Centroid-Based Document Classification: Analysis & Experimental Results*

Eui-Hong (Sam) Han and George Karypis

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

Abstract. In this paper we present a simple linear-time centroid-based document classification algorithm, that despite its simplicity and robust performance, has not been extensively studied and analyzed. Our experiments show that this centroid-based classifier consistently and substantially outperforms other algorithms such as Naive Bayesian, k-nearest-neighbors, and C4.5, on a wide range of datasets. Our analysis shows that the similarity measure used by the centroid-based scheme allows it to classify a new document based on how closely its behavior matches the behavior of the documents belonging to different classes. This matching allows it to dynamically adjust for classes with different densities and accounts for dependencies between the terms in the different classes.

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 intranets. It has been forecasted that these documents (with other unstructured data) will become the predominant data type stored online. Automatic text categorization [20, 16, 12, 4, 8], which is the task of assigning text documents to pre-specified classes (topics or themes) of documents, is an important task that can help people to find information on these huge resources. Text categorization presents unique challenges due to the large number of attributes present in the data set, large number of training samples, attribute dependency, and multi-modality of categories. This has led to the development of a variety of text categorization algorithms [8, 9, 1, 20] that address these challenges to varying degrees.

In this paper we present a simple centroid-based document classification algorithm. In this algorithm, a centroid vector is computed to represent the documents of each class, and a new document is assigned to the class that corresponds to its most similar centroid vector, as measured by the cosine function. Extensive experiments presented in Section 3 show that this centroid-based classifier consistently and substantially outperforms other algorithms such as Naive Bayesian [12], k-nearest-neighbors [20], and

* 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

C4.5 [15], on a wide range of datasets. Our analysis shows that the similarity measure used by the centroid-based scheme allows it to classify a new document based on how closely its behavior matches the behavior of the documents belonging to different classes. This matching allows it to dynamically adjust for classes with different densities and accounts for dependencies between the terms in the different classes. We believe that this feature is the reason why it consistently outperforms other classifiers that cannot take into account these density differences and dependencies.

The reminder of the paper is organized as follows. Section 3 experimentally evaluates this algorithm on a variety of data sets. Section 4 analyzes the classification model of the centroid-based classifier and compares it against those used by other algorithms. Finally, Section 5 provides directions for future research.

2 Centroid-Based Document Classifier

In the centroid-based classification algorithm, the documents are represented using the vector-space model [16]. 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 $\boldsymbol{d}_{tf} = (tf_1, tf_2, \ldots, tf_n)$, 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. This is commonly done [16] 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., $\boldsymbol{d}_{tfidf} = (tf_1 \log(N/df_1), tf_2 \log(N/df_2), \ldots, tf_n \log(N/df_n))$. 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., $\|\boldsymbol{d}_{tfidf}\|_2 = 1$. In the rest of the paper, we will assume that the vector representation $\boldsymbol{d}$ 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 [16], given by

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

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

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

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

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 $\boldsymbol{C}$. Analogously to documents, the similarity between between a document and a centroid vector is computed using the cosine measure as follows:

$$\cos(\boldsymbol{d}, \boldsymbol{C}) = \frac{\boldsymbol{d} \cdot \boldsymbol{C}}{\|\boldsymbol{d}\|_2 * \|\boldsymbol{C}\|_2} = \frac{\boldsymbol{d} \cdot \boldsymbol{C}}{\|\boldsymbol{C}\|_2}. \tag{3}$$

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 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 $\{\boldsymbol{C_1}, \boldsymbol{C_2}, \ldots, \boldsymbol{C_k}\}$, where each $\boldsymbol{C_i}$ 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 $\boldsymbol{x}$ is of unit length. Then, we compute the similarity between $\boldsymbol{x}$ 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

$$\arg \max_{j=1,\ldots,k} (\cos(\boldsymbol{x}, \boldsymbol{C_j})). \tag{4}$$

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 Experimental Results

We evaluated the performance of the centroid-based classifier by comparing against the naive Bayesian, C4.5, and k-nearest-neighbor classifiers on a variety of document collections. We obtained the naive Bayesian results using the Rainbow [13] with the multinomial event model [12]. The C4.5 results were obtained using a locally modified version of the C4.5 algorithm capable of handling sparse data sets. Finally, the k-nearest-neighbor results were obtained by using the *tf-idf* vector-space representation of the documents (identical to that used by the centroid-based classification algorithm), and using the number of neighbors $k = 10$.

3.1 Document Collections

We have evaluted the classification algorithms on data sets from West Group [5], TREC-5 [18], TREC-6 [18], and TREC-7 [18] collections, Reuters-21578 text categorization test collection Distribution 1.0 [11], OHSUMED collection [7], and the WebACE project (WAP) [2]. The detailed characteristics of the various document collections used in our experiments are available in [6] [1]. Note that for all data sets, we used a stop-list to remove common words, and the words were stemmed using Porter's suffix-stripping algorithm [14]. Furthermore, we selected documents such that each document has only one class (or label). In other words, given a set of classes, we collected documents that have only one class from the set.

3.2 Classification Performance

The classification accuracy of the various algorithms on the different data sets in our experimental testbed are shown in Table 1. These results correspond to the average classification accuracies of 10 experiments. In each experiment 80% of the documents were randomly selected as the training set, and the remaining 20% as the test set. The first three rows of this table show the results for the naive Bayesian, C4.5, and k-nearest neighbor schemes, whereas the last row shows the results achieved by the centroid-based classification algorithm (denoted as "Cntr" in the table). For each one of the data sets, we used a boldface font to highlight the algorithm that achieved the highest classification accuracy.

Table 1. The classification accuracy achieved by the different classification algorithms.

	west1	west2	west3	oh0	oh5	oh10	oh15	re0	re1	tr11	tr12	tr21
NB	86.7	76.5	75.1	89.1	87.1	81.2	84.0	**81.1**	**80.5**	85.3	79.8	59.6
C4.5	85.5	75.3	73.5	82.8	79.6	73.1	75.2	75.8	77.9	78.2	79.2	81.3
kNN	82.9	77.2	76.1	84.4	85.6	77.5	81.7	77.9	78.9	85.3	85.7	89.2
Cntr	**87.5**	**79.0**	**81.6**	**89.3**	**88.2**	**85.3**	**87.4**	79.8	80.4	**88.2**	**90.3**	**91.6**

	tr23	tr31	tr41	tr45	la1	la2	la12	fbis	wap	ohscal	new3
NB	69.3	94.1	94.5	84.7	**87.6**	**89.9**	**89.2**	77.9	80.6	74.6	74.4
C4.5	**90.7**	93.3	89.6	91.3	75.2	77.3	79.4	73.6	68.1	71.5	73.5
kNN	81.7	93.9	93.5	91.1	82.7	84.1	85.2	78.0	75.1	62.5	67.9
Cntr	85.2	**94.9**	**95.7**	**92.9**	87.4	88.4	89.1	**80.1**	**81.3**	**75.4**	**79.7**

Looking at the results of Table 1, we can see that naive Bayesian outperforms the other schemes in five out of the 23 data sets, C4.5 does better in one, the centroid-based scheme does better in 17, whereas the k-nearest-neighbor algorithm never outperforms the other schemes.

A more accurate comparison of the different schemes can be obtained by looking at what extent the performance of a particular scheme is statistically different from that of another scheme. We used the resampled paired t test [6] to compare the accuracy results obtained by the different classifiers. The statistical significance results using the resampled paired t test are summarized in Table 2, in which for each pair of classification algorithms, it shows the number of data sets that one performs statistically better,

[1] These data sets are available from *http://www.cs.umn.edu/˜han/data/tmdata.tar.gz*.

worse, or similarly than the other. Looking at this table, we can see that the centroid-based scheme compared to naive Bayesian, does better in ten data sets, worse in one data set, and they are statistically similar in twelve data sets. Similarly, compared to kNN, it does better in twenty, and it is statistically similar in three data sets. Finally, compared to C4.5, the centroid-based scheme does better in eighteen, worse in one, and statistically similar in four data sets.

Table 2. Statistical comparison of different classification algorithms using the resampled paired t test. The entries in the table show the number of data sets that the classifier in the row performs better, worse or similarly than the classifier in the column.

	NB	kNN	C4.5
Cntr	10/1/12	20/0/3	18/1/4
NB		12/4/7	15/3/5
kNN			13/3/7

From these results, we can see that the simple centroid-based classification algorithm outperforms all remaining schemes, with naive Bayesian being second, k-nearest-neighbor being third, and C4.5 being the last. Note that the better performance of NB and kNN over decision tree classification algorithms such as C4.5 agrees to results reported in [3, 20] using precision and recall of binary classification.

Recently, Support Vector Machines (SVM) has been shown to be very effective in text classification [8]. We were not able to directly compare the centroid-based scheme with the SVM, because the SVM code used in [8] was written for binary classification only. We plan to perform comparison studies between SVM and the centroid-based scheme by performing binary classification in the future.

4 Analysis

4.1 Classification Model

The surprisingly good performance of the centroid-based classification scheme suggests that it employs a sound underlying classification model. The goal of this section is to understand this classification model and compare it against those used by other schemes.

In order to understand this model we need to understand the formula used to determine the similarity between a document x, and the centroid vector $\boldsymbol{C}$ of a particular class (Equation 3), as this computation is essential in determining the class of x (Equation 4). From Equation 3, we see that the similarity (i.e., cosine) between $\boldsymbol{x}$ and $\boldsymbol{C}$ is the ratio of the dot-product between $\boldsymbol{x}$ and $\boldsymbol{C}$ divided by the length of $\boldsymbol{C}$. If S is the set of documents used to create $\boldsymbol{C}$, then from Equation 2, we have that:

$$\boldsymbol{x} \cdot \boldsymbol{C} = \boldsymbol{x} \cdot \left(\frac{1}{|S|} \sum_{d \in S} \boldsymbol{d} \right) = \frac{1}{|S|} \sum_{d \in S} \boldsymbol{x} \cdot \boldsymbol{d} = \frac{1}{|S|} \sum_{d \in S} \cos(\boldsymbol{x}, \boldsymbol{d}).$$

That is, the dot-product is the average similarity (as measured by the cosine function) between the new document x and all other documents in the set. The meaning of the length of the centroid vector can also be easily understood using the fact that $\|\boldsymbol{C}\|_2 = \sqrt{\boldsymbol{C} \cdot \boldsymbol{C}}$. Then, from Equation 2 we have that:

$$\|\boldsymbol{C}\|_2 = \sqrt{\left(\frac{1}{|S|}\sum_{d \in S} \boldsymbol{d}\right) \cdot \left(\frac{1}{|S|}\sum_{d \in S} \boldsymbol{d}\right)} = \sqrt{\frac{1}{|S|^2}\sum_{d_i \in S}\sum_{d_j \in S} \cos(\boldsymbol{d_i}, \boldsymbol{d_j})}.$$

Hence, the length of the centroid vector is the square-root of the average pairwise similarity between the documents that support the centroid.

In summary, the classification model used by the centroid-based document assigns a test document to the class whose documents better match the behavior of the test document, as measured by average document similarities. It computes the average similarity between the test document and all the other documents in that class, and then it amplifies that similarity, based on how similar to each other are the documents of that class. If the average pairwise similarity between the documents of the class is small (i.e., the class is *loose*), then that amplification is higher, whereas if the average pairwise similarity is high (i.e., the class is *tight*), then this amplification is smaller. More detailed ananlysis can be found in [6].

4.2 Comparison With Other Classifiers

One of the advantages of the centroid-based scheme is that it summarizes the characteristics of each class, in the form of the centroid vector. A similar summarization is also performed by naive Bayesian, in the form of the per-class term-probability distribution functions.

The advantage of the summarization performed by the centroid vectors is that it combines multiple prevalent features together, even if these features are not simultaneously present in a single document. That is, if we look at the prominent dimensions of the centroid vector (i.e., highest weight terms), these will correspond to terms that appear frequently in the documents of the class, but not necessarily all in the same set of documents. This is particularly important for high dimensional data sets for which the coverage of any individual feature is often quite low. Moreover, in the case of documents, this summarization has the additional benefit of addressing issues related to synonyms, as commonly used synonyms will be represented in the centroid vector (see [6] for some of the centroid vectors of data sets used in the experiments). For these reasons, the centroid-based classification algorithm (as well as naive Bayesian) tend to perform better than the C4.5 and the k-nearest neighbor classification algorithms.

The better performance of the centroid-based scheme over the naive Bayesian classifier is due to the method used to compute the similarity between a test document and a class. In the case of naive Bayesian, this is done using Bayes rule, assuming that when conditioned on each class, the occurrence of the different terms is independent. However, this is far from being true in real document collections [10]. The existence of positive and negative dependence between terms of a particular class causes naive

Bayesian to compute a distorted estimate of the probability that a particular document belongs to that class.

On the other hand, the similarity function used by the centroid-based scheme does account for term dependence within each class. From the discussion in Section 4, we know that the similarity of a new document x to a particular class is computed as the ratio of two quantities. The first is the average similarity of x to all the documents in the class, and the second is the square-root of the average similarity of the documents within the class. To a large extent, the first quantity is very similar, in character, to the probability estimate used by the naive Bayesian algorithm, and it suffers from similar over- and under-estimation problems in the case of term dependence.

However, the second quantity of the similarity function, (i.e., the square-root of the average similarity of the documents within the class) does account for term dependency. This average similarity depends on the degree at which terms co-occur in the different documents. In general, if the average similarity between the documents of a class is high, then the documents have a high degree of term co-occurrence (since the similarity between a pair of documents computed by the cosine function, is high when the documents have similar set of terms). On the other hand, as the average similarity between the documents decreases, the degree of term co-occurrence also decreases. Since this average internal similarity is used to amplify the similarity between a test document and the class, this amplification is minimal when there is a large degree of positive dependence among the terms in the class, and increases as the positive dependence decreases. Consequently, this amplification acts as a correction parameter to account for the over- and under-estimation of the similarity that is computed by the first quantity in the document-to-centroid similarity function. We believe that this feature of the centroid-based classification scheme is the reason that it outperforms the naive Bayesian classifier in the experiments shown in Section 3.

5 Discussion & Concluding Remarks

In this paper we focused on a simple linear-time centroid-based document classification algorithm. Our experimental evaluation has shown that the centroid-based classifier consistently and substantially outperforms other classifiers on a wide range of data sets. We have shown that the power of this classifier is due to the function that it uses to compute the similarity between a test document and the centroid vector of the class. This similarity function can account for both the term similarity between the test document and the documents in the class, as well as for the dependencies between the terms present in these documents.

There are many ways to further improve the performance of this centroid-based classification algorithm. First, in its current form it is not well suited to handle multimodal classes. However, support for multi-modality can be easily incorporated by using a clustering algorithm to partition the documents of each class into multiple subsets, each potentially corresponding to a different mode, or using similar techniques to those used by the generalized instance set classifier [9]. Second, the classification performance can be further improved by using techniques that adjust the importance of the different features in a supervised setting. A variety of such techniques have been developed in the context of k-nearest-neighbor classification [19], some of which can be extended to the centroid-based classifier [17].

References

1. L. Baker and A. McCallum. Distributional clustering of words for text classification. In *SIGIR-98*, 1998.
2. D. Boley, M. Gini, R. Gross, E.H. Han, K. Hastings, G. Karypis, V. Kumar, B. Mobasher, and J. Moore. Document categorization and query generation on the world wide web using WebACE. *AI Review (accepted for publication)*, 1999.
3. W.W. Cohen. Fast effective rule induction. In *Proc. of the Twelfth International Conference on Machine Learning*, 1995.
4. W.W. Cohen and H. Hirsh. Joins that generalize: Text classification using WHIRL. In *Proc. of the Fourth Int'l Conference on Knowledge Discovery and Data Mining*, 1998.
5. T. Curran and P. Thompson. Automatic categorization of statute documents. In *Proc. of the 8th ASIS SIG/CR Classification Research Workshop*, Tucson, Arizona, 1997.
6. E.H. Han and G. Karypis. Centroid-based document classification algorithms: Analysis & experimental results. Technical Report TR-00-017, Department of Computer Science, University of Minnesota, Minneapolis, 2000. Available on the WWW at URL *http://www.cs.umn.edu/˜karypis*.
7. W. Hersh, C. Buckley, T.J. Leone, and D. Hickam. OHSUMED: An interactive retrieval evaluation and new large test collection for research. In *SIGIR-94*, pages 192–201, 1994.
8. T. Joachims. Text categorization with support vector machines: Learning with many relevant features. In *Proc. of the European Conference on Machine Learning*, 1998.
9. Wai Lam and Chao Yang Ho. Using a generalized instance set for automatic text categorization. In *SIGIR-98*, 1998.
10. D. Lewis. Naive (bayes) at forty: The independence assumption in information retrieval. In *Tenth European Conference on Machine Learning*, 1998.
11. D. D. Lewis. Reuters-21578 text categorization test collection distribution 1.0. *http://www.research.att.com/∼lewis*, 1999.
12. A. McCallum and K. Nigam. A comparison of event models for naive bayes text classification. In *AAAI-98 Workshop on Learning for Text Categorization*, 1998.
13. Andrew Kachites McCallum. Bow: A toolkit for statistical language modeling, text retrieval, classification and clustering. http://www.cs.cmu.edu/ mccallum/bow, 1996.
14. M. F. Porter. An algorithm for suffix stripping. *Program*, 14(3):130–137, 1980.
15. J. Ross Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, San Mateo, CA, 1993.
16. G. Salton. *Automatic Text Processing: The Transformation, Analysis, and Retrieval of Information by Computer*. Addison-Wesley, 1989.
17. S. Shankar and G. Karypis. A feature weight adjustment algorithm for document classification. In *SIGKDD'00 Workshop on Text Mining*, Boston, MA, 2000.
18. TREC. Text REtrieval conference. *http://trec.nist.gov*.
19. D. Wettschereck, D.W. Aha, and T. Mohri. A review and empirical evaluation of feature-weighting methods for a class of lazy learning algorithms. *AI Review*, 11, 1997.
20. Y. Yang and X. Liu. A re-examination of text categorization methods. In *SIGIR-99*, 1999.