Simple and Accurate Feature Selection for Hierarchical Categorisation

Wahyu Wibowo Hugh E. Williams
School of Computer Science and Information Technology
RMIT University, GPO Box 2476V
Melbourne, Australia, 3001.
{wwibowo,hugh}@cs.rmit.edu.au

ABSTRACT

Categorisation of digital documents is useful for organisation and retrieval. While document categories can be a set of unstructured category labels, some document categories are hierarchically structured. This paper investigates automatic hierarchical categorisation and, specifically, the role of features in the development of more effective categorisers. We show that a good hierarchical machine learning-based categoriser can be developed using small numbers of features from pre-categorised training documents. Overall, we show that by using a few terms, categorisation accuracy can be improved substantially: unstructured leaf level categorisation can be improved by up to 8.6%, while top-down hierarchical categorisation accuracy can be improved by up to 12%. In addition, unlike other feature selection models — which typically require different feature selection parameters for categories at different hierarchical levels — our technique works equally well for all categories in a hierarchical structure. We conclude that, in general, more accurate hierarchical categorisation is possible by using our simple feature selection technique.

Categories and Subject Descriptors

H.3.4 [**Information Storage and Retrieval**]: Information Search and Retrieval Search Process; H.3.4 [**Information Storage and Retrieval**]: Information Search and Retrieval Information Filtering

General Terms

Algorithms, Design, Experimentation

Keywords

Categorisation, Hierarchical Categorisation, Web Hierarchies, Error Reduction

DocEng'02, November 8–9, 2002, McLean, Virginia, USA.

1. INTRODUCTION

Most popular categorisation techniques assign labels to documents from an unstructured list of categories. However, when large numbers of categories are involved, there are often relationships between the categories: for example, in web directories such as DMOZ and Yahoo![1] the categories are arranged hierarchically. Hierarchical structures simplify the information finding and categorisation tasks by allowing users to incrementally choose narrower, more specific categories.

In this paper we investigate the role of document features in the development of effective hierarchical categorisers; features are document fragments such as paragraphs, sentences, words, or word fragments. Features are selected from documents — usually during the categoriser training process — with the dual aims of finding a representative set that best describe the characteristics of each category and removing features that are not useful in the categorisation process. Unlike commonly used feature selection techniques — which usually rely on the properties of the document collection — we use a simple technique.

Our intuitive feature selection technique is to use a fixed number of terms extracted from the beginning of each document. This approach has been applied in non-hierarchical categorisation [19] but, to our knowledge, has not been investigated in hierarchical categorisation. Our approach is intuitive as it resembles the behaviour of a visitor searching for information at a library. During a visit to a library, only a small amount of information — which is typically similar to that found on the cover or in the preamble of the book — is needed to find the general location of the book of interest. This guides a visitor to the aisle that contains the book. More detailed information is then required to find the book within the aisle, and this information may be at the level of detail typically found on the first few pages. Examining the table of contents of the target book to find a chapter would require still more information.

We compare the performance of our categorisers to those developed without feature selection. We show — in agreement with other feature selection work — that categorisers developed using fewer terms exhibit better performance than those that are developed using all terms. This has two advantages: first, reduction in features results in faster categorisation and reduced main-memory requirements; and, second, the accuracy is improved.

[1]See `http://www.dmoz.org/` and `http://www.yahoo.com/`

We tested our approach to feature selection with three hierarchical test collections and four categorisation techniques. We found, as expected, that categories closer to the root of the hierarchy usually require less features for training than categories at or closer to the leaves. For higher level categories, we show that accuracy can be improved by around 9% by using only a few terms from each document. By using more terms — but still only a fraction of each document — an around 8.5% improvement is possible compared to categorisation with all features at the leaf level categories. We also show that this approach can be generalised for top-down hierarchical categorisation, where using fewer terms can improve the performance by around 6.5%. Moreover, we show that our approach produces better and more consistent performance than the use of a static stoplist.

2. CATEGORISATION

Automatic categorisation is an important tool for managing online documents. A search engine that attaches categories to the documents in its collection might face up to a million new or modified documents each day. A lesser, but still substantial, number of new incoming articles arrive at news bureaus. In both cases, the categorisation of incoming documents is desirable and an automatic categoriser is the only practical solution for categorising all documents.

In this paper, we use machine learning-based linear similarity categorisers to address the automatic categorisation problem and a word model to represent documents. This approach determines whether a document should be assigned to a category based on the computation of a linear function [11]. This approach is effective, and can be used efficiently on large scale datasets on general-purpose hardware [19].

Rocchio Categoriser

Rocchio categorisers [11, 14] assume that a category representation must combine the properties of the positive and negative example documents. Using a vector model, a category representative vector w is modified by adding the weight of the linear similarity of the positive training terms and subtracting the weight of the linear similarity of the negative training terms.

Rocchio categorisers are trained using the following approach. The jth term of the category representative feature vector w for a category C is updated using each training document feature vector v_i as follows:

$$w'_j = \alpha w_j + \frac{\beta}{|C|}\sum_{i\in\mathbf{C}} v_{ij} - \frac{\gamma}{n-|C|}\sum_{i\notin\mathbf{C}} v_{ij}$$

where n is the total number of training documents, C is the set of positive on-topic training documents, i is a training document, and α, β, and γ are constants.

Using the Rocchio measure, a category feature vector can be derived from a set of categories and training documents. For every training document, the weight of the vector of each category is modified so that when a document is a category member the weight of category terms are increased and when a document is not a category member the weight of non-category terms is decreased. The result is a vector for each category w of length t, where t is the count of distinct terms in the collection; in this way, all category vectors are of length t and contain all terms.

TFIDF Categoriser

Joachims [7] has proposed a direct application of the well-known TF.IDF scheme [16, 24] to categorisation. If d is a new document to categorise, C is a set of categories, and F is a set of terms, then the TFIDF score of document d for category c can be computed as follows:

$$\mathrm{TFIDF}(d,c) = \frac{\sum_{w\in F}(\mathrm{TF}(w,d).\mathrm{IDF}(w)).(\mathrm{TF}(w,c).\mathrm{IDF}(w))}{\sqrt{(\sum_{w\in F, c\in C}(\mathrm{TF}(w,c).\mathrm{IDF}(w))^2)}}$$

where $\mathrm{TF}(w,d)$ is the term frequency of term w in document d, while $\mathrm{TF}(w,c)$ is the term frequency of term w in all documents of category c.

The category of a document can be assigned by computing the TFIDF scores for all categories. The category with the highest score is then selected as the document category. A category can also be assigned if the score is above some threshold point for that category, permitting multiple category assignment.

Perhaps because of its simplicity, the TFIDF categoriser suffers from inaccuracy caused by highly frequent terms. These terms may dominate the score computation and cause miscategorisation. To address this problem, Joachims [7] removes the 100 most frequent words from the usenet newsgroup dataset in his experiment. We have similarly found that removing frequently occuring terms is necessary, and we use a similar approach in our work.

Probabilistic Categoriser

In a probabilistic approach, the category of a document is computed as the probability that the document belongs to a category. Bayes rules can be used to find the probability that a document belongs to a category by assuming that the presence of a document is independent of the other documents in a collection. By assuming that term distributions in relevant documents and in non-relevant documents are independent, and probable relevance is based on the presence and absence of the terms in the documents, Robertson and Sparck-Jones [13] proposed the following weighting scheme to compute the probability:

$$w(t,C_i) = \log\left(\frac{r+0.5}{R-r+0.5} \div \frac{n-r+0.5}{(N-n)-(R-r)+0.5}\right)$$

where N is the number of documents in the collection, n is the number of documents that contains term t, R is the number of training documents for category C_i, and r is the number of training documents for category C_i that contain term t.

The category assignment of document d is then computed using the following formula:

$$\mathrm{cat}(d) = \max_{c\in C} \frac{\sum_{f\in F}\alpha.w(f,c)}{\sqrt{(\sum_{f\in F}(\alpha.w(f,c))^2)}}$$

where F is a set of terms, C is a set of categories, and α is a constant. We use term frequency for α.

Support Vector Machine (SVM)

A categoriser often has to deal with an high dimensional input space that contains thousands of terms. A Support Vector Machine [8, 9] (SVM) is a linear categoriser for an high dimensional input space. For a new example vector x, an SVM will compute $f(x,y)$ to determine if x belongs to category y, where f is a linear function.

To solve this problem, an SVM non-linearly transforms the training data into a dot product or *feature space*. The transformation is performed using a kernel function that computes a hyperplane function that has a maximum margin, and will optimally separate positive and negative samples. In our experiments, we use a linear hyperplane of the form $(w.x) + b = 0$ where w is a weight or *support vector*, and b is a threshold. The support vector is in the form of $\sum_i^l v_i x_i$ that constitutes a subset of the training pattern. During categorisation, the hyperplane will separate positive samples that have the property $(w.x) + b = 1$ and negative samples that have the property $(w.x) + b = -1$.

Feature Selection

Some terms are not useful in categorisation: either they are uninformative or they do not influence overall performance. Removing these terms can speed up the categorisation process and accuracy may also be improved. Feature selection filters uninformative terms from a document [25, 26]. Several feature selection techniques have been proposed to measure how informative a term is in a collection, including DF-thresholding [7, 15, 26], the χ^2-Test [17, 26], the Mutual Information model [5, 26], the Information Gain formula [8, 26], and the Term-strength Criterion [26]. An excellent survey of these techniques can be found elsewhere [18].

A DF-thresholding strategy will remove terms that appear in either very few or almost all documents. The χ^2-Test is used to measure the role of a term in differentiating between two categories. Mutual Information measures the association between a term and a category, while Information Gain or the Term-strength Criterion measure the importance or strength of a term in a collection. Yang [26] reports that DF-Thresholding, Information Gain, and χ^2-Test can remove up to 90% of terms without reducing accuracy. Baker and McCallum [2] showed a similar result using a different technique: they clustered the features based on the distribution of category labels and then removed features.

While most work on feature selection does not address the specific application of the techniques to hierarchical categorisation, Koller and Sahami [10] integrated feature selection into their hierarchical categorisation approach. They develop several small categorisers based on the hierarchy structure, and the feature selection technique is specific to each categoriser. Using a Bayesian categoriser and cross-entropy feature selection they developed categorisers with good performance. Koller and Sahami's work has been repeated by Mladenic and Grobelnik [12] by applying the technique to different collections and different feature selection methods.

3. FEATURE SELECTION FOR HIERARCHICAL CATEGORISATION

Feature selection techniques may reduce the accuracy of categorisation if important features are excluded. For example, when a feature is important in one document, but is noise in most others, it is likely to be excluded from the categorisation process. Moreover, features are excluded when a categoriser is developed because they meet the criteria to be removed using the available training documents. However, when new training documents appear, it is possible that those excluded features may become important. Feature selection can therefore invalidate previous categorisation results, and this is why it should be used with caution.

In this work, we experiment with alternative techniques for feature selection that are not dependent on collection properties. Our techniques have the advantage that the features that are used from each document are chosen solely on the content of that document, that is, collection statistics are not used and do not need to be maintained. Our approach — which Shanks and Williams [19] refer to as *first m words* — is to extract as features the first fragment of each training document; in their approach, m is a constant. The rationale is that, in general, a summary of each document is present at its beginning and this is supported by their experimental results; in contrast, they show that the last words, middle words, and random words are not good representative summaries.

To our knowledge, the first m words approach has not previously been investigated for hierarchical categorisation. However, it is an intuitive approach that resembles the conventional information discovery task in hierarchical environments. For example, when using a web hierarchy to locate information on a specific topic, a user begins by selecting a broad category; in making this choice, very little information concerning the topic is needed and these features are likely to be present at the beginning of a document. After navigating through the hierarchy by progressively choosing narrower categories, more information is needed to make a decision and this requires further features from a document. We therefore expect that a first m words approach may offer both accurate and fast hierarchical categorisation, but that more features will be required as categorisation proceeds from the root to the leaves of a hierarchy.

We experiment with the four categorisers we have described, different numbers of features extracted using the first m words approach, different collections, and an existing feature selection technique for comparison. We apply categorisation to each group of categories at the same hierarchical level, and experiment with a top-down hierarchical categorisation that we compare to non-hierarchical categorisation.

4. COLLECTIONS

In this section, we describe the three collections we use in our experiments.

Reuters Topics Super Category (RTSC)

The *Reuters-21578* collection contains Reuters newswire documents from 1987[2]. The documents have been manually categorised into 6 global and 374 detailed categories.

There are several variants of the Reuters-21578 collection that are used in categorisation research and depend upon the interpretation of the SGML tags. Each variant divides the collection into training and test documents. The collection has been divided into different training and test sets by many authors, including the *mod-Apte split* [1] that removes all unlabelled documents. Hayes [6] recategorises documents in this variant: for all documents under the TOPICS global category — which consists of 135 of the detail categories — he developed 8 additional global categories. This creates the *Reuters Topics Super Category Collection* (RTSC) col-

[2]The collection is SGML tagged and is available from `http://www.daviddlewis.com/resources/testcollections/reuters21578/`

lection. In our experiments, we treat the RTSC global categories as parents in a hierarchy and the detail categories as their children.

The RTSC collection is divided into 7,775 training documents and 3,019 test documents. There is an average of 108 terms per training document.

20-Newsgroup Collection

The 20-Newsgroup collection contains discussion documents randomly selected from 20 newsgroups from Usenet news. The original version of this collection contained 20,000 documents. We use the second version where duplicates and headers are removed[3] leaving 18,836 documents. The 20-Newsgroup collection is divided into 12,566 training documents and 6,270 test documents, with an average of 158 terms per training document.

The names of the newsgroups represent the contents of the forum and are organised hierarchically. For example, the newsgroup `comp.os.ms-windows` belongs to the `comp` (computing) category, the `os` (operating system) subcategory, and the `ms-windows` specific category. However, articles are only posted to the leaves of the hierarchy, that is, in our example no documents are members of the categories `comp` or `comp.os`.

In our experiment, we treat the collection as a two level hierarchy. Using this approach, there are seven categories at the parent level and twenty categories at the child level. We train our categorisers at the parent level using all documents from all children of the parent.

Partial DMOZ Collection

The *Open Directory Project* is hosted and administered by the Netscape Communication Corporation. This project — also known as DMOZ or Directory Mozilla — provides an extensive hierarchy of human-edited web directories. We selected 53 categories that are structured hierarchically into a four level hierarchy. For each of the categories, we retrieved using a web spider up to 150 websites. Each web site was crawled to a depth of two or to a maximum of 10 web pages, whichever limit is reached first. We extracted words [23] from each document, and removed documents that had no content; for example, we removed documents that resulted from the HTTP response 404 or "Not Found".

The final collection contained 5,296 documents. We divided the collection into 3,550 training documents and 1,747 test documents. There are an average of 1,891 terms per training document.

5. EXPERIMENTS

We describe in this section our experiments with the categorisers and collections we have described. In most of our experiments, we use *single category assignment*, that is, we assign exactly one category label to each test document. We show a comparison of the results of document-wise feature selection under multiple and single category assignment using our SVM categoriser.

To experiment with document-based feature selection, we have trained categorisers for each collection. Specifically, for each collection, and for each of the Rocchio, TDIDF, and probabilistic categorisers we:

- Trained three classes of leaf level categoriser:
  1. A categoriser that uses all features from all training documents, that is, there is no feature selection process. This categoriser is used as a baseline, where documents are only assigned to leaf categories and the hierarchy is not considered in the categorisation process
  2. Two categorisers that use well-known feature selection techniques: first, we remove features from documents that are in a *small stoplist* that contains 75 words that are articles, prepositions, and some very common words such as *a, about, are, and, he, she, it*, and *more*; and, second, we remove words in a *smart stoplist* that contains around 500 terms removed in the SMART retrieval system [16]. We have found in experiments — which we do not report in detail here — that our stopping approach is similar in performance to a DF-thresholding feature selection strategy
  3. Up to ten categorisers that use the first m words feature selection strategy. For the RTSC collection, we developed a categoriser that uses only the first ten words of each training document, and then repeated this process for multiples of ten words between 20 and 100. For the 20-Newsgroups collections, we used multiples of 20 from 20 to 180 words, and for the Partial-DMOZ collection we used multiples of 200 from 200 to 2000 words. The number of words that are removed was empirically determined and based on the average number of words per document
- Trained three classes of *non-leaf level* categoriser. The three classes are the same as those described above for the leaf level categories. For example, the RTSC collection has one non-leaf or *parent* level, and the categorisers at this level are able to determine which of the parent categories best matches a document. These classes of categoriser are used for comparison purposes only: documents are never assigned to non-leaf nodes in the hierarchy.
- Trained three classes of top-down categoriser:
  1. A categoriser that uses all features. In this process, we determine which category at the highest level in the hierarchy best matches the document and select this as the level one category. Then, we compare the document to each of the level one category's children, and again select the best-matching category. We repeat this process until we assign a leaf level category, that is, we traverse one branch of the hierarchy to select the best-matching leaf category
  2. Up to ten top-down categorisers that use first m words feature selection. We used the same fixed numbers of terms that we have described above for leaf level categorisation
  3. Two top-down categorisers that use the two stoplists we have described above for leaf level categorisation

[3]This data is available from `http://www.ai.mit.edu/people/jrennie/20_newsgroups/`

Number of Terms Selected per Document	Parent Level	Child Level	Hierarchical
20	**82.76**	65.66	71.37
40	**83.85**	**69.59**	**73.26**
60	**84.11** (+1.41)	**70.25**	**73.74** (+1.72)
80	**83.79**	**70.54** (+1.45)	**73.65**
100	**83.73**	**70.72**	**73.29**
120	**83.73**	**70.66**	**73.29**
Small Stoplist	82.34	68.46	**72.78**
Smart Stoplist	**82.76**	68.43	**73.07**
All	82.70	69.09	72.02

Table 1: Performance of Rocchio Categorisation on the RTSC Collection. The second and third columns of the final row show the results of categorisation into the parent level and the child level using all features; the child level figure of 69.09% is a baseline. The rightmost column in the final row shows the result of top-down hierarchical categorisation using all features. The remaining rows show comparisons of first m words feature selection with different values of m, and a static feature selection strategy using two stoplists. Values that improve on the baseline are shown in bold face, and the maximum absolute improvements are shown bracketed.

Using this approach, we are able to compare the effects of different feature selection strategies in both flat and hierarchical categorisation.

Evaluation

We use F_1 to measure the categorisation performance [20]. This is a single measurement calculated by balancing the well-known recall and precision measures [24] to the same importance level:

$$F_1 = \frac{2rp}{r+p}$$

We compute the F_1 using the microaverage over all test documents.

The documents in the RTSC collection may belong to more than one category; indeed, there are 505 documents in the training set and 188 in the test set that are multiply assigned. In this case, we use a loose evaluation by applying a simplistic performance measure: a document is assumed to be correctly categorised if the category suggested is in the set of correct labels for the test document.

Results

For compactness, we present only selected results in this section. Table 1 shows the typical results we found when categorising into the RTSC collection; in this table we show the result of using Rocchio categorisers from each of our three classes. Hierarchical categorisation using all terms is around 3% more accurate than child level (flat) categorisation with all terms, a result that is consistent with those found in other hierarchical experiments [3, 4, 5, 21, 22]. Our results also show that the stoplist feature selection techniques have small positive or negative effects on performance: in all cases, the stopping results are within 1% of the performance of using all terms. However, the significant result is the effect of our first m words feature selection technique: when the first 80 terms from each document are used in child categorisation, the performance is 1.5% better than the baseline with all terms. Moreover, less terms are needed to achieve peak performance in the parent level and almost all values we tested improved performance. First m words feature selection also improves hierarchical categorisation: using 60 terms improves performance by 1.7% compared to the baseline hierarchical performance and more than 3.5% compared to baseline flat categorisation.

Table 2 shows similar trends using the 20-newsgroup collection and the Rocchio categoriser classes; again, we found similar results with the TFIDF and probabilistic categorisers. First m words feature selection is again more effective than using all terms except when only 20 terms are used. Similarly, fewer terms on average are required for peak performance at the parent level than at the child. The only significant difference is that hierarchical categorisation is less accurate than flat categorisation. We have identified that the high error rate is largely due to by two closely-related categories that do not share the same parent category: this leads to errors in selecting the correct path in the top-down approach. (Identifying incorrectly formed hierarchies or improving hierarchical structures is outside the scope of this paper.)

In Table 3 we show the results of using the same 20-newsgroup collection but using SVM categorisers. In all cases, we use a linear kernel function; we would expect that accuracy of the SVM categorisers could be improved with more complex functions with the trade-off that training is more computationally expensive. The three columns to the right of the table show the same experiments as in Tables 1 and 2. Again, the relative performance of schemes shows a similar trend: using fewer terms improves performance, with the difference that all schemes — parent, child, and hierarchical — perform best with 40 terms. The two multiple assignment columns show results if documents are able to be assigned to more than one of the categories at each level; this is the usual application of an SVM categoriser, that is, SVMs are generally used to make binary assignment decisions for each category. We are unsure why multiple assignment with first m words feature selection is less accurate than the baseline and plan investigation in our future work.

Tables 4 and 5 show the performance on the Partial DMOZ collection of our TFIDF and probabilistic classes of categoriser. The results again reinforce that accurate categorisation is possible using first m words feature selection and that, in general, categorisation at higher levels in the hi-

Number of Terms Selected Per Document	Parent Level	Child Level	Hierarchical
20	92.57	89.46	88.14
40	**93.59**	**91.29**	**89.73**
60	**93.73**	**91.20**	**90.18**
80	**93.97** (+1.40)	**91.36**	**90.43** (+0.61)
100	**93.96**	**91.25**	**90.38**
120	**93.97**	**91.44** (+1.44)	**90.64**
140	**93.86**	**91.15**	**90.43**
Small Stoplist	**92.82**	**90.13**	**89.04**
Smart Stoplist	**93.02**	**90.02**	**89.35**
All	92.57	90.00	88.82

Table 2: Performance of Rocchio Categorisation on the 20-Newsgroup Collection. The structure of the table is described further in the caption of Table 1.

Number of Terms	Multiple Assignment		Single Assignment		
Selected per Document	Parent	Child	Parent	Child	Hierarchical
20	91.27	85.17	**93.03**	**86.56**	**87.42**
40	91.07	85.39	**93.62** (+0.75)	**87.37** (+4.40)	**87.91** (+4.99)
60	90.84	86.04	**93.10**	**86.94**	**87.72**
80	90.76	86.24	**93.41**	**86.99**	**87.71**
100	90.75	86.43	**93.37**	**85.92**	**86.67**
180	91.16	87.20	92.22	**84.95**	**84.96**
Small Stoplist	**92.04** (+0.04)	**88.21** (+0.02)	**92.95**	82.62	82.57
Smart Stoplist	91.94	88.12	**92.95**	**83.24**	**83.05**
All	92.00	88.19	92.87	82.97	82.92

Table 3: Feature Selection on the 20-Newsgroup collection using a Support Vector Machine. The fourth and fifth columns of the final row show the results of categorisation into the parent level and the child level using all features; the child level figure is a baseline. The rightmost column shows the result of top-down hierarchical categorisation. The second and third columns of the final row show the results of permitting more than one category to be assigned to each document. The other rows show comparisons of first m words feature selection with different values of m, and a static feature selection strategy using two stoplists. Values that improve on the baseline are shown in bold face, and the maximum absolute improvements are shown bracketed.

Number of Terms Selected per Document	Level 1	Level 2	Level 3	Level 4	Hierarchical
200	**93.19** (+9.16)	**85.86**	**82.48** (+3.09)	**76.02**	**74.01**
400	**93.13**	**85.92** (+7.5)	**81.85**	**75.96**	**74.13**
600	**93.02**	**85.75**	**82.20**	**76.53** (+4.73)	**74.53** (+12.25)
800	**93.02**	**85.52**	**82.14**	**76.47**	**73.84**
1000	**92.90**	**85.40**	**81.97**	**76.36**	**73.84**
2000	**92.90**	**85.17**	**81.63**	**75.96**	**73.27**
Small Stoplist	84.03	78.42	79.39	71.89	62.28
Smart Stoplist	83.92	78.36	79.39	71.89	62.11
All	84.03	78.42	79.39	71.89	62.28

Table 4: Performance of TFIDF Categorisation on the Partial DMOZ Collection. The structure of the table is described further in the caption of Table 1.

Number of Terms Selected per Document	Level 1 (L1)	Level 2 (L2)	Level 3 (L3)	Level 4 (L4)	Hierarchical
200	**92.67** (+1.31)	**85.00** (+1.43)	**80.42** (+5.21)	**73.73**	**72.12**
400	**92.39**	**84.43**	**79.39**	**74.41** (+8.58)	**71.04** (+6.59)
600	**92.56**	**84.09**	**78.99**	**74.41**	**70.23**
800	**92.33**	**84.09**	**78.88**	**74.01**	**70.23**
1000	**92.27**	**84.03**	**78.88**	**74.18**	**70.12**
2000	**92.50**	**84.03**	**77.90**	**73.67**	**69.83**
Small Stoplist	91.36	83.51	75.16	65.83	64.45
Smart Stoplist	91.36	**83.57**	75.16	65.83	64.57
All	91.36	83.51	75.21	65.83	64.45

Table 5: Performance of Probabilistic Categorisation on the Partial DMOZ Collection. The structure of the table is described further in the caption of Table 1.

erarchy requires fewer terms. Moreover, stoplist-based feature selection has only a small impact on accuracy. Perhaps the most surprising result with the Partial DMOZ collection is the poor performance of hierarchical categorisation relative to the baseline level four (child) results. As has also been noted by others [5], we attribute this result to the approach of top-down hierarchical categorisation: errors in higher level categorisation are not recoverable in the lower level process and, therefore, hierarchical categorisation is less effective in deeper hierarchies.

In general, our experiments show that the use of fewer terms when selected using our techniques can deliver better performance than use of all features. Moreover, the techniques can be applied to categories at any hierarchical level without requiring feature selection analysis. Overall, therefore, our feature selection technique works well for hierarchical categorisation.

6. CONCLUSIONS AND FUTURE WORK

In this paper we have experimentally investigated the performance of feature selection techniques for hierarchical categorisation. Overall, we have shown that effective categorisers can be developed using selected terms from each document rather than all available terms. We have also found that terms are best selected using a simple first m word approach where the first section of each document is used in training a categoriser. Our results also show the effects of selecting fewer terms in different levels of hierarchies and, in general, we have found that fewer terms are required at higher levels to develop an effective categoriser.

We conclude that our simple feature selection technique is useful for developing better automatic categorisers. While feature selection reduces the number of features, it can consistently be used to develop a more efficient and effective categoriser at any hierarchical level. We also believe that by applying other hierarchical algorithms — as alternatives to top-down decision making — that hierarchical categorisation results can be further improved.

We plan to extend this work by investigating other simple feature selection schemes for hierarchical catagorisation. We also plan to experiment with other categorisers and to investigate feature selection in multiple category assignment.

7. REFERENCES

[1] C. Apte, F. Damerau, and S. Weiss. Automated learning of decision rules for text categorization. *ACM Transactions on Information Systems*, 12(3):233–251, 1994.

[2] L.D. Baker and A.K. McCallum. Distributional clustering of words for text classification. In R. Wilkinson, B. Croft, K. van Rijsbergen, A. Moffat, and J. Zobel, editors, *Proc. ACM-SIGIR International Conference on Research and Development in Information Retrieval*, pages 81–89, Melbourne, Australia, July 1998.

[3] S. D'Alessio, K. Murray, R.Schiaffino, and A. Kershenbaum. The effect of using hierarchical classifiers in text categorization. In *Proceeding of RIAO-00, 6th International Conference "Recherche d'Information Assistee par Ordinateur"*, pages 302–313, Paris, 2000.

[4] S. D'Alessio, K. Murray, R. Schiaffino, and A. Kershenbaum. Category levels in hierarchical text categorization. In *Proc. of EMNLP-98, 3rd Conference on Empirical Methods in Natural Language Processing*, Granada, Spain, 1998. Association for Computational Linguistics, Morristown.

[5] S. T. Dumais and H. Chen. Hierarchical classification of Web content. In N.J. Belkin, P. Ingwersen, and M.-K. Leong, editors, *Proc. ACM-SIGIR International Conference on Research and Development in Information Retrieval*, pages 256–263, Athens, 2000.

[6] P.J. Hayes and S.P. Weinstein. CONSTRUE/TIS: a system for content-based indexing of a database of news stories. In A. Rappaport and R. Smith, editors, *Proceedings of IAAI-90, 2nd Conference on Innovative Applications of Artificial Intelligence*, pages 49–66. AAAI Press, Menlo Park, 1990.

[7] T. Joachims. A probabilistic analysis of the Rocchio algorithm with TFIDF for text categorization. In D.H. Fisher, editor, *Proc. of the 14th International Conference on Machine Learning*, pages 143–151, Nashville, 1997. Morgan Kaufmann, San Francisco.

[8] T. Joachims. Text categorization with support vector machines: Learning with many relevant features. In C. Nédellec and C. Rouveirol, editors, *Proceedings of the 10th European Conference on Machine Learning (ECML-98)*, volume 1398, pages 137–142, Berlin, 1998. Springer.

[9] T. Joachims. Making large-scale SVM learning practical. In B. Schölkopf, C. Burges, and A. Smola, editors, *Advances in Kernel Methods - Support Vector Learning*, chapter 11, pages 169–184. The MIT Press, 1999.

[10] D. Koller and M. Sahami. Hierarchically classifying documents using very few words. In D.H. Fisher, editor, *Proc. of the 14th International Conference on Machine Learning (ICML97)*, pages 170–178, Nashville, 1997. Morgan Kaufmann, San Francisco.

[11] D.D. Lewis, R.E. Schapire, J.P. Callan, and R. Papka. Training algorithms for linear text classifiers. In Hans-Peter Frei, Donna Harman, Peter Schäuble, and Ross Wilkinson, editors, *Proc. ACM-SIGIR International Conference on Research and Development in Information Retrieval*, pages 298–306, Zurich, Switzerland, 1996.

[12] D. Mladenic and M. Grobelnik. Feature selection for classification based on text hierarchy. In *Working notes of Learning from Text and the Web, Conference on Automated Learning and Discovery CONALD-98*, Pittsburg, USA, 1998.

[13] S.E. Robertson and K. Sparck-Jones. Relevance weighting of search terms. *Journal of the American Society for Information Science*, pages 129–146, May 1976.

[14] J.J. Rocchio. Relevance feedback in information retrieval. In *The Smart Retrieval System — Experiments in Automatic Document Processing*, pages 313–323. Prentice-Hall, Englewood, Cliffs, New Jersey, 1971.

[15] M. E. Ruiz and P. Srinivasan. Hierarchical neural networks for text categorization. In M.A. Hearst, F. Gey, and R. Tong, editors, *Proc. ACM-SIGIR International Conference on Research and Development in Information Retrieval*, pages 281–282, Berkeley, CA, 1999.

[16] G. Salton, editor. *The SMART Retrieval System—Experiments in Automatic Document Processing*. Prentice-Hall, New Jersey, 1971.

[17] H. Schütze, D. A. Hull, and J. O. Pedersen. A comparison of classifiers and document representations for the routing problem. In E.A. Fox, P. Ingwersen, and R. Fidel, editors, *Proc. ACM-SIGIR International Conference on Research and Development in Information Retrieval*, pages 229–237, Seattle, WA, 1995.

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

[19] V. Shanks and H.E. Williams. Fast categorisation of large document collections. In *8th International Symposium on String Processing and Information Retrieval (SPIRE2001)*, pages 194–204, San Rafael, Chile, 2001.

[20] C.J. van Rijsbergen. *Information Retrieval*. Butterworths, second edition, 1979.

[21] A.S. Weigend, E.D. Wiener, and J.O. Pedersen. Exploiting hierarchy in text categorization. *Information Retrieval*, 1(3):193–216, 1999.

[22] W. Wibowo and H.E. Williams. On using hierarchies for document classification. In *Proc. Australian Document Computing Conference*, pages 31–37, Coffs Harbour, Australia, 1999.

[23] H.E. Williams and J. Zobel. Searchable words on the web. *International Journal of Digital Libraries*. To appear.

[24] I.H. Witten, A. Moffat, and T.C. Bell. *Managing Gigabytes: Compressing and Indexing Documents and Images*. Morgan Kaufmann Publishers, Los Altos, CA 94022, USA, second edition, 1999.

[25] Y. Yang. Noise reduction in a statistical approach to text categorization. In E.A. Fox, P. Ingwersen, and R. Fidel, editors, *Proc. ACM-SIGIR International Conference on Research and Development in Information Retrieval*, pages 256–263, Seattle, Washington, 1995.

[26] Y. Yang and J.O. Pedersen. A comparative study on feature selection in text categorization. In D.H. Fisher, editor, *Proceedings of ICML-97, 14th International Conference on Machine Learning*, pages 412–420, Nashville, TX, 1997. Morgan Kaufmann Publishers, San Francisco.