Machine Learning",* Morgan Kaufmann, 1993.

[Rocchio, 1971] J. Rocchio. *"Relevance Feedback in Information Retrieval"*, in The SMART Retrieval System: Experiments in Automatic Document Processing, Chapter 14, pages 313-323, Prentice-Hall Inc., 1971.

[Salton, 1991] G. Salton, *"Developments in Automatic Text Retrieval,* Science, Vol. 253, pages 974-979, 1991.

[Salton, Buckley, 1987] G. Salton, C. Buckley, *"Term Weighting Approaches in Automatic Text Retrieval"*, Technical Report 87-881, Department of Computer Science, Cornell University, 1987, http://cs-tr.cs.cornell.edu/TR/CORNELLCS:TR87-881

[Schütze et al., 1995] H. Schütze, D. Hull, J. Pedersen, *"A Comparison of Classifiers and Document Representations for the Routing Problem"*, International ACM SIGIR Conference on Research and Development in Information Retrieval, 1995, ftp://parcftp.xerox.com/pub/qca/SIGIR95.ps

[Vapnik, 1982] V. Vapnik, *"Estimation of Dependencies Based on Empirical Data"*, Springer Series in Statistics, Springer-Verlag, 1982.

[Wang et al., 1992] Z. Wang, S. Wong, Y. Yao, *"An Analysis of Vector Space Models Based on Computational Geometry"*, International ACM SIGIR Conference on Research and Development in Information Retrieval, 1992.

[Wong, Yao, 1989] S. Wong, Y. Yao, *"A Note on Inverse Document Frequency Weighting Scheme"*, Technical Report 89-990, Department of Computer Science, Cornell University, 1989, http://cs-tr.cs.cornell.edu/TR/CORNELLCS:TR89-990

[Wu, Salton, 1981] H. Wu, G. Salton, *"A Comparison of Search Term Weighting: Term Relevance vs. Inverse Document Frequency"*, Technical Report 81-457, Department of Computer Science, Cornell University, 1981, http://cs-tr.cs.cornell.edu/TR/CORNELLCS:TR81-457