A Feature Weight Adjustment Algorithm for Document Categorization [*]

Shrikanth Shankar
University of Minnesota, Department of Computer Science
Minneapolis, MN 55455
shankar@cs.umn.edu

George Karypis
University of Minnesota, Department of Computer Science / Army HPC Research Center
Minneapolis, MN 55455
karypis@cs.umn.edu

ABSTRACT
In recent years we have seen a tremendous growth in the volume of text documents available on the Internet, digital libraries, news sources, and company-wide intra-nets. Automatic text categorization, which is the task of assigning text documents to pre-specified classes (topics or themes) of documents, is an important task that can help both in organizing as well as in finding information on these huge resources. In this paper we present a fast iterative feature weight adjustment algorithm for the linear-complexity centroid based classification algorithm. Our algorithm uses a measure of the discriminating power of each term to gradually adjust the weights of all features concurrently. We experimentally evaluate our algorithm on the Reuters-21578 and OHSUMED document collection and compare it against a variety of other categorization algorithms. Our experiments show that feature weight adjustment improves the performance of the centroid-based classifier by 2%–5% , substantially outperforms Rocchio and Widrow-Hoff and is competitive with SVM.

1. INTRODUCTION
We have seen a tremendous growth in the volume of online text documents available on the Internet, digital libraries, news sources and company-wide intra-nets. It has been forecasted that these documents along with other unstructured data will become the predominant data type stored online. Automatic text categorization [15, 21, 9] which is the task of assigning text documents to pre-specified classes of documents, is an important task that can help people find information from these huge resources. Text categorization presents huge challenges due to a large number of attributes, attribute dependency, multi-modality and large training set.

[*]This work was supported by NSF CCR-9972519, by Army Research Office contract DA/DAAG55-98-1-0441, by the DOE ASCI program, and by Army High Performance Computing Research Center contract number DAAH04-95-C-0008. Access to computing facilities was provided by AHPCRC, Minnesota Supercomputer Institute. Related papers are available via WWW at URL: http://www.cs.umn.edu/~karypis

The various document categorization algorithms that have been developed over the years [18, 2, 4, 5, 13, 9, 10, 3, 20, 11, 7] fall under two general categories. The first category contains traditional machine learning algorithms such as decision trees, rule sets, instance-based classifiers, probabilistic classifiers, support vector machines, *etc.*, that have either been used directly or being adapted for use in the context of document data sets. The second category contains specialized categorization algorithms developed in the Information Retrieval community. Examples of such algorithms include relevance feedback, linear classifiers, generalized instance set classifiers, *etc.* In the rest of this section we describe the classifiers we used and their implementation details.

A general class of algorithms that has been shown to produce good document categorization performance are similarity based. This class contains algorithms such as k-nearest neighbor [20], generalized instance set [10] and centroid based classifiers [6]. In these algorithms the class of a new document is determined by computing the similarity between the test document and individual instances or aggregates of the training set, and determining the class based on the class distribution of the nearest instances or aggregates.

A major drawback of these algorithms is that they use all the features while computing the similarity between a test document and the training set instances or aggregates. In many document data sets, only a relatively small number of the total features may be useful in categorizing documents, and using all the features may affect performance. A possible approach to overcome this problem is to learn weights for different features (*i.e.*, words). In this approach, each feature has a weight associated with it. A higher weight implies that this feature is more important for classification. When the weights are either 0 or 1 this approach become the same as feature selection. We refer to such algorithms as feature weight adjustment or just weight adjustment techniques.

In this paper we present a fast iterative feature weight adjustment algorithm for the linear-complexity centroid based classification algorithm. Our algorithm uses a measure of the discriminating power of each term to gradually adjust the weights of all features concurrently. Our analysis shows that this approach gradually eliminates the least discriminating features in each document thus improving its classification accuracy. We experimentally evaluate our algorithm on the Reuters-21578 [12] and OHSUMED [8] document collection and compare it against a variety of other categorization algorithms such as Rocchio [14], Widrow-Hoff [19] and Support vector machines [17, 9]. Our experiments show that feature weight adjustment improves the performance of the centroid-based classifier by 2%–5% , substantially outperforms Rocchio and Widrow-Hoff and is competitive with SVM.

The rest of this paper is organized as follows. Section 2 describes the centroid based classifier. Section 3 covers the feature weight adjustment algorithm. Section 4 gives the experimental results for this classifier and also compares its performance against other classifiers.

2. CENTROID-BASED DOCUMENT CLASSIFIER

In the centroid-based classification algorithm, the documents are represented using the vector-space model [15]. In this model, each document d is considered to be a vector in the term-space. In its simplest form, each document is represented by the *term-frequency* (TF) vector $\vec{d}_{tf} = (tf_1, tf_2, \ldots, tf_n)$, where tf_i is the frequency of the ith term in the document. A widely used refinement to this model is to weight each term based on its *inverse document frequency* (IDF) in the document collection. The motivation behind this weighting is that terms appearing frequently in many documents have limited discrimination power, and for this reason they need to be de-emphasized. This is commonly done [15] by multiplying the frequency of each term i by $\log(N/df_i)$, where N is the total number of documents in the collection, and df_i is the number of documents that contain the ith term (*i.e.*, document frequency). This leads to the *tf-idf* representation of the document, *i.e.*,

$$\vec{d}_{tfidf} = (tf_1 \log(N/df_1), tf_2 \log(N/df_2), \ldots, tf_n \log(N/df_n)).$$

Finally, in order to account for documents of different lengths, the length of each document vector is normalized so that it is of unit length, *i.e.*, $\|\vec{d}_{tfidf}\|_2 = 1$. In the rest of the paper, we will assume that the vector representation $\vec{d}$ of each document d has been weighted using *tf-idf* and it has been normalized so that it is of unit length.

In the vector-space model, the similarity between two documents d_i and d_j is commonly measured using the cosine function [15], given by

$$\cos(\vec{d_i}, \vec{d_j}) = \frac{\vec{d_i} \cdot \vec{d_j}}{\|\vec{d_i}\|_2 * \|\vec{d_j}\|_2}, \tag{1}$$

where "·" denotes the dot-product of the two vectors. Since the document vectors are of unit length, the above formula simplifies to $\cos(\vec{d_i}, \vec{d_j}) = \vec{d_i} \cdot \vec{d_j}$.

Given a set S of documents and their corresponding vector representations, we define the ***centroid*** vector $\vec{C}$ to be

$$\vec{C} = \frac{1}{|S|} \sum_{d \in S} \vec{d}, \tag{2}$$

which is nothing more than the vector obtained by averaging the weights of the various terms present in the documents of S. We will refer to the S as the ***supporting set*** for the centroid $\vec{C}$. Analogously to documents, the similarity between two centroid vectors and between a document and a centroid vector are computed using the cosine measure. In the first case,

$$\cos(\vec{C_i}, \vec{C_j}) = \frac{\vec{C_i} \cdot \vec{C_j}}{\|\vec{C_i}\|_2 * \|\vec{C_j}\|_2}, \tag{3}$$

whereas in the second case,

$$\cos(\vec{d}, \vec{C}) = \frac{\vec{d} \cdot \vec{C}}{\|\vec{d}\|_2 * \|\vec{C}\|_2} = \frac{\vec{d} \cdot \vec{C}}{\|\vec{C}\|_2}. \tag{4}$$

Note that even though the document vectors are of length one, the centroid vectors will not necessarily be of unit length.

The idea behind the centroid-based classification algorithm [6] is extremely simple. For each set of documents belonging to the same class, we compute their centroid vectors. If there are k classes in the training set, this leads to k centroid vectors $\{\vec{C}_1, \vec{C}_2, \ldots, \vec{C}_k\}$, where each $\vec{C}_i$ is the centroid for the ith class. The class of a new document x is determined as follows. First we use the document-frequencies of the various terms computed from the training set to compute the *tf-idf* weighted vector-space representation of x, and scale it so $\vec{x}$ is of unit length. Then, we compute the similarity between $\vec{x}$ to all k centroids using the cosine measure. Finally, based on these similarities, we assign x to the class corresponding to the most similar centroid. That is, the class of x is given by

$$\arg \max_{j=1,\ldots,k} (\cos(\vec{x}, \vec{C_j})). \tag{5}$$

The computational complexity of the learning phase of this centroid-based classifier is linear on the number of documents and the number of terms in the training set. The computation of the vector-space representation of the documents can be easily computed by performing at most three passes through the training set. Similarly, all k centroids can be computed in a single pass through the training set, as each centroid is computed by averaging the documents of the corresponding class. Moreover, the amount of time required to classify a new document x is at most $O(km)$, where m is the number of terms present in x. Thus, the overall computational complexity of this algorithm is very low, and is identical to fast document classifiers such as Naive Bayesian.

3. WEIGHT ADJUSTMENT FOR CENTROID BASED CLASSIFIER

Any scheme that adjusts the weights of the various features (*i.e.*, terms) has to perform two tasks. First, it must rank the various features according to their discriminating power. Second, it must adjust the weight of the various features in order to emphasize features with high discriminating power and/or de-emphasize features with none or limited discriminating power.

Over the years, a number of schemes have been developed to measure the discriminating power of the various features such as information gain, entropy , gini index , and χ^2 statistic . In our algorithm, the discriminating power of each feature is computed using a measure similar to the gini index [1], as follows. Let m be the number of different classes, and let $\{\vec{C}_1, \vec{C}_2, \ldots, \vec{C}_m\}$ be the centroid vectors of these classes. For each term i, let $\vec{T}_i = \{C_{1,i}, C_{2,i}, \ldots, C_{m,i}\}$ be the vector derived from the weight of the ith term in each one of the m centroids, and let $\vec{T'}_i = \vec{T}_i/||\vec{T}_i||_1$ be the one-norm scaled version of $\vec{T}_i$. The discriminating power of the ith term P_i is given by

$$P_i = \sum_{j=1}^{m} T_{j,i}^2, \tag{6}$$

which is nothing more than the square of the length of the $\vec{T'}_i$ vector. Note that the value of P_i is always in the range $[1/m, 1]$. The lowest value of P_i is achieved when $T'_{i,1} = T'_{i,2} = \cdots = T'_{i,m}$ *i.e.*, a term is equally distributed amongst all the classes; whereas the highest is achieved when the ith term occurs in only a single class. Thus, a value close to one indicates that the term has a high discriminating power, whereas a value close to $1/m$ indicates that the terms has little if any discriminating power. In the rest of this paper, we will refer to P_i as the ***purity*** of the ith term, and we will refer to the vector $\vec{P} = \{P_1, P_2, \ldots, P_n\}$ of the purities of all the n terms as the ***purity vector***.

Having ranked the various terms using the purity as a measure of their discriminating power, the next step is to adjust their weights so that terms with higher discriminating power become more important than terms with lower discriminating power. A simple way of doing this, is to scale each one of the terms according to their purity. In particular, each document vector $\vec{d}$ is transformed to a new vector $\vec{d'} = \{P_1 d_1, P_2 d_2, \ldots, P_i d_i, \ldots P_n d_n\}$. Given this set of transformed document vectors, the centroid classification algorithm will proceed to scale each document to be of unit length, and then build a new set of centroid vectors for the various classes. A new document will be classified by first scaling its terms according to the purity vector and then computing its similarity to the new set of centroids. Since the purity values are always less or equal to one, the weight of the various terms in each transformed document d' will always be equal or smaller than their original weights. However, the re-normalization operation performed by the centroid classification algorithm causes the purest terms in each document to actually gain weight, achieving the desired feature weight adjustments.

Unfortunately, this simple scheme has two drawbacks. The first is that this weight adjustment approach may cause too steep of a change in the weights of terms. When this happens the weight of a document tends to get concentrated into a very small number of terms. As a result there could a loss of information that can negatively affect the classification performance. The second is that in some cases, this simple one step processes may not sufficiently change the weights of the various terms. Consequently, the new representation will be similar to the original one, with almost no change in the classification accuracy. For this reason, our weight adjustment algorithm adopts a somewhat different approach that attempts to address these problems.

Our algorithm solves the first problem by changing the weights of the various features by a smaller factor than that indicated by their purity. In particular, for each term i, we scale its weight by $P_i^{1/\delta}$, where $\delta > 1$. Since, the purities are less than one, $P_i^{1/\delta}$ will be closer to one, thus leading to smaller changes. To address the second problem, we perform the weight-adjustment operation multiple times. For each data set, we use the classification accuracy on a portion of the training set (*i.e.*, validation set) in order to determine how many times to perform the weight adjustment. The weight-adjustment process is stopped when the classification performance on the validation set starts to decrease. The details of the algorithm are shown in Figure 1.

Once the number of weight adjustment iterations l has been computed, a new test document d is classified by first adjusting the weights of its terms by going through the same sequence of l weight adjustment iterations, and then using the centroid classification algorithm on the weight-adjusted training set to determine its class. This process can be speeded up by using the fact that applying l iterations of weight-adjustment followed by unit-length scaling is the same as applying a single weight-adjustment in which the weight of each term j is multiplied by $P_j^{l/\delta}$, followed by a single unit-length scaling ([16]).

Note that each iteration of the weight adjustment algorithms is linear to the number of non-zeros in the document term matrix. Furthermore the number of iterations is quite small (less than 20 in all of our experiments), leading to an essentially linear complexity algorithm.

Discussion

One way of understanding this proposed weight adjustment scheme is to focus on how it modifies the weights of the various terms in a particular document. In the initial iterations as a result of the unit-length renormalizations an impure term may gain weight due to the presence of other terms of even lesser purity. However as more iterations are performed, the weight transfer process causes these terms to have lesser weight and thus reduces the weight transfer into higher terms. As a result of this process, initally only the terms having the lowest purity in the document will lose weight. As these lose weight, terms which are more pure will no longer be able to compensate their loss of weight from these terms and will also start losing weight. Thus the weight of each term will have a curve that looks like Figure 3 (B). The term having low purity (Figure 3(A)) does not show the initial increase while the purest term (Figure 3(C)) does not exhibit the final falling part. The figures shows the change in the weight of a 3 terms with different purities in the same document for 10 iterations.

3.1 Binary Classification

A common classification problem in information retrieval is that of developing a classifier that can correctly identify documents that belong to a particular *target* class from a large collection of documents. This is a typical binary classification problem in which we try to develop a model for the *target* class versus the *rest*. The weight adjustment scheme described in the previous section can be directly used in this kind of problems.

```
1.   Split the training set $A$ into training $A'$ and validation $V$
2.   Compute the accuracy on $V$ of the centroid classifier built on $A'$
3.   Compute $\vec{P}$ using the documents in $A'$
4.   $l = 0$
5.   For each document $d_i \in A$
6.       For each term $j$
7.           $d_{i,j} = P_j^{1/\delta} d_{i,j}$
8.       Scale $d_i$ so that $||d_i||_2 = 1$
9.   Compute the accuracy on $V$ of the centroid classifier built on $A'$
10.  If accuracy does not decrease
11.      $l = l + 1$
12.      Goto 5
```

Figure 1: The fixed weight adjustment algorithm.

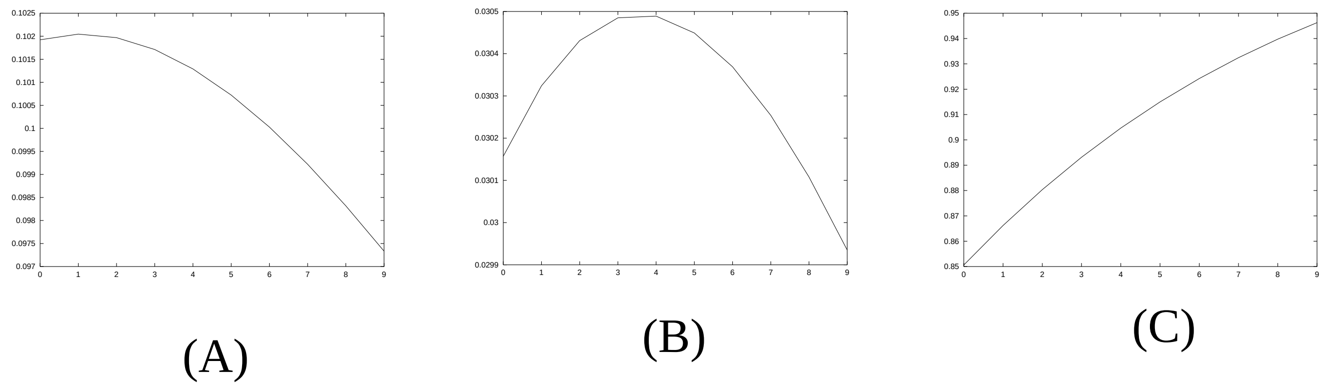

Figure 2: Column weight against number of iterations

The analysis presented in [6] suggest that the centroid scheme does best when each class contains related documents. The negative class *i.e.*, *rest* is however too diffuse for this. As a result instances of the negative class tend to get classified as positive. We propose the following solution to handle this. We cluster the negative set into k clusters. While computing the centroids and the purity we treat this as a $k + 1$ class problem. When classifying, we compute the similarity of the document to the k negative clusters. We take the largest value amongst these, and treat it as the similarity of the document to the negative set. The similarity of the document to the positive set is directly computed from the centroid of the positive class. Similarly if we find that a class has a multi-modal distribution we can run a clustering algorithm to identify sub-classes within it. We could then treat each of these sub-classes as a separate class.

4. EXPERIMENTAL SETUP AND RESULTS

In this section we experimentally evaluate the effect of using our feature weight adjustment algorithm on the classification accuracy of a centroid based classifier. Two different sets of experiments are presented. Both experiments were targeted on problems in which each document could have multiple class labels. Accuracy was measured by considering binary classification for each individual class. The effectiveness of classification was computed using precision and recall at the break even point.

In our experiments, we compared the performance of our weight adjustment scheme against the performance achieved by the following classifiers. We obtained results using two linear classifiers which used Rocchio and Widrow-Hoff respectively to learn the weight vectors. In the case of Rocchio we used $\alpha = 0, \beta = 16, \gamma = 4$ [13], whereas in WH we used $\eta = 0.5$. We also ran SVM^{lite} [9] using a polynomial kernel with d = 1 and a RBF kernel with $\gamma = 0.8$ against the data sets. Also the parameter δ in the feature weight adjustment algorithm is set to 8.

4.1 Reuters

The first data set that we used was the Reuters-21578 [12] text collection. In particular, we used the "ModApte" split to divide the text collection into a set of 9603 training documents and 3299 test documents. After eliminating stopwords and removing terms that occur less than two times, the training corpus contains 11,001 distinct terms.

Table 1 shows the performance of the classifiers on the 10 most frequent classes in the Reuters data set. The columns labeled "Rocchio" and "WH" shows the performance achieved by a linear classifier using the Rocchio and Widrow-Hoff algorithms respectively to learn the weights. The next two columns show the performance of the SVM classifier using a degree one polynomial kernel and a RBF kernel. The fifth column labeled "Centroid" shows the performance of the centroid classifier. The last column shows the performance of the centroid classifier after feature weights have been adjusted by our feature-weight adjustment (FWA) algorithm. Note that this algorithm was run without using clustering. Table 1 also shows the the micro-average [20] over all classes, the micro-average over the top 10 classes and the macro-average over the top 10 classes.

A number of interesting observations can be made from the results in this Table 1. First, comparing Rocchio, Widrow-Hoff and the basic centroid scheme (the three fastest schemes), we see that overall the centroid scheme performs substantially better than the rest followed by WH and then Rocchio. In 6 of the top 10 categories the centroid scheme does best with WH dominating in the remaining 4. Second, we see that the weight adjustment scheme improves the performance of the centroid classifier, sometimes dramatically. Third, SVM using a RBF kernel is the overall winner doing about 5% better than the other schemes.

In addition to this we also tested the effect of clustering of the negative set (as described in section 3.1). These results are presented in table 2 for 5, 10, 15 and 20 clusters. As can be seen clustering has a dramatic improvement in the performance of the scheme. The number of clusters only slightly affects overall performance but using 10 clusters gives the best results. Comparing the results after clustering with the SVM results we see that the SVM (poly) scheme now has an overall micro-average about one percent less than using weight adjustment. SVM (rbf) does better by about 2% now. The weight adjustment scheme dominate SVM(rbf) in 3 of the top 10 classes.

	FWA			
	5	10	15	20
earn	95.58	95.76	94.94	94.66
acq	94.02	94.16	91.93	92.9
money-fx	72.07	77.1	77.1	77.65
grain	85.23	91.28	87.92	88.59
crude	85.71	84.66	86.24	86.24
trade	77.78	79.49	79.49	78.63
interest	71.76	73.28	74.05	74.05
wheat	87.32	87.32	85.92	85.92
ship	82.02	85.39	84.27	84.27
corn	76.79	87.5	85.71	87.5
Micro-average (top 10)	89.56	90.71	89.67	89.88
Micro-average (all)	83.16	84.69	84.25	84.33
Average (top 10)	82.83	85.59	84.76	85.04

Table 2: Effect of clustering

4.2 OHSUMED results

Table 3 gives the same data in Table 1 for the OHSUMED [8] data set. We used from the OHSUMED data, those documents with id's between 100000 and 120000 which had either the title or both the title and the abstract. The classification task considered here is to assign the document to one or multiple categories of the 23 MeSH "diseases" categories.. There were 19858 entries in the data set. The first 12000 of these were used in the training set and the remaining formed the test set. A total of 6561 documents did not have class label assigned to them. 6199 documents belonged to multiple classes.

Once again comparing Rocchio, Widrow-Hoff and the centroid scheme we see that the centroid scheme performs the best among the three on this data set. Rather surprisingly Widrow-Hoff has a very poor performance on this data set and is dominated completely by Rocchio, performing better in only 2 of the 23 categories. The centroid based scheme dominates both of them in all 23 categories. The weight adjustment scheme again improves the accuracy of basic centroid scheme by about 3%. Even without clustering the scheme achieves a higher micro-average than the SVM(poly) scheme and perform better than it in 16 of the 23 classes. SVM(rbf) again performs the best. It achieves a micro-average about 5% higher than the FWA scheme and performs better in 22 of the 23 categories.

The results for the weight-adjustment schemes after clustering are shown in Table 4. Clustering has almost no effect on accuracy. In fact in some cases it actually reduces accuracy. Overall using 5 clusters gives the best result for this data set and the results are about 4%–5% lower than those for the SVM(rbf) scheme. One interesting result in this table is that each of the different levels of clustering improves accuracy in some of the data sets For example, for FWA no clustering gives the best results for classes *c23* and *c0*, while using 5 clusters gives the best result for classes *c14* and *c04*. Similarly trends can be seen in the other classes.

	FWA			
	5	10	15	20
c23	50.19	43.37	42.73	41.14
c20	59.97	55.38	48.16	42.26
c14	71.99	69.98	65.74	65.29
c04	60.21	56.06	55.88	47.23
c06	64.96	64.07	62.48	58.23
c21	61.05	56.29	55.34	54.63
c10	54.79	54.11	46.92	45.21
c08	53.36	50.00	48.66	51.01
c19	69.20	70.25	70.04	63.71
c17	58.66	58.66	56.54	56.18
c01	56.71	56.20	60.76	57.97
c05	53.01	51.37	50.82	48.09
c13	52.66	52.37	55.03	54.44
c12	54.11	54.34	55.71	55.38
c15	49.72	45.86	43.82	41.98
c16	51.88	54.27	49.83	49.49
c18	40.17	37.18	36.32	29.49
c11	60.00	58.18	60.91	62.73
c07	42.11	40.79	42.11	40.79
c09	64.57	61.42	65.35	66.93
c22	17.91	17.91	14.93	13.43
c03	42.86	41.18	47.90	48.74
c02	43.90	47.56	48.78	53.75
average	53.79	52.03	51.51	49.92
micro-average	57.51	54.57	53.21	50.78

Table 4: OHSUMED clustering

4.3 Efficiency

One of the advantages of our weight adjusted centroid scheme is its speed. As discussed in Section 3 model learning time is linear in the number of non zero terms in the document-term matrix and classification time is linear in number of classes. A comparison of running time was performed between the svm_lite [9] code with the polynomial kernel, the RBF kernel and the centroid based scheme with FWA and is reported in Table 5. These times were obtained on a P3

	Rocchio	WH	SVM(poly)	SVM(rbf)	Centroid	FWA
earn	96.23	95.86	98.62	98.71	93.74	96.32
acq	79.00	87.60	91.24	95.27	91.79	91.93
money-fx	55.31	67.04	70.39	78.21	63.68	66.48
grain	77.85	79.19	91.94	93.29	77.85	77.85
crude	75.66	72.49	86.77	89.42	85.71	84.12
trade	73.5	68.38	70.94	76.92	77.78	77.78
interest	70.23	66.41	63.36	74.81	75.56	75.56
wheat	74.65	85.92	78.87	84.51	74.65	80.28
ship	79.77	73.03	77.53	85.39	85.39	84.27
corn	60.71	64.29	80.36	85.71	62.5	62.5
Micro-average (top 10)	82.81	85.25	89.37	92.5	87.01	88.23
Micro-average (all)	76.73	76.57	83.49	86.62	80.62	81.4
Average (top 10)	74.29	76.02	81.00	86.22	78.87	79.71

Table 1: Precision / Recall break even points on Reuters

	rocc	wh	SVM(poly)	SVM(rbf)	centroid	FWA
c23	38.65	42.16	47.70	56.38	51.59	53.19
c20	46.72	37.80	57.48	70.08	53.81	64.83
c14	58.26	56.03	67.41	75.11	69.53	71.99
c04	48.96	46.19	51.21	64.19	59.34	60.21
c06	47.26	38.23	59.82	68.32	62.83	66.37
c21	47.51	42.04	52.02	58.57	57.96	60.10
c10	31.16	25.00	45.55	57.53	41.10	51.71
c08	43.96	37.92	51.34	58.72	52.35	54.36
c19	54.85	36.08	62.45	73.20	60.13	61.39
c17	31.45	20.49	50.53	59.72	51.94	58.30
c01	40.25	36.96	52.66	61.52	52.15	52.15
c05	29.51	24.59	46.45	56.83	42.08	43.17
c13	39.65	34.62	50.89	61.83	51.48	53.25
c12	46.12	38.36	52.97	61.64	50.91	51.37
c15	16.57	7.73	46.41	53.89	40.33	51.93
c16	43.34	33.10	47.44	58.36	50.85	52.21
c18	23.08	9.40	41.45	45.30	27.35	32.05
c11	52.73	14.55	60.91	66.36	62.73	63.63
c07	35.52	19.74	36.84	46.05	42.11	42.11
c09	56.69	24.41	63.78	66.93	58.27	62.20
c22	13.43	20.90	13.64	16.42	17.91	16.42
c03	36.13	21.85	42.86	50.42	37.81	37.82
c02	34.15	12.20	51.22	56.10	50.00	50.00
average	39.82	29.58	50.13	58.41	49.76	52.64
micro-average	43.30	36.98	53.12	62.23	53.89	57.04

Table 3: OHSUMED results

500MHz workstation running Redhat 6 with 1 Gig of memory, under similar load conditions. Looking at this table we can see that FWA is about 2–10 times faster than SVM(rbf) in the learning phase and about 10–20 times faster in the classification phase.

5. CONCLUSION

In this paper we showed how a weight adjustment scheme improves the accuracy of a centroid based classifier. This scheme retains the power of the centroid based classifier while further enhancing its ability. Also it retains much of the speed of the original scheme. In terms of future work, clustering has been shown to be useful in improving the accuracy of this scheme. Clustering is needed in order to handle multi-modal distributions which this scheme cannot handle in its current form. Automatically determining whether a class needs to be clustered and how many clusters it should be divided into would be an interesting problem. As it stands the analysis of the algorithm is still incomplete. It would be beneficial to have a more rigorous analysis of this scheme and its strengths and weaknesses.

6. REFERENCES

[1] C. C. Aggarwal, S. C. Gates, and P. S. Yu. On the merits of building categorization systems by supervised clustering. In *Proc. of the Fifth ACM SIGKDD Int'l Conference on Knowledge Discovery and Data Mining*, pages 352–356, 1999.

[2] M. B. Amin and S. Shekhar. Generalization by neural networks. *Proc. of the 8th Int'l Conf. on Data Eng.*, April 1992.

[3] L. Baker and A. McCallum. Distributional clustering of words for text classification. In *SIGIR-98*, 1998.

[4] D. S. D. Michie and C. Taylor. *Machine Learning, Neural and Statistical Classification.* Ellis Horwood, 1994.

[5] D. E. Goldberg. *Genetic Algorithms in Search, Optimizations and Machine Learning.* Morgan-Kaufman, 1989.

[6] E. Han and G. Karypis. Centroid-based document classification algorithms: Analysis & experimental results. In *European Conference on Principles of Data Mining and*

	SVM(poly)		SVM(rbf)		FWA (10 clusters)	
	learn	classify	learn	classify	learn	classify
earn	66	17	136	36	19	1
acq	103	17	190	42	20	1
money-fx	125	10	144	23	24	1
grain	33	8	71	19	29	1
crude	46	9	78	20	21	1
trade	63	9	88	23	22	2
interest	146	7	119	15	23	1
wheat	39	5	55	11	24	2
ship	30	5	59	15	25	1
corn	25	6	41	11	31	1

Table 5: Run time comparison (all times in seconds)

Knowledge Discovery (PKDD), 2000. Also available on the WWW at URL *http://www.cs.umn.edu/˜karypis*.

[7] E.-H. Han. *Text Categorization Using Weight Adjusted k-Nearest Neighbor Classification*. PhD thesis, University of Minnesota, October 1999.

[8] W. Hersh, C. Buckley, T. Leone, and D. Hickam. OHSUMED: An interactive retrieval evaluation and new large test collection for research. In *SIGIR-94*, pages 192–201, 1994.

[9] T. Joachims. Text categorization with support vector machines: Learning with many relevant features. In *Proc. of the European Conference on Machine Learning*, 1998.

[10] W. Lam and C. Y. Ho. Using a generalized instance set for automatic text categorization. In *SIGIR-98*, 1998.

[11] B. Larsen and C. Aone. Fast and effective text mining using linear-time document clustering. In *Proc. of the Fifth ACM SIGKDD Int'l Conference on Knowledge Discovery and Data Mining*, pages 16–22, 1999.

[12] D. D. Lewis. Reuters-21578 text categorization test collection distribution 1.0. *http://www.research.att.com/∼lewis*, 1999.

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

[14] J. J. Rocchio. The SMART retrieval system: Experiments in automatic document processing. In G. Salton, editor, *Relevance feedback in information retrieval*. Prentice-Hall, Inc., 1971.

[15] G. Salton. *Automatic Text Processing: The Transformation, Analysis, and Retrieval of Information by Computer*. Addison-Wesley, 1989.

[16] S. Shankar. Feature weight adjustment schemes for centroid based classifiers. Master's thesis, Department of Computer Science, University of Minneapolis, Minneapolis, MN, 2000.

[17] V. Vapnic. *The Nature of Statistical Learning Theory*. Springer, 1995.

[18] S. Weiss and C. A. Kulikowski. *Computer Systems that Learn: Classification and Prediction Methods from Statistics, Neural Nets, Machine Learning, and Expert Systems*. Morgan Kaufmann, San Mateo, CA, 1991.

[19] B. Widrow and S. Stearns. *Adaptive Signal Processing*. Prentic-Hall, Inc., 1985.

[20] Y. Yang and X. Liu. A re-examination of text categorization methods. In *SIGIR-99*, 1999.

[21] Y. Yang and J. Pederson. A comparative study on feature selection in text categorization. In *Proc. of the Fourteenth International Conference on Machine Learning*, 1997.