Empirical Evaluation of Centroid-based Models for Single-label Text Categorization

Ana Cardoso-Cachopo Arlindo L. Oliveira

Instituto Superior Técnico — DEI ; Av. Rovisco Pais, 1 ; 1049-001 Lisboa — Portugal
INESC-ID ; Rua Alves Redol, 9 ; 1000-029 Lisboa — Portugal
acardoso@gia.ist.utl.pt, aml@inesc.pt

INESC-ID Technical Report 7/2006, June 2006

Abstract

Centroid-based models have been used in Text Categorization because, despite their computational simplicity, they show a robust behavior and good performance. In this paper we experimentally evaluate several centroid-based models on single-label text categorization tasks. We also analyze document length normalization and two different term weighting schemes. We show that: (1) Document length normalization is not always the best option in a classification task. (2) The traditional *tfidf* term weighting approach remains very effective, even when compared to more recent approaches. (3) Despite the fact that several ways to calculate the centroid of a class in a dataset have been proposed, there is one that always outperforms the others. (4) A computationally simple and fast centroid-based model can give results similar to the top-performing SVM model.

1 Introduction and Previous Work

The main goal of *text categorization* (TC) is to derive models for the categorization of natural language text [19]. The objective is to derive models that, given a set of training documents with known categories and a new document, which is usually called the *query*, will predict the query's category. Here, we are interested in the case where the query belongs to a single category, a problem called *single-label text categorization*. In these models, usually based on statistical analysis, a text document is represented as an n-dimensional vector of index terms or keywords. Each index term corresponds to a word in the initial text and has a weight associated to it, which should reflect how important this index term is, for that document and/or for the collection of documents. Usually, *tfidf* term weighting is used and documents are normalized so that their length is one [19], [23].

In this paper we are interested in finding out how a relatively simple centroid-based model behaves, when compared with two well known traditional TC models and with a state-of-the-art classifier based on Support Vector Machines. We also analyze the effect that document length normalization and term weighting have on the accuracy of the obtained results.

1.1 Classification Models

In this section we briefly describe the TC models we compare in this paper.

Vector Model — In the Vector model [18, 16], documents are represented as a set of index terms which are weighted according to their importance. Using these terms, documents and queries are represented as vectors in an n-dimensional space, where n is the total number of index terms. Based on these vectors, documents can be ranked by decreasing similarity with the query, which is computed as the cosine of the angle formed by the vectors that represent each of them. The category of the query is then determined as the category of the most similar document found.

k-Nearest Neighbors — The initial application of k-NN to TC was reported by Masand and colleagues [5, 15]. The idea is to determine the category of a given query based on the categories of the k documents that are nearest to it in the document space. For this study, we first computed each document's similarity with the query, by considering the cosine between each document and the query, using the same representation as for the Vector model, as in [22, 23]. Then, we used a voting strategy to find the query's class: each retrieved document contributes a vote for its class, weighted by its similarity to the query. The query's possible classifications will be ranked according to the votes they got in the previous step, and the query will be classified in the class that had more votes.

Support Vector Machines — The Support Vector Machines (SVM) model was introduced by [21] and was first applied to TC by [13], where documents are also represented by vectors as for the Vector model. SVM is a framework for efficiently training linear classifiers, in high dimensional feature spaces. Based on the support vectors (the examples with the minimum distance to the hyperplane), SVM learn a separating hyperplane, that provides the widest margins between two different types of documents. For sets of documents that are not linearly separable, the SVM model uses *convolution functions* (or kernels), that transform the initial feature space into another one, where it finds an hyperplane that separates the data. By generating several classifiers, these ideas can easily be generalized for datasets with more than two classes of documents. SVM are acknowledged as one of the top performing methods for TC [23, 1].

Centroid-based Models — Both the Vector model and k-NN represent each document in the training set individually during the training phase and have to consider each one of them every time a new document needs to be classified. On the other hand, the SVM model finds a "description" for each class of documents that distinguishes it from the others.

Like SVM, centroid-based models find a description for each class that is shorter than the various documents that compose it, but still reasonably accurate. In particular, centroid-based models find a representation for a "prototype" document that summarizes all the known documents for a given class, which is called the centroid of the class. There are several ways to calculate this centroid during the training phase, and several proposals have appeared in the literature. In centroid-based models, the centroid of a particular class C_j is represented by a vector $\overrightarrow{c_j}$, which is a combination of the document vectors $\overrightarrow{d_i}$ belonging (or not) to C_j.

Centroid computation using the Rocchio formula — In the Centroid-Rocchio model, the centroid of a class is the sum of all the document vectors for the positive training examples for this class, minus the sum of all the vectors for the negative training examples, weighted by control parameters β and γ:

$$\overrightarrow{c_j} = \beta \cdot \textstyle\sum_{\overrightarrow{d_i} \in C_j} \overrightarrow{d_i} - \gamma \cdot \sum_{\overrightarrow{d_i} \notin C_j} \overrightarrow{d_i}$$

The application of this model to TC was first proposed by Hull [9] and it has been used in other works where the role of negative examples is deemphasized, by setting β to a higher value than γ (usually $\beta = 16$ and $\gamma = 4$) [4, 11, 12].

Centroid as average — In the Centroid-Average model, each class C_j, which has $|C_j|$ documents, is represented by the average of all the vectors for the positive training examples for this class [8, 20]:

$$\overrightarrow{c_j} = \tfrac{1}{|C_j|} \cdot \textstyle\sum_{\overrightarrow{d_i} \in C_j} \overrightarrow{d_i}$$

Centroid as sum — In the Centroid-Sum model, each class is represented by a vector which is the sum of all the vectors for the positive training examples for this class [3]:

$$\overrightarrow{c_j} = \textstyle\sum_{\overrightarrow{d_i} \in C_j} \overrightarrow{d_i}$$

Centroid as normalized sum — In the Centroid-NormalizedSum model, each class is represented by a vector which is the sum of all the vectors for the positive training examples for this class, normalized so that it has unitary length [14]:

$$\overrightarrow{c_j} = \tfrac{1}{\|\overrightarrow{c_j}\|} \cdot \textstyle\sum_{\overrightarrow{d_i} \in C_j} \overrightarrow{d_i}$$

During the classification phase, each test document (or query) is represented by its vector, $\overrightarrow{d_i}$, and it is compared with each of the centroids, $\overrightarrow{c_j}$. The document will be classified as belonging to the class to whose centroid it has the greatest cosine similarity:

$$sim(\overrightarrow{d_i}, \overrightarrow{c_j}) = \frac{\overrightarrow{d_i} \cdot \overrightarrow{c_j}}{||\overrightarrow{d_i}|| \times ||\overrightarrow{c_j}||}$$

With centroid-based models, there is a big reduction in time and memory required during the classification phase, because now they are proportional to the number of classes instead of the number of training documents. They also have the advantage that it is easy to add more training documents for a particular class and easily recalculate its centroid (on-line methods, see [19, page 24]).

1.2 Document Representation and Length Normalization

Traditionally, documents are represented as a set of index terms which are weighted according to their importance for a particular document and for the general collection [18, 16]. The index terms usually correspond to the words or tokens in the document (or query) and index term weights can be computed in several ways. The most usual is *tfidf* (term frequency/inverse document frequency) [17], which increases with the number of times that the term occurs in the document and decreases with the number of times the term occurs in the collection.

A recent approach [14] proposes a more sophisticated weighting method, based on term frequencies within a particular class and within the collection of training documents. We will call this approach term distributions (*td*).

It has been generally assumed that normalizing document vectors so that they have unit length is beneficial for TC applications.

2 Experimental Setup

In this section we present the experimental setup that was used for this paper, namely the datasets that were used, the implementation and parameters for each TC model and the evaluation measure that was used.

2.1 Datasets

To allow the comparison of our work with previously published results, we used two standard TC benchmarks in our evaluation, downloaded from a publicly available website [6]. In this website there is also a description of the datasets and of the pre-processing techniques that were applied to each dataset, namely character clean-up, removal of short words, removal of stopwords, and stemming.

20 Newsgroups — This dataset is a collection of approximately 20,000 newsgroup documents, partitioned (nearly) evenly across 20 different newsgroups. For this dataset, we used the files `20ng-train-stemmed` and `20ng-test-stemmed`, available from the website.

Reuters 21578 — The documents in Reuters-21578 appeared on the Reuters newswire in 1987 and were manually classified by personnel from Reuters Ltd. Due to the fact that the class distribution for these documents is very skewed, two sub-collections are usually considered for text categorization tasks [7]. Because we are concerned with single-label TC, we used **r8** and **r52**, which correspond to the documents with a single topic and the classes which still have at least one training and one test example after removing documents with more than one topic. For this dataset, we used the files `r8-train-stemmed`, `r8-test-stemmed`, `r52-train-stemmed` and `r52-test-stemmed`, available from the website.

The name of each dataset indicates how many classes it has. Moreover, in order for our work to be comparable to other published works in this area, we use the standard train/test split for each dataset, as described in [6]. In order to analyze the effect that the size of the training set has on the classification task, we split each *training* dataset into 10 “slices”, each one containing 10% of the training set, with the restriction that the initial “slice” contains at least one document from each class.

2.2 Algorithm Implementation and Parameters

For the Vector model we used a Sourceforge project called IGLU [10]. IGLU aims at being a software platform suitable for testing Information Retrieval models. At the time of this writing, only the Vector model is implemented.

For k-NN we implemented a “voting strategy”, where the possible classes of a document are voted on by the documents that belong to that class. We used the cosine similarity, returned by the Vector model, as the weight for each vote, and considered only the 10 nearest documents.

For SVM we used LIBSVM [2]. LIBSVM is an integrated software for Support Vector classification that supports multi-class classification. In our experiments we used a linear kernel.

For the centroid-based models, we calculated each centroid, as described in Section 1.1.

For the *td* term weighting approach, we implemented it, according to the description in [14], and used exponent factors $\alpha = 0.5$, $\beta = -1$, and $\gamma = -0.5$.

2.3 Evaluation Criteria

Evaluating the performance of computational systems is often done in terms of the resources (time and space) they need to operate, assuming that they perform the task that they are supposed to. In TC it is not enough to return a ranked list of categories in a reasonable amount of time, because the categories should also be the “right” ones. Measures based on Precision and Recall, like F1 or PRBP have been widely used to compare the performance of TC models [19]. However, to evaluate single-label TC tasks, these measures are not adequate. So, Accuracy, the percentage of correctly classified documents, is used to evaluate this kind of tasks.

$$Accuracy = \frac{\#\text{Correctly classified queries}}{\#\text{Total queries}}$$

3 Results

In this section we present the results obtained using the centroid-based models, and compare them with the results obtained with other classification models. We also discuss the effect that term weighting, document length normalization and classification model have on the accuracy of TC.

Except where stated otherwise, we use the standard train/test split for each dataset and not n-fold cross validation, in order for our work to be comparable to other published works in this area.

3.1 Effect of Term Weighting and Length Normalization

Figures 1 through 5 vary term weighting (*tfidf* or *td*) and document length normalization using the Vector, k-NN, and each of the centroid-based models.[1] For these models, we only considered normalizing or not the training documents, because, once similarity is determined as a cosine, the length of the query is irrelevant for it. We do not show the results for the Centroid-Sum model, because they are always worse than the ones for Centroid-NormalizedSum. We will not discuss the Centroid-Sum model any further.

In Figures 1, 3 and 4 we can see that using *tfidf* term weighting and normalizing the length of the training documents always yields the best results with the Vector, Centroid-Rocchio and Centroid-Average models, for any proportion of the training set that is used.

In Figure 2 we can see that, for the k-NN model, normalizing training documents' length is better for r52 and 20ng, but it has little effect for r8, being slightly better not to normalize the training documents. In any case, it is better to use *tfidf* than *td* weighting.

Figure 5 shows that document length normalization has a smaller influence on the Centroid-NormalizedSum model, which is probably due to the fact that each centroid is normalized in the end. For r8 and 20ng, it is even better not to normalize the training documents. Regarding term weighting, *td* is better for r8 and 20ng and almost as good as *tfidf* for r52. This is to be expected, because *td* term weighting was proposed as an improvement for the Centroid-NormalizedSum model (see [14] for more details).

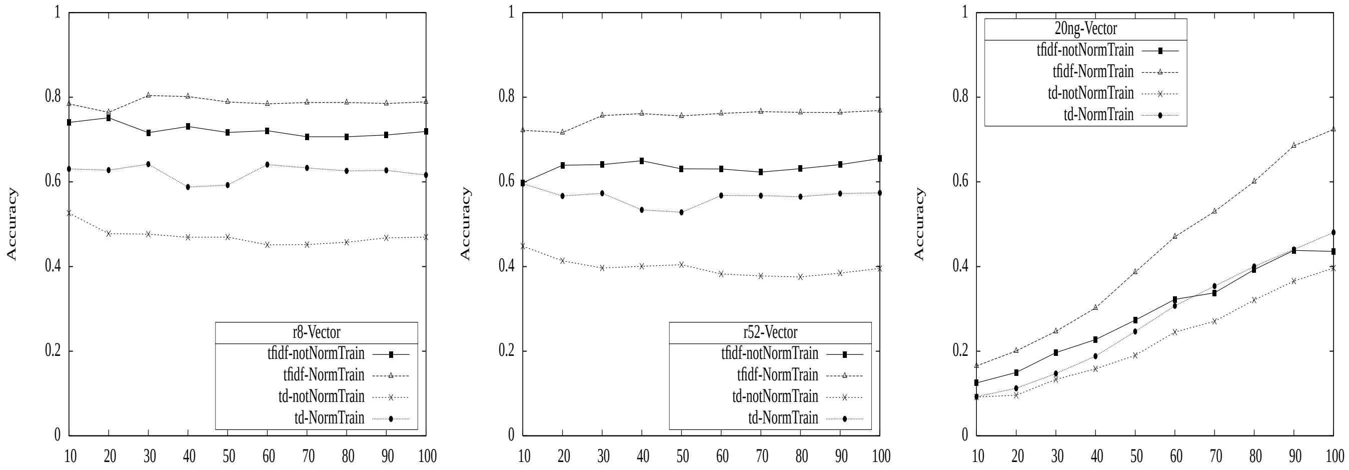

Figure 1: Accuracy for the three datasets varying document representation and normalization using the Vector model, as a function of the percentage of the training set that is used.

For the SVM model, it is not enough to consider training document normalization, because query length also influences the results. For this model, Figures 6 and 7 show the effect of normalizing training documents and queries, using *tfidf* or *td* term weighting, respectively. We can see that, for this model, it is always better to normalize both training and query documents. Surprisingly, for 20ng, not normalizing either of them shows even better performance, when smaller portions of the training set are used. By comparing both figures, we can see that, for all datasets, using *tfidf* term weighting provides the best results with the SVM model.

By looking at Figures 1 through 7, we can also observe that, when using *td* term weighting, performance sometimes decreases when the used fraction of the training data increases, for datasets r8 and r52. This is probably due to the fact that these collections are very skewed and so term distributions, which depend on the classes, are bad predictors. This doesn't happen with 20ng because this collection is more balanced than the other two.

3.2 Effect of the Classification Model

Figure 8 allows us to compare performance across different models for these datasets, as a function of the percentage of the training set that is used. Each line corresponds to the best normalization/weighting combination for the respective model.

If we compare the lines for Centroid-NormalizedSum to the lines of Vector or k-NN, we can see that Centroid-NormalizedSum is a lot better than the other two, in every dataset. If we compare the different centroid-based models, we can see

[1] All the figures in this section and the next show Accuracy for each of the three datasets as a function of the percentage of the training set that is used for training.

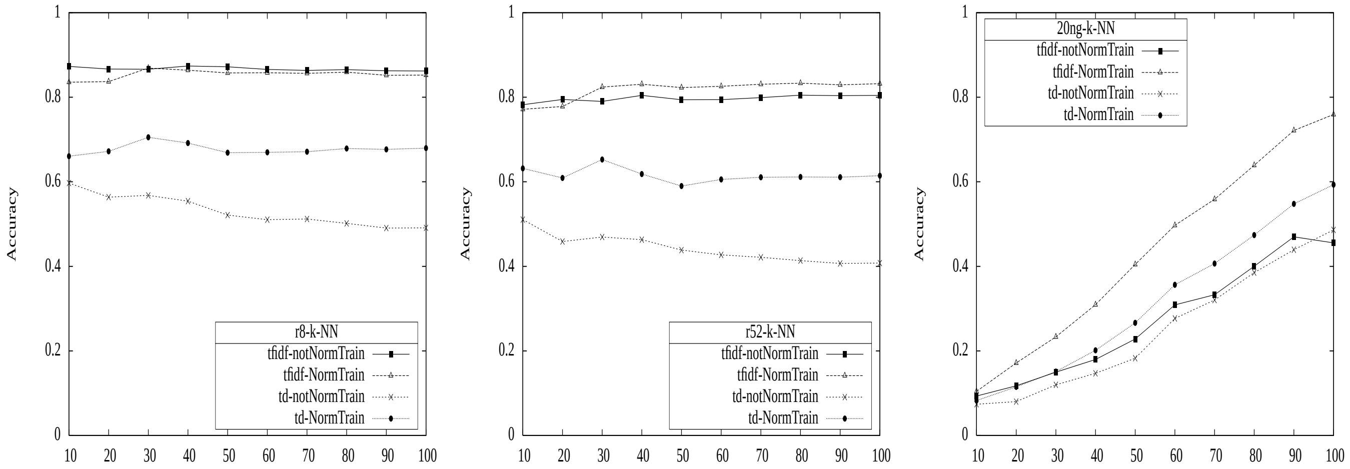

Figure 2: Accuracy for the three datasets varying document representation and normalization using the k-NN model, as a function of the percentage of the training set that is used.

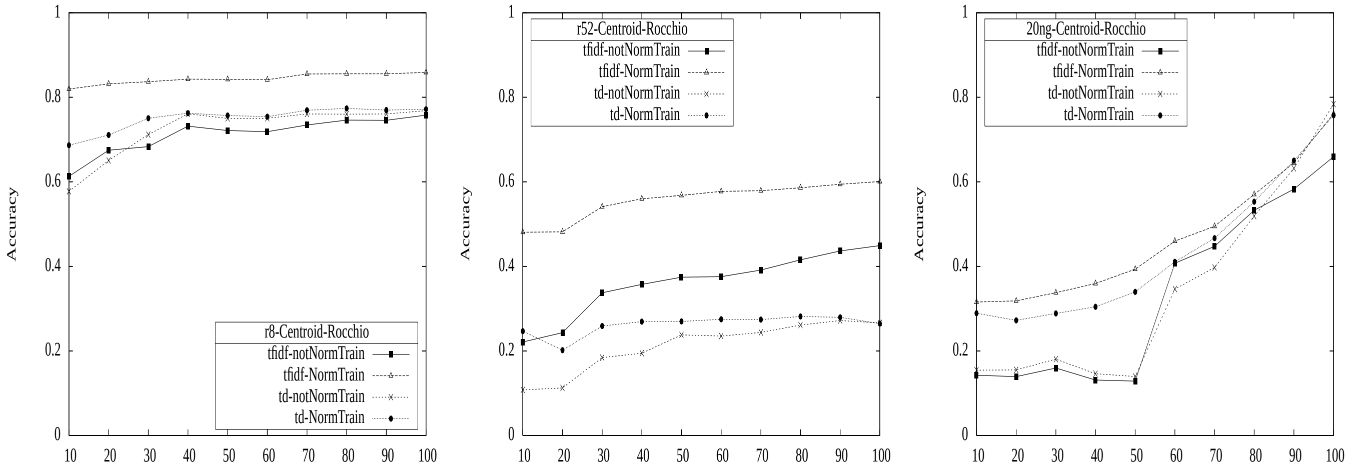

Figure 3: Accuracy for the three datasets varying document representation and normalization using the Centroid-Rocchio model, as a function of the percentage of the training set that is used.

that Centroid-NormalizedSum always performs better than the others. The only model that is consistently better than Centroid-NormalizedSum is SVM, which already is known as a top-performing model. This leads us to conclude that, overall, Centroid-NormalizedSum is a very interesting model, when one takes into consideration the efficiency vs accuracy trade-off.

If we look at the accuracy reports for r8 and r52 across the different models, normalization, and weighting schemes, we can confirm what was already stated in [7], that r52 is a harder collection than r8, because every TC model behaves worse for r52. If we further compare with the accuracy achieved for 20ng, we can see that 20ng is even harder than the other two.

If we look at term weighting and normalization across the different figures, we can see that *td* only improves performance for the Centroid-NormalizedSum model. This can be due to the exponent factors that were used. We plan to do further testing with other factors. For this model only, it is also better not to normalize document length.

3.3 Comparison of Execution Times for the Models

Besides the quality of the results that a TC model yields, another important aspect to consider when evaluating TC models is the time and memory they require to execute.

Figure 9 shows a comparison of the time that each model[2] takes to execute, in both the training and the classification

[2] Because execution times for every centroid-based model are very similar, we only show times for the Centroid-NormalizedSum model.

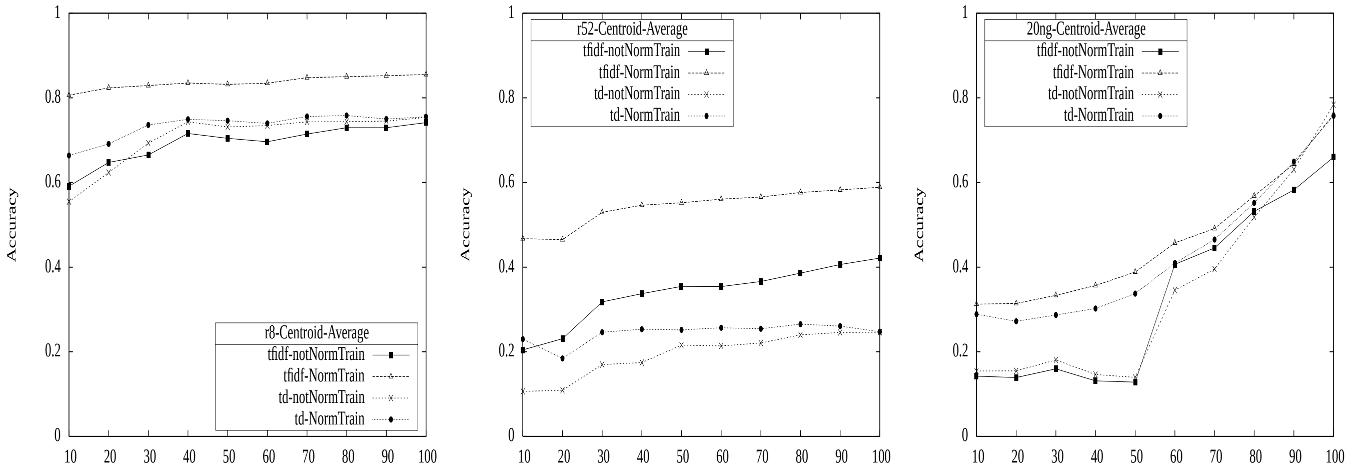

Figure 4: Accuracy for the three datasets varying document representation and normalization using the Centroid-Average model, as a function of the percentage of the training set that is used.

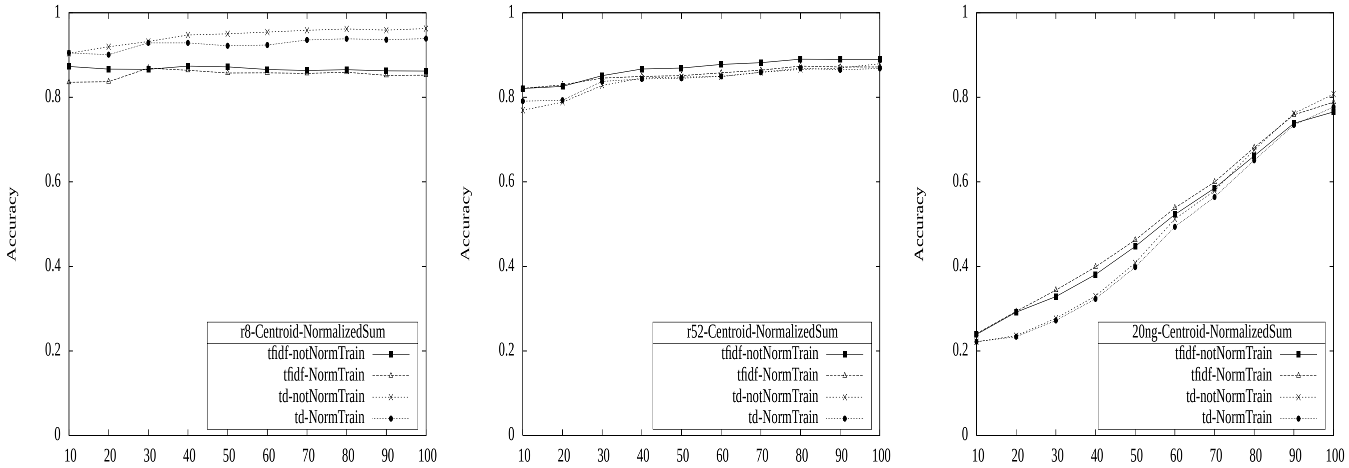

Figure 5: Accuracy for the three datasets varying document representation and normalization using the Centroid-NormalizedSum model, as a function of the percentage of the training set that is used.

phases, for each dataset, using *tfidf*. The X axis represents the time spent during the training phase, while the Y axis represents the time spent during the classification phase, both in seconds, on a logarithmic scale. These times are the average obtained over five runs for each model and each dataset.

By looking at the X axis of Figure 9, we can compare the time that each model takes for training. As expected, Vector and k-NN are the fastest (and spend almost the same time), because they simply read the training documents and save the term/document matrix for future use. The other models need to build this matrix and then determine a new model of the data. The simplest transformation is done by the Centroid-NormalizedSum model, because it only calculates the centroid for each class.

To compare the time that each model takes for classification, we can look at the Y axis of Figure 9. In this case, we can see that the fastest model for every dataset is the Centroid-NormalizedSum model. This should be expected, because the operation that is required for each test document is a cosine similarity with each class' centroid. The second fastest is SVM. The slowest are Vector and k-NN, because they have to compare each test document to each training document, making test time proportional to the number of documents in the training set. Using an inverted index structure would speed up these methods, but they would still be considerably slower than the centroid-based methods.

Overall, the centroid-based model provides a significant reduction in time and memory required during the classification phase, because these requirements are proportional to the number of classes instead of the number of documents.

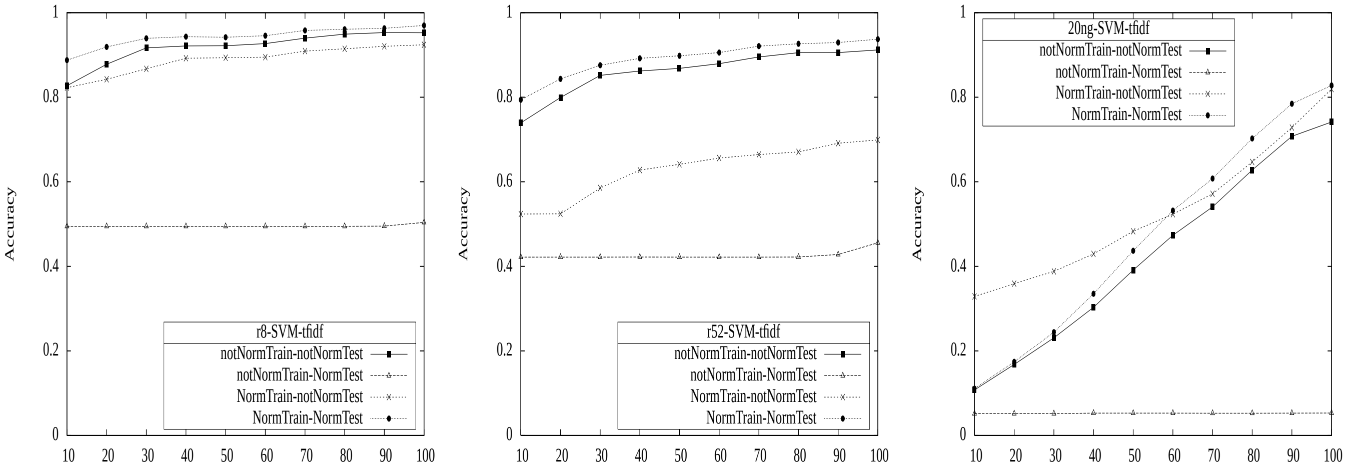

Figure 6: Accuracy for the three datasets varying normalization of the train and test documents using the SVM model and *tfidf* term weighting, as a function of the percentage of the training set that is used.

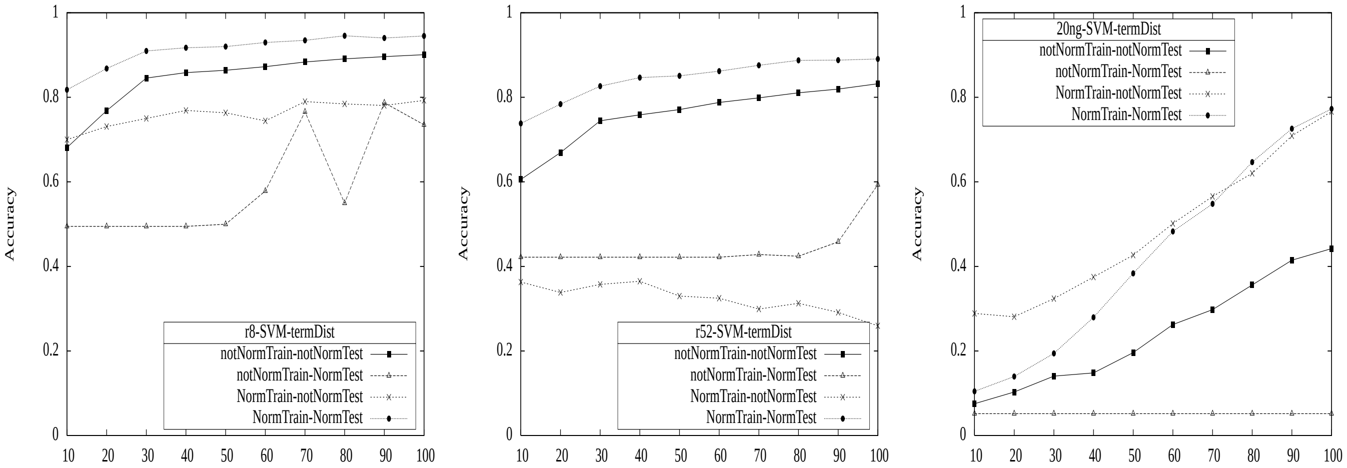

Figure 7: Accuracy for the three datasets varying normalization of the train and test documents using the SVM model and *td* term weighting, as a function of the percentage of the training set that is used.

4 Conclusions and Future Work

We have shown that document length normalization is not always the best option in a classification task and that the traditional *tfidf* term weighting approach remains very effective, even when compared to *td*, a more recent and computer-intensive term weighting approach.

Regarding centroid-based models, we have shown that Centroid-NormalizedSum always outperforms other centroid-based approaches. Compared to the Vector and k-NN models, Centroid-NormalizedSum significantly improves accuracy for three standard datasets.

Finally, we have shown that Centroid-NormalizedSum, a computationally simple and fast model can obtain accuracy results similar to the more sophisticated and computational expensive state-of-the-art SVM model.

Overall, the Centroid-NormalizedSum model presents a good trade-off between time spent during the training and classification phases and the quality of the results obtained. This model also has the advantage of being amenable to changes in the training set, unlike the SVM model.

In the future, we will consider more sophisticated models for determining the class centroid. We also plan to study how this model behaves when we consider more than one vector to characterize each class. Another interesting aspect to study is how this model behaves when new documents are incrementally added to the training set.

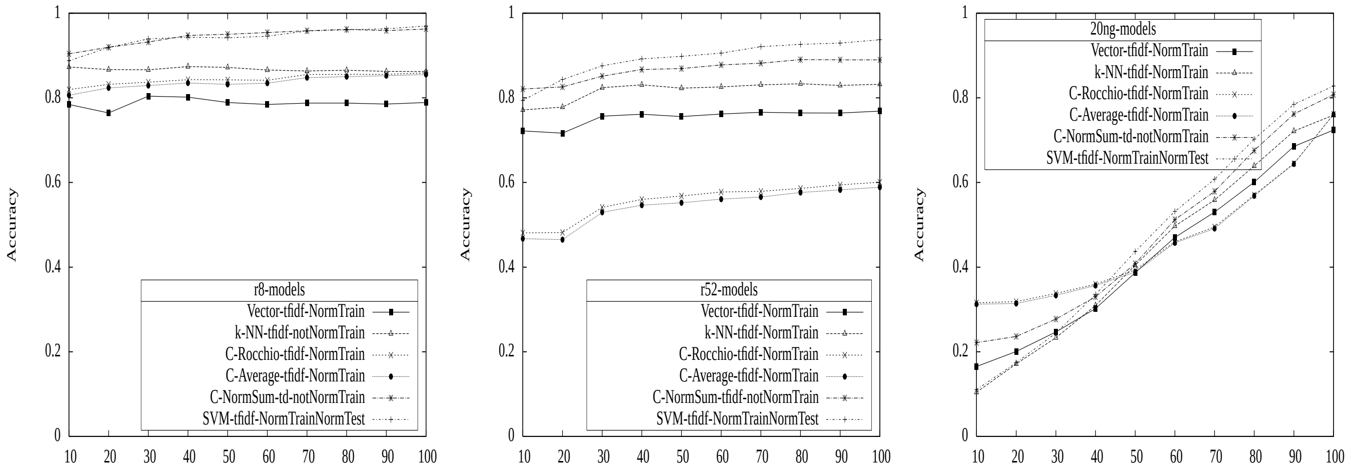

Figure 8: Accuracy for the three datasets varying the TC model, as a function of the percentage of the training set that is used.

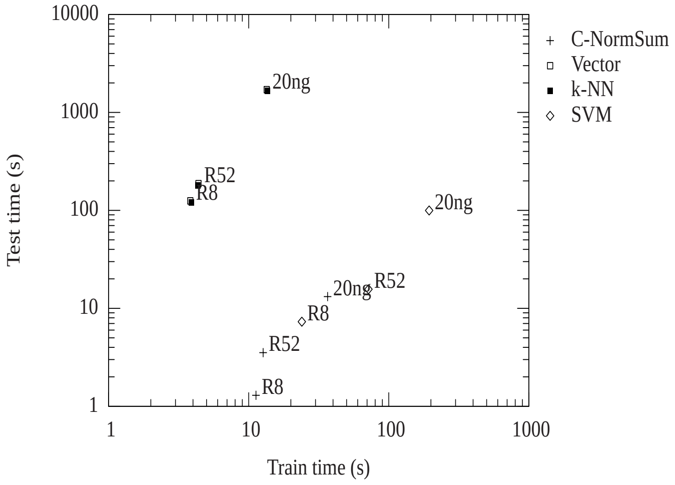

Figure 9: Time spent by each TC model for each dataset.

References

[1] A. Cardoso-Cachopo and A. L. Oliveira. An empirical comparison of text categorization methods. In *Proceedings of SPIRE-03, 10th International Symposium on String Processing and Information Retrieval*, pages 183–196. Springer Verlag, 2003.

[2] C.-C. Chang and C.-J. Lin. *LIBSVM: a library for support vector machines*, 2001. Software available at `http://www.csie.ntu.edu.tw/˜cjlin/libsvm`.

[3] W. T. Chuang, A. Tiyyagura, J. Yang, and G. Giuffrida. A fast algorithm for hierarchical text classification. In *Proceedings of DaWaK-00, 2nd International Conference on Data Warehousing and Knowledge Discovery*, pages 409–418. Springer Verlag, 2000.

[4] W. W. Cohen and Y. Singer. Context-sensitive learning methods for text categorization. *ACM Transactions on Information Systems*, 17(2):141–173, 1999.

[5] R. M. Creecy, B. M. Masand, S. J. Smith, and D. L. Waltz. Trading mips and memory for knowledge engineering: classifying census returns on the connection machine. *Communications of the ACM*, 35(8):48–63, 1992.

[6] Datasets for single-label text categorization, 2005. Publicly available at `http://www.gia.ist.utl.pt/˜acardoso/datasets/`.

[7] F. Debole and F. Sebastiani. An analysis of the relative hardness of reuters-21578 subsets. *Journal of the American Society for Information Science and Technology*, 56(6):584–596, 2004.

[8] E.-H. Han and G. Karypis. Proceedings of the 4th european conference on centroid-based document classification: Analysis and experimental results. In *Principles of Data Mining and Knowledge Discovery*, pages 424–431, 2000.

[9] D. A. Hull. Improving text retrieval for the routing problem using latent semantic indexing. In *Proceedings of SIGIR-94, 17th ACM International Conference on Research and Development in Information Retrieval*, pages 282–289. Springer Verlag, 1994.

[10] IGLU Java—Java Library for Information Retrieval Research, 2002.

[11] D. J. Ittner, D. D. Lewis, and D. D. Ahn. Text categorization of low quality images. In *Proceedings of SDAIR-95, 4th Annual Symposium on Document Analysis and Information Retrieval*, pages 301–315, 1995.

[12] T. Joachims. A probabilistic analysis of the Rocchio algorithm with TFIDF for text categorization. In *Proceedings of ICML-97, 14th International Conference on Machine Learning*, pages 143–151. Morgan Kaufmann Publishers, 1997.

[13] T. Joachims. Text categorization with support vector machines: learning with many relevant features. In *Proceedings of ECML-98, 10th European Conference on Machine Learning*, pages 137–142. Springer Verlag, 1998.

[14] V. Lertnattee and T. Theeramunkong. Effect of term distributions on centroid-based text categorization. *Information Sciences*, 158(1):89–115, 2004.

[15] B. Masand, G. Linoff, and D. Waltz. Classifying news stories using memory-based reasoning. In *Proceedings of SIGIR-92, 15th ACM International Conference on Research and Development in Information Retrieval*, pages 59–65. ACM Press, 1992.

[16] G. Salton. *The SMART Retrieval System*. Prentice Hall, 1971.

[17] G. Salton and C. Buckley. Term-weighting approaches in automatic text retrieval. *Information Processing and Management*, 24(5):513–523, 1988.

[18] G. Salton and M. Lesk. Computer evaluation of indexing and text processing. *Journal of the ACM*, 15(1):8–36, 1968.

[19] F. Sebastiani. Machine learning in automated text categorization. *ACM Computing Surveys*, 34(1):1–47, 2002.

[20] S. Shankar and G. Karypis. Weight adjustment schemes for a centroid based classifier, 2000. Computer Science Technical Report TR00-035, Department of Computer Science, University of Minnesota, Minneapolis, Minnesota.

[21] V. Vapnik. *The Nature of Statistical Learning Theory*. Springer Verlag, 1995.

[22] Y. Yang. Expert network: effective and efficient learning from human decisions in text categorisation and retrieval. In *Proceedings of SIGIR-94, 17th ACM International Conference on Research and Development in Information Retrieval*, pages 13–22. Springer Verlag, 1994.

[23] Y. Yang and X. Liu. A re-examination of text categorization methods. In *Proceedings of SIGIR-99, 22nd ACM International Conference on Research and Development in Information Retrieval*, pages 42–49. ACM Press, 1999.