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

Thorsten Joachims

Universität Dortmund, Fachbereich Informatik, Lehrstuhl 8
Baroper Str. 301
44221 Dortmund, Germany
thorsten@ls8.informatik.uni-dortmund.de

Abstract

The Rocchio relevance feedback algorithm is one of the most popular and widely applied learning methods from information retrieval. Here, a probabilistic analysis of this algorithm is presented in a text categorization framework. The analysis gives theoretical insight into the heuristics used in the Rocchio algorithm, particularly the word weighting scheme and the similarity metric. It also suggests improvements which lead to a probabilistic variant of the Rocchio classifier. The Rocchio classifier, its probabilistic variant, and a naive Bayes classifier are compared on six text categorization tasks. The results show that the probabilistic algorithms are preferable to the heuristic Rocchio classifier not only because they are more well-founded, but also because they achieve better performance.

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 personalized netnews filter which learn about the news-reading preferences of a user [Lang, 1995]. They are used to index news stories [Hayes et al., 1988] or guide a user's search on the World Wide Web [Joachims et al., 1997].

One of the most widely applied learning algorithms for text categorization is the Rocchio relevance feedback method [Rocchio, 1971] developed in information retrieval. Originally designed for optimizing queries from relevance feedback, the algorithm can be adapted to text categorization and routing problems. Although the algorithm is intuitive, it has a number of problems which - as I will show - lead to comparably low classification accuracy: (1) The objective of the Rocchio algorithm is to maximize a particular functional (introduced in section 3.2.1). Nevertheless Rocchio does not show why maximizing this functional should lead to a high classification accuracy. (2) Heuristic components of the algorithm offer many design choices and there is little guidance when applying this algorithm to a new domain. (3) The algorithm was developed and optimized for relevance feedback in information retrieval; it is not clear which heuristics will work best for text categorization.

The major heuristic component of the Rocchio algorithm is the TFIDF (term frequency / inverse document frequency) [Salton, Buckley, 1988] word weighting scheme. Different flavors of this heuristic lead to a multitude of different algorithms. Due to this heuristic this class of algorithms will be called TFIDF classifiers in the following.

A more theoretically founded approach to text categorization provide naive Bayes classifiers. These algorithms use probabilistic models for classification and allow the explicit statement of simplifying assumptions.

The contribution of this paper is a probabilistic analysis of a TFIDF classifier. This analysis makes the implicit assumption of the TFIDF classifier as explicit as for the naive Bayes classifier. Furthermore it provides insight into how the TFIDF algorithm can be improved, leading to a probabilistic version of the TFIDF algorithm, called PrTFIDF. PrTFIDF optimizes the different design choices of the TFIDF algorithm as a whole and gives clear recommendations on how to set the parameters involved. Empirical results on six categorization tasks show that PrTFIDF not only enables a better theoretical understanding of the TFIDF algorithm, but also performs better in practice without being conceptually or computationally more complex.

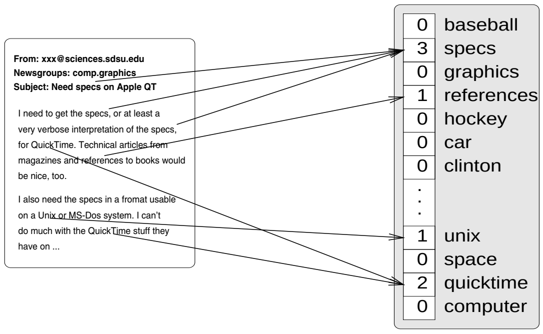

Figure 1: Bag-of-words representation in an attribute value style.

This paper is structured as follows. Section 2 introduces the definition of text categorization used throughout this paper. A TFIDF classifier and a naive Bayes classifier are described in section 3. Section 4 presents the probabilistic analysis of the TFIDF classifier and states its implications. Empirical results and conclusions can be found in sections 5 and 6.

2 Text Categorization

The goal of text categorization is the classification of documents into a fixed number of predefined categories. The working definition used throughout this paper assumes that each document d is assigned to exactly one category. To put it more formally, there is a set of classes $\mathcal{C}$ and a set of training documents $D \subseteq \mathcal{D}$. Furthermore, there is a target concept $T : \mathcal{D} \rightarrow \mathcal{C}$ which maps documents to a class. $T(d)$ is known for the documents in the training set. Through supervised learning the information contained in the training examples can be used to find a model or hypothesis $H : \mathcal{D} \rightarrow \mathcal{C}$ which approximates T. $H(d)$ is the function defining the class to which the learned hypothesis assigns document d; it can be used to classify new documents. The objective is to find a hypothesis which maximizes accuracy, i.e. the percentage of times H and T agree.

3 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 provides the basis for the analysis in section 4.

3.1 Representation

The representation of a problem has a strong impact on the generalization accuracy of a learning system. For 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. IR research suggests that words work well as representation units and that their ordering in a document is of minor importance for many tasks. This leads to a representation of documents as bags of words.

This bag-of-words representation is equivalent to an attribute-value representation as used in machine learning. Each distinct word corresponds to a feature with the number of times the word occurs in the document as its value. Figure 1 shows an example feature vector for a particular document. To avoid unnecessarily large feature vectors words are considered as features only if they occur in the training data at least m (e.g. $m = 3$) times. The set of considered features (i.e. words) will be called F.

3.2 Learning Algorithms

3.2.1 TFIDF Classifier

This type of classifier is based on the relevance feedback algorithm originally proposed by Rocchio [Rocchio, 1971] for the vector space retrieval model [Salton, 1991]. Due to its heuristic components, there are a number of similar algorithms corresponding to the particular choice of those heuristics. The three main design choices are

- the word weighting method
- the document length normalization
- the similarity measure.

An overview of some heuristics is given in [Salton, Buckley, 1988]. In the following the most popular combination will be used (known as "tfc"): "tf" word weights [Salton, Buckley, 1988], document length normalization using Euclidian vector length and cosine similarity.

Originally developed for information retrieval, the algorithm returns a ranking of documents without providing a threshold to define a decision rule for class membership. Therefore the algorithm has to be adapted to be used for text categorization. The variant presented here seems to be the most straightforward adaptation of the Rocchio algorithm to text categorization and domains with more than two categories.

The algorithm builds on the following representation of documents. Each document d is represented as a vector $\vec{d} = (d^{(1)}, \ldots, d^{(|F|)})$ so that documents with similar content have similar vectors (according to a fixed similarity metric). Each element $d^{(i)}$ represents a distinct word w_i. $d^{(i)}$ for a document d is calculated as a combination of the statistics $TF(w_i, d)$ and $DF(w_i)$ [Salton, 1991]. The *term frequency* $TF(w_i, d)$ is the number of times word w_i occurs in document d and the *document frequency* $DF(w_i)$ is the number of documents in which word w_i occurs at least once. The *inverse document frequency* $IDF(w_i)$ can be calculated from the document frequency.

$$IDF(w_i) = log\left(\frac{|D|}{DF(w_i)}\right) \tag{1}$$

Here, $|D|$ is the total number of documents. Intuitively, 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 so-called weight $d^{(i)}$ of word w_i in document d is then

$$d^{(i)} = TF(w_i, d) \cdot IDF(w_i) \tag{2}$$

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.

Learning is achieved by combining document vectors into a prototype vector $\vec{c_j}$ for each class C_j. First, both the normalized document vectors of the positive examples for a class as well as those of the negative examples for a class are summed up. The prototype vector is then calculated as a weighted difference of each.

$$\vec{c_j} = \alpha \frac{1}{|C_j|} \sum_{\vec{d} \in C_j} \frac{\vec{d}}{||\vec{d}||} - \beta \frac{1}{|D - C_j|} \sum_{\vec{d} \in D - C_j} \frac{\vec{d}}{||\vec{d}||} \tag{3}$$

α and β are parameters that adjust the relative impact of positive and negative training examples. As recommended in [Buckley et al., 1994] $\alpha = 16$ and $\beta = 4$ will be used in the following. C_j is the set of training documents assigned to class j and $||\vec{d}||$ denotes the Euclidian length of a vector $\vec{d}$. Additionally Rocchio requires that negative elements of the vector c_j are set to 0. Using the cosine as a similarity metric and $\alpha = \beta = 1$, Rocchio shows that each prototype vector maximizes the mean similarity of the positive training examples with the prototype vector c_j minus the mean similarity of the negative training examples with the prototype vector c_j.

$$\frac{1}{|C_j|} \sum_{\vec{d} \in C_j} \cos(\vec{c_j}, \vec{d}) - \frac{1}{|D - C_j|} \sum_{\vec{d} \in D - C_j} \cos(\vec{c_j}, \vec{d}) \tag{4}$$

Nevertheless it is unclear if or how maximizing this functional connects to the accuracy of the resulting classifier.

The resulting set of prototype vectors, one vector for each class, 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 cosines of the prototype vectors $\vec{c_j}$ with $\vec{d'}$ are calculated. d' is assigned to the class with which its document vector has the highest cosine.

$$H_{TFIDF}(d') = \operatorname*{argmax}_{C_j \in \mathcal{C}} cos(\vec{c_j}, \vec{d'}) \tag{5}$$

$\operatorname{argmax} f(x)$ returns the argument x for which $f(x)$ is maximum and $H_{TFIDF}(d')$ is the category to which the algorithm assigns document d'. The algorithm can be summarized in the following decision rule:

$$H_{TFIDF}(d') = \operatorname*{argmax}_{C_j \in \mathcal{C}} \frac{\vec{c_j}}{||\vec{c_j}||} \cdot \frac{\vec{d'}}{||\vec{d'}||} \tag{6}$$

$$= \operatorname*{argmax}_{C_j \in \mathcal{C}} \frac{\sum_{i=1}^{|F|} c_j^{(i)} \cdot d'^{(i)}}{\sqrt{\sum_{i=1}^{|F|} (c_j^{(i)})^2}} \tag{7}$$

In (7) the normalization with the length of the document vector is left out since it does not influence the argmax.

3.2.2 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.

In the following word-based unigram models of text will be used, i.e. words are assumed to occur independently of the other words in the document. There are $|\mathcal{C}|$ such models, one for each category. All documents assigned to a particular category are assumed to be generated according to the model associated with this category.

The following describes one approach to estimating $\Pr(C_j|d')$, the probability that a document d' is in class C_j. Bayes' rule says that to achieve the highest classification accuracy, d' should be assigned to the class for which $\Pr(C_j|d')$ is highest.

$$H_{BAYES}(d') = \operatorname*{argmax}_{C_j \in \mathcal{C}} \Pr(C_j|d') \tag{8}$$

$\Pr(C_j|d')$ can be split up by considering documents separately according to their length l.

$$\Pr(C_j|d') = \sum_{l=1}^{\infty} \Pr(C_j|d', l) \cdot \Pr(l|d') \tag{9}$$

$\Pr(l|d')$ equals one for the length l' of document d'

and is zero otherwise. After applying Bayes' theorem to $\Pr(C_j|d', l)$ we can therefore write:

$$\Pr(C_j|d') = \frac{\Pr(d'|C_j, l') \cdot \Pr(C_j|l')}{\sum_{C' \in \mathcal{C}} \Pr(d'|C', l') \cdot \Pr(C'|l')} \tag{10}$$

$\Pr(d'|C_j, l')$ is the probability of observing document d' in class C_j given its length l'. $\Pr(C_j|l')$ is the prior probability that a document of length l' is in class C_j. In the following we will assume, that the category of a document does not depend on its length, so $\Pr(C_j|l') = \Pr(C_j)$. An estimate $\hat{\Pr}(C_j)$ for $\Pr(C_j)$ can be calculated from the fraction of training documents that is assigned to class C_j.

$$\hat{\Pr}(C_j) = \frac{|C_j|}{\sum_{C' \in \mathcal{C}} |C'|} = \frac{|C_j|}{|D|} \tag{11}$$

$|C_j|$ denotes the number of training documents in class C_j and $|D|$ is the total number of documents.

The estimation of $\Pr(d'|C_j, l')$ is more difficult. $\Pr(d'|C_j, l')$ is the probability of observing a document d' in class C_j given that we consider only documents of length l'. Since there is - even for a simplifying representation as used here - 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 assumed to be generated. The unigram models introduced above imply that a word's occurrence is only dependent on the class the document comes from, but that it occurs independently[1] of the other words in the document and that it is not dependent on the document length. So $\Pr(d'|C_j, l')$ can be written as:

$$\Pr(d'|C_j, l') \approx \prod_{i=1}^{|d'|} \Pr(w_i|C_j) \tag{12}$$

w_i ranges over the sequence of words in document d' which are element of the feature vector F. $|d'|$ is the number of words in document d'. The estimation of $\Pr(d'|C_j)$ is reduced to estimating each $\Pr(w_i|C_j)$ independently. A Bayesian estimate is used for $\Pr(w_i|C_j)$.

$$\hat{\Pr}(w_i|C_j) = \frac{1 + TF(w_i, C_j)}{|F| + \sum_{w' \in |F|} TF(w', C_j)} \tag{13}$$

$TF(w, C_j)$ is the overall number of times word w occurs within the documents in class C_j. This estimator, which is often called the 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, since it does not falsely estimate probabilities to be zero.

[1] The weaker assumption of "linked-dependence" is actually sufficient [Cooper, 1991], but not considered here for simplicity.

The following is the resulting decision rule if equations (8), (10) and (12) are combined.

$$H_{BAYES}(d') = \operatorname*{argmax}_{C_j \in \mathcal{C}} \frac{\Pr(C_j) \cdot \prod_{i=1}^{|d'|} \Pr(w_i|C_j)}{\sum_{C' \in \mathcal{C}} \Pr(C') \cdot \prod_{i=1}^{|d'|} \Pr(w_i|C')} \tag{14}$$

$$= \operatorname*{argmax}_{C_j \in \mathcal{C}} \frac{\Pr(C_j) \cdot \prod_{w \in F} \Pr(w|C_j)^{TF(w,d')}}{\sum_{C' \in \mathcal{C}} \Pr(C') \cdot \prod_{w \in F} \Pr(w|C')^{TF(w,d')}} \tag{15}$$

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

4 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, and then show its relationship to the TFIDF algorithm. In terms of the design choices listed above I will show that the PrTFIDF algorithm is equivalent to a TFIDF classifier using the following settings:

- the word weighting mechanism uses a refined IDF weight especially adapted to text categorization,
- document length normalization is done using the number of words and
- the similarity measure is the inner product.

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 on text categorization.

4.1 The PrTFIDF Algorithm

The naive Bayes classifier proposed in the previous section provided an estimate of the probability $\Pr(C_j|d')$ that document d' is in class C_j, making the simplifying assumption of word independence. The PrTFIDF Algorithm uses a different way approximating $\Pr(C_j|d')$ inspired by the "retrieval with probabilistic indexing" (RPI) approach proposed in [Fuhr, 1989]. In this approach a set of descriptors X is used to represent the content of documents. A descriptor x is assigned to a document d with a certain probability $\Pr(x|d)$. So using the theorem of total probability (in line (16)) and Bayes' theorem (in line (17)) we can write

$$\Pr(C_j|d) = \sum_{x \in X} \Pr(C_j|x,d) \cdot \Pr(x|d) \quad (16)$$

$$= \sum_{x \in X} \frac{\Pr(d|C_j,x)}{\Pr(d|x)} \Pr(C_j|x) \cdot \Pr(x|d) \quad (17)$$

To make the estimation tractable the simplifying assumption that $\Pr(d|C_j,x) = \Pr(d|x)$ is made now.

$$\Pr(C_j|d) \approx \sum_{x \in X} \Pr(C_j|x) \cdot \Pr(x|d) \quad (18)$$

The validity of the assumption depends on the classification task and the choice of the set of descriptors X. It states that descriptor x provides enough information about d so that no information about document d is gained by taking its category C_j into account.

As mentioned above the set of descriptors X is part of the design. A pragmatic choice for X used in the following is to consider all bags with n words from the feature set F as potential descriptors; e.g., for $n = 3$ these are all bags containing three words from F. The number n of words is a parameter which controls the quality of the approximation versus the complexity of the estimation.

Another way of looking at equation (18), especially suited for the choice of X considered here, is the following. $\Pr(C_j|d)$ is approximated by the expectation of $\Pr(C_j|x)$, where x consists of a sequence of n words drawn randomly from document d. For both interpretations the underlying assumption is that text documents are highly redundant with respect to the classification task and that any sequence of n words from the document is equally sufficient for classification. For example, classifying documents according to whether they are cooking recipes or not, it is probably equally sufficient to know either of the sentences from the document. For $n = |d|$ $\Pr(C_j|d)$ equals $\Pr(C_j|x)$, but with decreasing n this simplifying assumption (like the independence assumption for the naive Bayes classifier) will be violated in practice. Nevertheless this simplification is worth trying as a starting point.

In the following the simplest case, namely $n = 1$, will be used and will lead to a TFIDF classifier like the one introduced in section 3.2.1. For $n = 1$ line (18) can be written as

$$\Pr(C_j|d) \approx \sum_{w \in F} \Pr(C_j|w) \cdot \Pr(w|d) \quad (19)$$

It remains to estimate the two probabilities from line (19). $\Pr(w|d)$ can be estimated from the representation of document d.

$$\hat{\Pr}(w|d) = \frac{TF(w,d)}{\sum_{w' \in F} TF(w',d)} = \frac{TF(w,d)}{|d|} \quad (20)$$

$|d|$ denotes the number of words in document d. $\Pr(C_j|w)$, the remaining part of equation (19), is the probability that C_j is the correct category of d given that we only know the randomly drawn word w from d. Bayes' formula can be used to rewrite $\Pr(C_j|w)$:

$$\Pr(C_j|w) = \frac{\Pr(w|C_j) \cdot \Pr(C_j)}{\sum_{C' \in \mathcal{C}} \Pr(w|C') \cdot \Pr(C')} \quad (21)$$

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

$$\hat{\Pr}(C_j) = \frac{|C_j|}{\sum_{C' \in \mathcal{C}} |C'|} = \frac{|C_j|}{|D|} \quad (22)$$

Finally $\Pr(w|C_j)$ can be estimated as

$$\hat{\Pr}(w|C_j) = \frac{1}{|C_j|} \sum_{d \in C_j} \hat{\Pr}(w|d) \quad (23)$$

The resulting decision rule for PrTFIDF is

$$H_{PrTFIDF}(d') = \underset{C_j \in \mathcal{C}}{\operatorname{argmax}} \sum_{w \in F} \frac{\Pr(w|C_j) \cdot \Pr(C_j)}{\sum_{C' \in \mathcal{C}} \Pr(w|C') \cdot \Pr(C')} \cdot \Pr(w|d') \quad (24)$$

4.2 The Connection between TFIDF and PrTFIDF

This section will show the relationship of the PrTFIDF classification rule to the TFIDF algorithm from section 3.2.1. In the following I will start with the decision rule for PrTFIDF and then transform it into the shape of a TFIDF classifier. From equation (24) we have

$$H_{PrTFIDF}(d') = \underset{C_j \in \mathcal{C}}{\operatorname{argmax}} \sum_{w \in F} \frac{\Pr(C_j) \cdot \Pr(w|C_j)}{\sum_{C' \in \mathcal{C}} \Pr(C') \cdot \Pr(w|C')} \cdot \Pr(w|d') \quad (25)$$

The term $\sum_{C' \in \mathcal{C}} \Pr(C') \cdot \Pr(w|C')$ in equation 25 can be re-expressed using a modified version of the inverse document frequency $IDF(w)$. The definition of inverse document frequency as stated in section 3.2.1 was

$$IDF(w) = log\left(\frac{|D|}{DF(w)}\right) \quad (26)$$

$$DF(w) = \sum_{d \in D} \begin{cases} 1 & d \texttt{ contains } w \\ 0 & \texttt{otherwise} \end{cases} \quad (27)$$

I now introduce a refined version of $IDF(w)$ suggested by the PrTFIDF algorithm.

$$IDF'(w) = sqrt\left(\frac{|D|}{DF'(w)}\right) \quad (28)$$

$$DF'(w) = \sum_{d \in D} \frac{TF(w,d)}{|d|} \quad (29)$$

There are two differences between this definition of $IDF(w)$ and the usual one. First, $DF'(w)$ is not the

$$H_{PrTFIDF}(d') = \operatorname*{argmax}_{C_j \in \mathcal{C}} \sum_{w \in F} \frac{|C_j|}{|D|} \cdot \left(\frac{1}{|C_j|} \cdot \sum_{d \in C_j} \frac{TF(w,d)}{|d|}\right) \cdot IDF'(w)^2 \cdot \frac{TF(w,d')}{|d'|} \quad (36)$$

$$= \operatorname*{argmax}_{C_j \in \mathcal{C}} \sum_{w \in F} \left(\frac{|C_j|}{|D|} \cdot \frac{1}{|C_j|} \cdot \sum_{d \in C_j} \frac{TF(w,d) \cdot IDF'(w)}{|d|}\right) \cdot \left(\frac{TF(w,d') \cdot IDF'(w)}{|d'|}\right) \quad (37)$$

number of documents with an occurrence of word w, but rather is the sum of the relative frequencies of w in each document. So $IDF'(w)$ can make use of frequency information instead of just considering binary occurrence information. 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 $DF'(w)$ will be. 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 the square root is used to dampen the effect of the document frequency instead of the logarithm. Nevertheless, both functions are similar in shape and reduce the impact of high document frequencies.

Replacing probabilities with their estimators, the expression $\sum_{C' \in \mathcal{C}} \Pr(C') \cdot \Pr(w|C')$ can be reduced to a function of $IDF'(w)$.

$$\sum_{C' \in \mathcal{C}} \Pr(C') \cdot \Pr(w|C') \quad (30)$$

$$= \sum_{C' \in \mathcal{C}} \frac{|C'|}{|D|} \cdot \frac{1}{|C'|} \cdot \sum_{d \in C'} \frac{TF(w,d)}{|d|} \quad (31)$$

$$= \sum_{C' \in \mathcal{C}} \frac{1}{|D|} \cdot \sum_{d \in C'} \frac{TF(w,d)}{|d|} \quad (32)$$

$$= \frac{\sum_{C' \in \mathcal{C}} \sum_{d \in C'} \frac{TF(w,d)}{|d|}}{|D|} \quad (33)$$

$$= \frac{\sum_{d \in D} \frac{TF(w,d)}{|d|}}{|D|} \quad (34)$$

$$= \frac{1}{IDF'(w)^2} \quad (35)$$

Using this and again substituting probabilities with their estimators, the decision rule can be rewritten as in (37) above. Extracting the prototype vector component and the document representation component we get to the decision rule

$$H_{PrTFIDF}(d') = \operatorname*{argmax}_{C_j \in \mathcal{C}} \frac{\vec{c_j}}{1} \cdot \frac{\vec{d'}}{|d'|} \quad (38)$$

$$\vec{c_j} = \frac{|C_j|}{|D|} \cdot \frac{1}{|C_j|} \cdot \sum_{d \in C_j} \frac{\vec{d}}{|d|} \quad (39)$$

$$d^{(i)} = TF(w_i, d) \cdot IDF'(w_i) \quad (40)$$

From the form of the decision rule in the previous lines it is easy to see that the PrTFIDF decision rule is equivalent with the TFIDF decision rule using the modified inverse document frequency weight $IDF'(w)$, the number of words as document length normalization and the inner product for measuring similarity. Furthermore it suggests how to set the parameters α and β. For each category $\alpha_j = \frac{|C_j|}{|D|}$ whereas $\beta = 0$.

4.3 Implications of the Analysis

The analysis shows how and under which preconditions the TFIDF classifier fits into a probabilistic framework. The PrTFIDF classifier offers a new view on the vector space model and the TFIDF word weighting heuristic for text categorization and advances the theoretical understanding of their interactions. The analysis also suggests improvements to the TFIDF algorithm and that the following changes should lead to a better classifier. PrTFIDF is an implementation of TFIDF incorporating these changes, namely

- Incorporation of prior probabilities $\Pr(C_j)$ through α.
- Use of $IDF'(w)$ for word weighting instead of $IDF(w)$.
- Use of the number of words for document normalization instead of the Euclidian length.
- Use of the inner product for computing similarity.

5 Experiments

The following experiments were performed to find out in how far the implications of the theoretical analysis lead to an improved classification algorithm in practice. The performances of PrTFIDF, TFIDF, and the naive Bayes classifier BAYES are compared on six categorization tasks.

5.1 Data Sets

5.1.1 Newsgroup Data

This data set consists of Usenet articles Ken Lang collected from 20 different newsgroups (table 1). 1000 articles were taken from each of the newsgroups, which

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

Table 1: Usenet newsgroups used in newsgroup data.

	PrTFIDF	BAYES	TFIDF
Newsgroups	91.8	89.6	86.3
“acq”	88.9	88.5	84.5
“wheat”	93.9	94.8	90.9
“crude”	90.2	95.5	85.4
“earn”	90.5	90.9	90.6
“cbond”	91.9	90.9	87.7

Table 2: Maximum accuracy in percentages.

makes a total 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 posted to[2].

The results reported on this dataset are averaged over a number of random test/training splits using binomial sign tests to estimate significance. In each experiment 33% of the data was used for testing.

5.1.2 Reuters Data

The Reuters-22173 data was collected by the Carnegie group from the Reuters newswire in 1987. Instead of averaging over all 135 categories, the following presents a more detailed analysis of five categories - namely the three most frequent categories (“earn”, “acq”, and “cbond”) and two categories with special properties (“wheat” and “crude”).

The “wheat” and the “crude” category have very narrow definitions. Classifying according to whether a document contains the word `wheat` yields an accuracy of 99.7% for the “wheat” category. The category “acq” (corporate acquisitions) for example does not have such an obvious definition. Its concept is more abstract and a number of words are reasonable predictors.

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

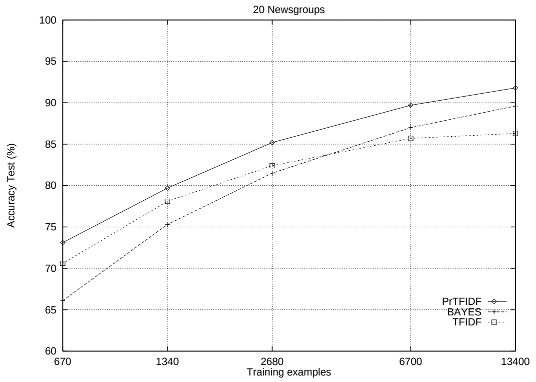

Figure 2: Accuracy versus the number of training examples on the newsgroup data.

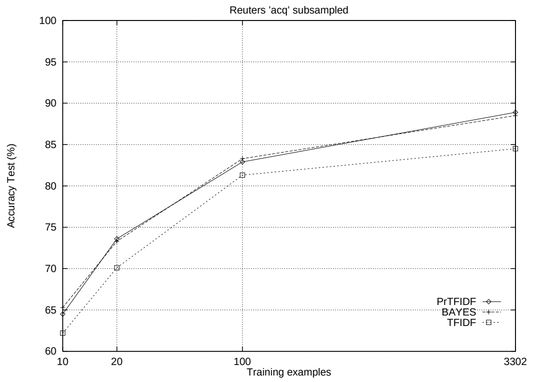

Figure 3: Accuracy versus the number of training examples on the Reuters categories “acq”.

In the following experiments articles which appeared on April 7, 1987 or before are in the training set. Articles which appeared later are in the test set. This results in a corpus of 14,704 training examples and 6,746 test examples. Since the TFIDF classifier does not have a principled way of dealing with uneven class distributions, to allow a fair comparison the data is subsampled randomly so that there is an equal number of positive and negative examples. The results presented here are averaged over a number of trials and binomial sign tests are used to estimate significance.

5.2 Experimental Results

Table 2 shows the maximum accuracy each learning method achieves. On the newsgroup data PrTFIDF performs significantly better than BAYES and BAYES is significantly better than TFIDF. Compared to TFIDF, PrTFIDF leads to a reduction of error of about 40%.

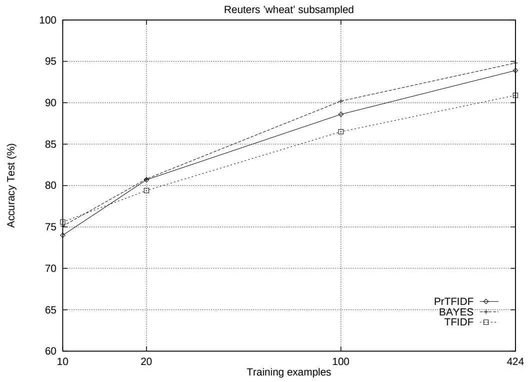

Figure 4: Accuracy versus the number of training examples on the Reuters categories "wheat".

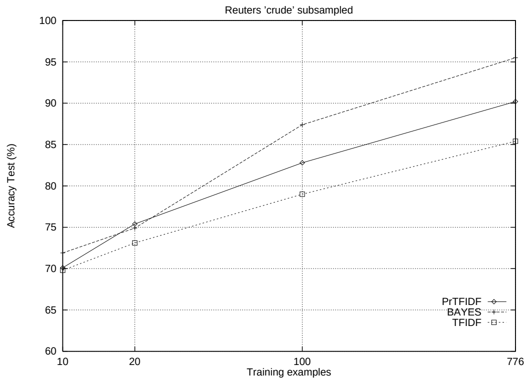

Figure 5: Accuracy versus the number of training examples on the Reuters categories "crude".

PrTFIDF and BAYES outperform TFIDF on the Reuters categories "acq", "wheat", "crude", and "cbond" as well. Comparing PrTFIDF and BAYES, BAYES tends to work better on the tasks where certain single keywords have very high prediction accuracy - namely the tasks "wheat" and "crude". The opposite is true for the PrTFIDF classifier. It achieves comparable performance or performance gains over BAYES on the categories "acq" and "cbond" as well as on the newsgroup data. This behaviour is interesting, since it is plausible given the different simplifying assumptions PrTFIDF and BAYES make. All classifiers perform approximately the same on the category "earn".

Figures 2 to 7 show accuracy in relation to the number of training examples. 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. In contrast to BAYES,

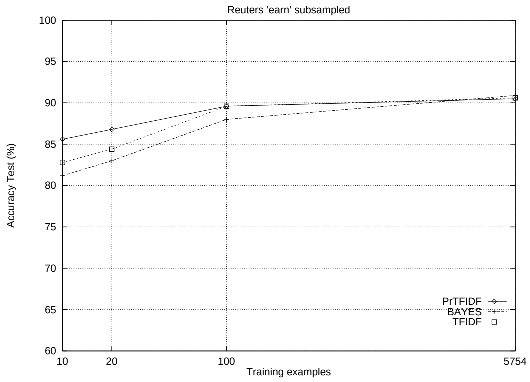

Figure 6: Accuracy versus the number of training examples on the Reuters categories "earn".

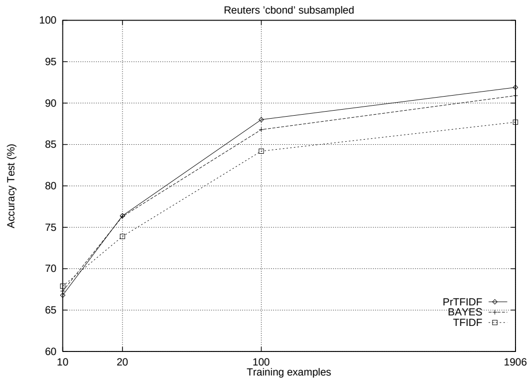

Figure 7: Accuracy versus the number of training examples on the Reuters categories "cbond".

PrTFIDF does particularly well in the newsgroup experiment (figure 2) for small numbers of training examples. The performance of BAYES approaches the one of PrTFIDF for high numbers, but stays below TFIDF for small training sets. The accuracy of the TFIDF classifier increases less steeply with the number of training examples compared to the probabilistic methods.

For the Reuters category "acq" BAYES and PrTFIDF show nearly identical curves (figure 3). TFIDF is significantly below the two probabilistic methods over the whole spectrum. For the tasks "wheat" (figure 4), "crude" (figure 5), and "cbond" (figure 7) all classifiers perform similar for small training sets and the difference generally increases with an increasing number of training examples.

6 Conclusions

This paper shows the relationship between text classifiers using the vector space model with TFIDF word weighting and probabilistic classifiers. It presents a probabilistic analysis of a particular TFIDF classifier and describes the algorithm using the same basic techniques from statistical pattern recognition that are used in probabilistic classifiers like BAYES. The analysis offers a theoretical explanation for the TFIDF word weighting heuristic in combination with the vector space retrieval model for text categorization and gives insight into the underlying assumptions.

Conclusions drawn from the analysis lead to the PrTFIDF classifier, which eliminates the inefficient parameter tuning and design choices of the TFIDF method. This makes the PrTFIDF classifier easy to use and empirical results on six classification tasks support its applicability on real world classification problems. Although the TFIDF method showed reasonable accuracy on all classification tasks, the two probabilistic methods BAYES and PrTFIDF showed performance improvements of up to 40% reduction of error rate on five of the six tasks. These empirical results suggest that a probabilistically founded modelling 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. The relaxation as well as the combination of those assumptions provide promising starting points for future research.

Acknowledgements

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

References

[Bookstein, 1982] A. Bookstein, *"Explanation and Generalization of Vector Models in Information Retrieval"*, in G. Salton, H. Schneider: 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, pages 292-300, 1994.

[Cooper, 1991] W. Cooper, *"Some Inconsistencies and Misnomers in Probabilistic Information Retrieval"*, International ACM SIGIR Conference, pages 57-61, 1991.

[Fuhr, 1989] N. Fuhr, *"Models for Retrieval with Probabilistic Indexing"*, 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.

[Joachims et al., 1997] T. Joachims, D. Freitag, T. Mitchell, *"WebWatcher: A Tour Guide for the World Wide Web"*, International Joint Conference on Artificial Intelligence (IJCAI), 1997.

[Lang, 1995] K. Lang, *"NewsWeeder: Learning to Filter Netnews"*, International Conference on Machine Learning, 1995.

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

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

[Salton, Buckley, 1988] G. Salton, C. Buckley, *"Term Weighting Approaches in Automatic Text Retrieval"*, Information Processing and Management, Vol. 24, No. 5, pages 513-523, 1988.

[Vapnik, 1982] V. Vapnik, *"Estimation of Dependencies Based on Empirical Data"*, Springer, 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, 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.

[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.