Working Paper Series
ISSN 1170-487X

Text categorization using compression models

by Eibe Frank, Chang Chui and Ian H Witten

Working Paper 00/2
January 2000

Department of Computer Science
The University of Waikato
Private Bag 3105
Hamilton, New Zealand

Text categorization using compression models

Eibe Frank, Chang Chui and Ian H. Witten

Department of Computer Science
University of Waikato
Hamilton, New Zealand
{eibe, ckc1, ihw}@cs.waikato.ac.nz

1 Introduction

Text categorization, or the assignment of natural language texts to predefined categories based on their content, is of growing importance as the volume of information available on the internet continues to overwhelm us. The use of predefined categories implies a "supervised learning" approach to categorization, where already-classified articles—which effectively define the categories—are used as "training data" to build a model that can be used for classifying new articles that comprise the "test data." This contrasts with "unsupervised" learning, where there is no training data and clusters of like documents are sought amongst the test articles. With supervised learning, meaningful labels (such as keyphrases) are attached to the training documents, and appropriate labels can be assigned automatically to test documents depending on which category they fall into.

Text categorization is a hot topic in machine learning. Typical approaches extract "features" from articles, and use the feature vectors as input to a machine learning scheme that learns how to classify articles. The features are generally words. Because there are so many of them, a selection process is applied to determine the most important ones, and the remainder are discarded. This "bag of words" model neglects word order and contextual effects. It also raises some problems: how to define a "word," what to do with numbers and other non-alphabetic strings, and whether to apply stemming.

It has often been observed that compression seems to provide a very promising alternative approach to categorization. The overall compression of an article with respect to different models can be compared to see which one it fits most closely. Such a scheme has several potential advantages:

- it yields an overall judgement on the document as a whole, rather than discarding information by pre-selecting features;
- it avoids the messy and rather artificial problem of defining word boundaries;
- it deals uniformly with morphological variants of words;
- depending on the model (and its order), it can take account of phrasal effects that span word boundaries;
- it offers a uniform way of dealing with different types of documents—for example, arbitrary files in a computer system;
- it generally minimizes arbitrary decisions that inevitably need to be taken to render any learning scheme practical.

Furthermore, a compression-based approach to text categorization does offer potential improvements in compression performance, by selecting the most suitable model for each

text on an individual basis and transmitting its identity to the receiver—although it is categorization, not compression performance, that is our primary motivator.

We have performed extensive experiments on the use of compression models for categorization using a standard dataset. This has involved working out how to deal with the (normal) situation where a document may belong to several categories (not merely choosing the one that it fits best). We report some encouraging results on two-category situations, and the results on the general problem seem reasonably impressive—in one case outstanding. Compression-based methods certainly succeed in categorizing the majority of documents correctly, and compare quite well with simple machine learning schemes.

However, we find that compression-based methods do not compete with the published state of the art in the use of machine learning for text categorization (although, as mentioned in Section 2.1, the art is rather difficult to replicate because it is not fully described in the literature). Some reasons why this is the case are discussed in the closing section.

We have two overall conclusions. First, a negative result: we do not recommend the use of compression models for text categorization if one seeks the best possible categorization performance. Second, a methodological point: results in this area should be evaluated comparatively with respect to the state of the art—it is too easy to give a positive impression by avoiding direct, quantitative, comparison with other work.

2 Existing approaches to text categorization

Text categorization is a supervised learning task where a test document is classified into categories using a mapping derived from a set of labeled training documents. This is a standard setting in machine learning, and there is a host of learning algorithms for such problems (Witten and Frank, 2000)—many of which have also been applied to text categorization. They all require documents to be transformed into feature vectors before learning can take place. In the following we briefly review how this is done, and which supervised learning schemes have been applied.

2.1 Data preparation

Standard approaches to text categorization using supervised learning represent each document by the set of words it contains. Generally, each word is a binary feature (Dumais *et al.*, 1998), although more complicated procedures based on combinations of term frequencies and inverse document frequencies are also possible (Yang and Pedersen, 1997).

The low-level problems of word identification and extraction are generally brushed under the carpet. For example, Dumais *et al.* (1998) state only that "text files are processed using Microsoft's Index Server." Yang (1999) uses the SMART system (Salton, 1989) for removing stop words and stemming. Yet it is possible that text categorization results are quite sensitive to the precise details of word extraction. For example, financial articles may be distinguished by a prevalence of numeric dollar figures, which may well be discarded wholesale by a preprocessor. There have been no studies of the robustness of text categorization to changes in these low-level decisions.

Most schemes perform feature extraction prior to learning by selecting a small number of words to participate in the learning phase and discarding the rest (Yang and Pedersen, 1997). For learning schemes that are sensitive to irrelevant features, this improves performance markedly; if a scheme's computational complexity depends heavily on the number of features, it may be the only way to make learning feasible. For example, Dumais *et*

al. (1998) selected between 50 and 300 features for each category, based on a mutual information measure between a feature and a category.

2.2 Learning schemes

Many supervised learning methods have been applied to the problem of text categorization. *Information retrieval metrics*, used by full-text retrieval systems to allow users to sharpen their queries using relevance feedback (Rocchio, 1971), have been used by imagining a query that contains all the words in the test document and using weights derived from the documents in each class (Dumais *et al.*, 1998). *Naive Bayes* classifiers estimate the probability of each feature given each category from the training data (Langley *et al.*, 1992), assuming statistical independence of the features (which is why the method is called "naive"). The *Bayes net* technique (Sahami, 1996) models limited dependence between different features, and has also been applied to text categorization (Dumais *et al.*, 1998). *Nearest-neighbor classifiers* assign to a test document the class of the training document that most closely matches it. A "divide-and-conquer" approach leads naturally to a *decision tree* (Lewis and Ringuette, 1994). A *linear model* assigns weights to the features during the training phase, and sums them for each feature that appears in the test document (Lewis *et al.*, 1996). *Neural nets* use multi-stage combinations of simple non-linear models (Ng *et al.*, 1997). *Linear support vector machines* select a small number of critical boundary instances (i.e. documents) from each category and build a linear discriminant function that separates them as widely as possible.

The best results for text categorization have been obtained using support vector machines (Dumais *et al.*, 1998), committees of decision trees (Apte *et al.*, 1998), and nearest-neighbor classifiers that consider k nearest neighbors instead of only one (Yang, 1999).

3 Text categorization using PPM

All these approaches to text categorization share the disadvantage that input documents must be converted into feature vectors before they can be processed. This involves many arbitrary decisions, making experimental results hard to replicate. The effect of these decisions has never been thoroughly investigated. Pre-processing requires language-dependent mechanisms like stemming that may not be readily available for the language in question. Finally, if text categorization is considered from a broader point of view—where a "text" can be any character stream—word-based approaches will necessarily exhibit deficiencies. Ideally, a text categorization scheme should be able to classify arbitrary files, not just English-language documents. Its success should depend only on the availability of sufficient training data, not on the type of documents to which it is applied.

In contrast to general-purpose classification methods that require extensive data preparation, compression techniques deal with arbitrary sequences of characters. Hence they offer the prospect of a uniform approach to text categorization. The question is whether they can be successfully applied to the task of discriminating between classes of documents. In the following we investigate this question using the PPM compression scheme with order 2 and escape method C (Bell *et al.*, 1990). Other orders were tried, but both lower and higher choices were found to degrade performance in almost all cases—presumably because the amount of training data available is insufficient to justify more complex models.

	Training data		Test data	
	Articles	Text (Kb)	Articles	Text (Kb)
corn	181	210	56	81
corporate acquisitions	1650	1307	719	542
crude oil	389	522	189	206
earnings	2877	1460	1087	457
grain	433	478	149	166
interest	347	329	131	147
money market	538	610	179	211
shipping	197	212	89	94
trade issues	369	569	117	180
wheat	212	235	71	77

Table 1: Corpus of Reuters articles used in experiments

3.1 The benchmark data

All our results are based on the Reuters-21578 collection of newswire stories,[1] divided into training and test documents using the ModApte split—the standard testbed for the evaluation of text categorization schemes. In total there are 12,902 stories, averaging 200 words each, that have been classified into 118 categories. However, the distribution of stories among categories is highly skewed: the ten largest contain 75% of stories. These ten categories—earnings, corporate acquisitions, money market, grain, crude oil, trade issues, interest, shipping, wheat, and corn—are shown in Table 1, along with the number of training and test stories that each one contains. A story does not necessarily belong to only one category; many stories are assigned to multiple categories, and some are not assigned to any category at all.

3.2 Experiments using pairwise discrimination

Application of a straightforward compression methodology to the problem of text categorization quickly yields encouraging results. Consider the two-class case. To distinguish documents of class A from documents of class B, we form separate compression models M_A and M_B from the training documents of each class. Then, given a test document (different from the training documents), we compress it according to each model and calculate the gain in per-symbol compression obtained by using M_A instead of M_B. We assign the document to one or the other class depending on whether this difference is positive or negative, on the principle that M_A will compress documents of class A better and similarly for M_B.

Figure 3.2 shows results for ten pairs of categories from the Reuters data, using the ModApte split from Table 1.[2] The graphs show, on the vertical axis, the difference in compression. The vertical line to the left of each plot shows the test documents of one class, and the vertical line to the right shows the test documents of the other. The fact that almost all the points in the lefthand line lie above the zero line, and almost all in the righthand line lie below it, indicates that almost all test documents are classified correctly. Superimposed on each vertical line is a box whose center indicates the average compression difference for that class, and whose extent indicates the standard deviation of the distribution. The lefthand boxes lie comfortably above the line and the righthand ones comfortably below it.

In Figure 3.2a just one article is miscategorized, and that by only a small margin. In Figures 3.2b and 3.2c there are no miscategorizations. Figure 3.2d also shows very few

[1] The collection is publicly available at *www.research.att.com/~lewis/reuters21578.html*.

[2] However, for legibility Figure 3.2 only shows one-half of the test results.

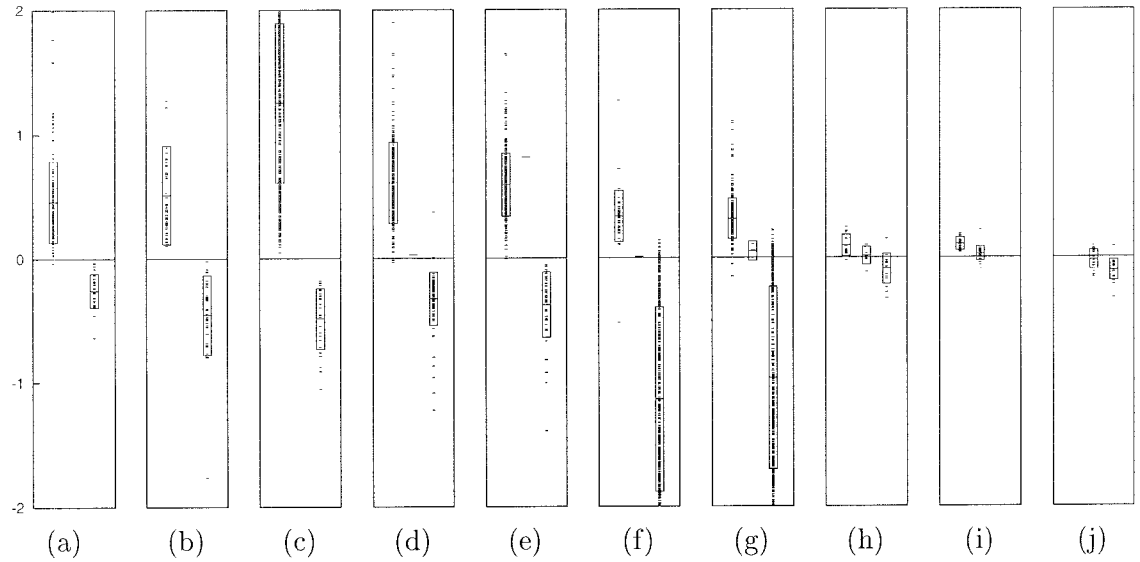

Figure 1: Pairwise discrimination using compression: (a) money market vs shipping; (b) grain vs interest; (c) earnings vs wheat; (d) corporate acquisitions vs trade issues; (e) corporate acquisitions vs grain; (f) crude oil vs earnings; (g) corporate acquisitions vs earnings; (h) corn vs wheat; (i) grain vs wheat; (j) corn vs grain.

errors, but one is by a rather large margin. Figures 3.2d and 3.2e show a new phenomenon: an article (just one in each case) that is assigned to *both* categories, displayed in the middle. Clearly a pairwise discrimination policy cannot handle such cases. In Figure 3.2d the doubly-classified article is near the zero point, while in Figure 3.2e it is far above it.

Figure 3.2f–j show some less satisfactory results. In Figure 3.2f, several (19) *earnings* articles are misclassified as *crude oil*, one *crude oil* article is miscategorized by a large margin, and there is one article that belongs to both categories. Results would be improved by choosing a small positive number, instead of zero, as the threshold. In Figure 3.2g many (44) *earnings* articles are misclassified as *corporate acquisitions* and two *corporate acquisitions* articles are assigned to the category *earnings*; in addition, there are two articles that belong to both. Again, results would be improved by choosing a small positive threshold. Figure 3.2h shows an article being misclassified by a rather large margin—the compression metric assigns one of the *wheat* articles to the *corn* category by a very large margin; in fact, this article is more *corn*-like (one hesitates to say "corny") than most *corn* articles. A significant number of articles belong to both categories. Figures 3.2i and 3.2j show an extreme situation where all but one of the *wheat* articles, and all of the *corn* articles, also belong to the *grain* category. *Wheat* and *corn* evidently form a subclass of *grain* and cannot be distinguished using this methodology.

Table 2 summarizes the situation for all possible pairwise discriminations. We compress articles corresponding to the row but not the column according to (a) the model corresponding to the row and (b) the model corresponding to the column, and present the mean compression difference in bits/character, (b) – (a), averaged over the test articles. The means are positive, indicating that they lie on the correct side of the zero line. The only exception is that corresponding to the right-hand "bar" of Figure 3.2i, *wheat* vs *grain*, where there is only one article on *wheat* that is not also on *grain*, and that article is classified

	corn	corp. acq.	crude oil	earn-ings	grain	inter-est	money market	ship-ping	trade issues	wheat
corn	—	0.43	0.39	0.53	0.00	0.62	0.50	0.38	0.45	0.09
corp. acq.	0.79	—	0.40	0.31	0.59	0.65	0.63	0.47	0.61	0.68
crude oil	0.45	0.26	—	0.35	0.37	0.48	0.43	0.37	0.38	0.46
earnings	1.47	0.99	1.15	—	1.18	1.53	1.48	1.34	1.39	1.27
grain	0.13	0.36	0.33	0.47	—	0.54	0.44	0.33	0.41	0.11
interest	0.65	0.26	0.33	0.36	0.46	—	0.17	0.52	0.55	0.59
money market	0.57	0.39	0.37	0.50	0.44	0.18	—	0.46	0.32	0.55
shipping	0.27	0.19	0.16	0.32	0.20	0.38	0.29	—	0.31	0.24
trade issues	0.36	0.32	0.25	0.45	0.26	0.33	0.20	0.35	—	0.34
wheat	0.11	0.37	0.35	0.49	–0.06	0.58	0.46	0.33	0.45	—

Table 2: Mean difference in compression between model corresponding to row and model corresponding to column, for articles corresponding to row but not to column

incorrectly.

These results paint a generally encouraging picture, but underline the fact that the pairwise methodology needs extending to cope with multiply-classified articles.

3.3 Building positive and negative models

For multiply-classified articles, we decide whether a model belongs to a particular category independently of whether it belongs to any other category. We build positive and negative models for each category, the first from all articles that belong to the category and the second from those that do not. For a particular category C, call these models M_P and M_N respectively.

Given a new article A, denote its length when compressed according to these models by $L[A|M_P]$ and $L[A|M_N]$. From these lengths, the article's probability given the categories C and $\bar{C}$ is:

$$Pr[A|C] = 2^{-L[A|M_P]} \qquad Pr[A|\bar{C}] = 2^{-L[A|M_N]}$$

Bayes' formula gives the probability that a particular article A belongs to category C:

$$Pr[C|A] = \frac{Pr[A|C]Pr[C]}{Pr[A|C]Pr[C] + Pr[A|\bar{C}]Pr[\bar{C}]}$$

The prior probability of C is the proportion of articles belonging to that category, and the denominator is the prior probability of article A.

3.3.1 Setting the threshold

Now we turn to the question of deciding whether a new article should in fact be assigned to C or not. This presents the tradeoff between making the decision liberally, increasing the chance that a category-C article is correctly identified but also increasing the number of "false positives"; or conservatively, reducing the number of false positives but also increasing the number of "false negatives." This tradeoff is familiar in information retrieval, where a search engine must decide how long a list of articles to present to the user, balancing the disadvantage of too many false positives (irrelevant documents that are displayed) if the list is too long against too many false negatives (relevant documents that are not displayed) if it is too short.

Following standard usage, we quantify this tradeoff in terms of recall and precision. In order to allow comparison of our results with others, we strive to maximize the average of recall and precision—a figure that is called the "breakeven point."

	PPM	Naive Bayes	LSVM	Overlap	Dumais *et al.* (1998)	Number of features 5	50	300
corn	54.2	65.3	90.3	0.049	65.3	83.3	61.2	57.8
corporate acquisitions	91.0	87.8	93.6	0.030	87.8	66.2	84.5	85.7
crude oil	80.7	79.5	88.9	0.044	79.5	76.9	82.6	83.5
earnings	96.3	95.9	98.0	0.020	95.9	91.1	95.1	96.4
grain	74.6	78.8	94.6	0.038	78.8	84.3	82.2	78.9
interest	60.4	64.9	77.7	0.045	64.9	55.4	59.8	52.8
money market	76.3	56.6	74.5	0.053	56.6	50.9	61.2	61.0
shipping	81.9	85.4	85.6	0.039	85.4	71.3	83.1	83.7
trade issues	65.0	63.9	75.9	0.047	63.9	63.4	67.0	57.0
wheat	64.9	69.7	91.8	0.051	69.7	85.3	74.9	68.2

Table 3: Recall/precision breakeven point for compression-based categorization compared with Naive Bayes and linear support vector machines; also (on the right), subsidiary results for Naive Bayes

The basic strategy is to compare the predicted probability $Pr[C|A]$ with a predetermined threshold t, and declare A to have classification C if the probability exceeds the threshold. The threshold is chosen individually, for each class, to maximize the average of recall and precision for that class. To this end the training data is further divided into a new training set (2/3 of the training data) and a validation set (1/3 of the training data). The threshold t is chosen to maximize the average of recall and precision for the category (the breakeven point) on the validation set. Once it is obtained, maximum utility is made of the training data by rebuilding the models M_P and M_N based on the full training data.

As an additional benefit, threshold selection automatically adjusts for the fact that M_P and M_N are formed from different amounts of training data. In general, one expects to achieve better compression with more training data. On the other hand, the results in Figure 3.2 indicate (to our surprise) that differing amounts of training data do not have a strong influence on pairwise discrimination: it does not seem essential for good performance to compensate for training set size.

3.3.2 Results

Table 3 shows the breakeven points obtained from our experiments, and compares them with the results obtained by Dumais *et al.* (1998) for the Naive Bayes and Linear Support Vector Machine methods. (Ignore the rightmost block of figures; we return to them in Section 4). PPM performs better than Naive Bayes on the six largest categories (*grain* is the only exception) and worse on the four smallest ones. It is almost uniformly inferior to the support vector method, *money market* being the only exception.

Compared to the support vector method, PPM produces particularly bad results on the categories *wheat* and *corn*. These two categories are (almost) proper subsets of the category *grain*. This is because articles in *grain* summarize the result of harvesting grain products—for example, by listing the tonnage obtained for each crop. These articles use very similar terminology. Consequently the model for *wheat* is very likely to assign a high score to *every* article in *grain*.

It is the occurrence of the term "wheat" that is the only notable difference between an article in *grain* that belongs to *wheat* and one that does not. The presence of a single word is unlikely to have a significant effect on overall compression of an article, and this is why PPM performs poorly on these categories.

Support vector machines perform internal feature selection, and can focus on a single

	corn	corp. acq.	crude oil	earn-ings	grain	inter-est	money market	ship-ping	trade issues	wheat
corn	0	0	0	0	43	0	0	0	2	29
corporate acquisitions	0	0	10	18	0	0	0	0	0	0
crude oil	0	5	0	2	2	0	1	14	2	0
earnings	1	8	0	0	1	0	0	0	0	0
grain	0	0	2	0	0	0	0	4	8	0
interest	2	48	11	15	5	0	108	3	20	1
money market	0	0	0	0	0	43	0	0	19	0
shipping	3	0	8	0	7	0	0	0	0	4
trade issues	0	3	4	0	7	16	38	5	0	1
wheat	13	0	0	0	26	0	0	0	2	0

Table 4: "False positive" confusion matrix for the predictions made by PPM

word if that is the only discriminating feature of a category. In comparison, Naive Bayes performs badly on the same categories as PPM (*money market* is the only exception). This is because, like PPM, it has no mechanism for internal feature selection. Section 4 presents empirical evidence for the importance of feature selection in text categorization.

PPM performs badly on *wheat* and *corn* because the category *grain* occurs in substantial numbers in both the positive and the negative training data for these two categories. The effect can be quantified by computing the entropy of the distribution of *grain* articles among the positive and negative training articles. The same entropy figure can be computed for all other categories. The sum of these entropies, weighted according to the prevalence of the corresponding category in the training data, represents a coarse measure of the "overlap," or similarity, between the positive and negative training data for a category.[3]

The *Overlap* column of Table 3 shows this measure. It correlates well with the performance difference between the support vector method and PPM. The only exception is the category *money market*, and we conjecture from Naive Bayes's poor performance on it that this category is an outlier that occurs because it is poorly suited to the word-based approach. Excluding *money market*, the correlation coefficient for the difference in breakeven performance between LSVM and PPM on the one side, and the entropy measure on the other, is 0.71, and the correlation is statistically significant with a p-value of 0.03.

Table 4 summarizes some of the errors made by PPM on the test data. It shows how the false positives associated with the category corresponding to a row are distributed among the categories corresponding to the columns. Most false positives occur when articles belong to related categories. This is particularly striking for *wheat* and *corn*: the first row shows that 29 articles belonging to *corn* are incorrectly identified as *wheat*; the last row shows that 13 *wheat* articles are incorrectly assigned to *corn*. Most false positives for the *wheat* and *corn* models belong to the category *grain*, which comprises *wheat*, *corn*, and several smaller categories (*oat*, *rice*, etc.) This adds further support to the argument that PPM performs poorly with overlapping categories. A similar "false negative" confusion matrix confirms that several articles belonging to both *wheat* and *corn* are not identified as *wheat* (5 out of 15 false negatives).

[3]When calculating the entropy, we divide the weight of an article by the number of categories it belongs to, giving every article a weight of 1 in the final sum.

3.3.3 Modifications

The results in Table 3 were obtained quickly, and we found them encouraging. Subsequently we made many attempts to improve them, all of which met with failure.

In order to force PPM to build models that are more likely to discriminate successfully between similar categories, we experimented with a more costly approach. Instead of building one positive and one negative model, we built one positive model and 117 negative ones for each of the 118 categories. For each negative model we only used articles belonging to the corresponding category that did not occur in the set of positive articles. During classification, an article was assigned to a category if the positive model compressed it more than all negative models did. Results were improved slightly for categories like *wheat* and *corn*. However, the support vector method still performed far better. Moreover, compared to the standard PPM method, performance deteriorated on some other categories.

We also experimented with the following modifications of the standard procedure, none of which produced any significant improvement over the results reported above:

- not rebuilding the two models from the full training data;
- using the same number of stories for building M_P and M_N (usually there are far more stories available for building M_N);
- priming the models with fresh Reuters data from outside the training and test sets;
- priming the models with the full training data (positive and negative articles);
- artificially increasing the counts for the priming data compared with those for the training data and *vice versa*;
- using only a quarter of the original training data for validation;
- using escape method A instead of C;
- using a word model of order 0, escaping to a character model of order 2 for unseen words.

4 The importance of feature selection

In order to test the importance of feature selection we performed experiments with the Naive Bayes learning scheme, varying the number of features it had access to. We employed the standard word-based approach where the occurrence of a particular word is treated as a binary feature. Before the input text was split into words, we removed all non-letter characters and stop words. We did not perform stemming.

Naive Bayes does not incorporate any mechanism for feature selection. Before it is applied, features must be pre-selected according to their influence on category membership. Following Dumais *et al.* (1998), we used a different set of features for each category, choosing the k features that had the greatest mutual information with the category.

The rightmost block of Table 3 shows the breakeven performance for the ten largest categories for three different numbers of features: $k = 5$, $k = 50$ and $k = 300$. It also includes the results obtained by Dumais *et al.* (1998) using fifty features. Although our results are similar overall, they differ slightly for some categories, possibly because we did not perform stemming.

The results show that the optimum number of features varies significantly among categories. For several categories (*earnings*, *corporate acquisitions*, *crude oil* and *shipping*) a large number of features is best. However, for *grain*, *wheat* and *corn* performance peaks with only five features. Moreover, for *wheat* and *corn* the breakeven point increases dramatically when five features are used instead of fifty—and in fact for *wheat* it increases

further when just one feature is used. This is consistent with the conjecture above that often the occurrence of just a few words is sufficient to predict category membership.

5 Conclusions

Compared to state-of-the-art machine learning techniques for categorizing English text, PPM produces inferior results because it is insensitive to subtle differences between articles that belong to a category and those that do not. We do not believe our results are specific to PPM. If the occurrence of a single word determines whether an article belongs to a category or not, any compression scheme will likely fail to classify the article correctly. Machine learning schemes fare better because they automatically eliminate irrelevant features.

Compared to word-based approaches, compression-based methods avoid *ad hoc* decisions when preparing the text for the actual learning task. Morever, compression-based methods apply immediately to the categorization of arbitrary documents, not just English text. However, it is hard to see how efficient feature selection could be incorporated into PPM. Hence it seems appropriate to abandon this method and to move to a classical machine learning setting where, instead of using words, each n-gram is treated as a separate feature for the learning algorithm.

Anecdotal evidence indicates that the idea of using compression to classify documents is one that has been reinvented many times. One of us (IHW) investigated its use for document classification in the mid-1980s. We know of few records of such investigations, although Teahan (1998) concludes, based on some experiments, that compression methods are capable of ascribing authorship and identifying language dialects. We are less sanguine, and tend to believe that compression-based methods will not compete with other, state of the art, methods for such problems. Given our interest in the use of compression for text mining (Witten *et al.*, 1999), we would like to be proved wrong.

References

Bell, T.C., Cleary, J.G. and Witten, I.H. (1990) *Text compression.* Prentice Hall, Englewood Cliffs, NJ.

Dumais, S., Platt, J., Heckerman, D. and Sahami, M. (1998) "Inductive learning algorithms and representations for text categorization." *Proc Int Conf on Info and Knowledge Management*, pp. 148–155.

Langley, P., Iba, W. and Thompson, K. (1992) "An analysis of Bayesian classifiers. In *Proc National Conference on Artificial Intelligence*, pp. 223–228.

Lewis, D.D. and Ringuette, M. (1994) "A comparison of two learning algorithms for text categorization." *Proc Annual Symposium on Document Analysis and Information Retrieval*, pp. 37–50.

Ng, H.T., Goh, W.B. and Low, K.L. (1997) "Feature selection, perceptron learning, and a usability case study for text categorization." *Proc ACM SIGIR*, pp. 67–73.

Rocchio, J.J. Jr. (1971) "Relevance feedback in information retrieval." In *The SMART Retrieval System: Experiments in automatic document processing*, edited by G. Salton, pp. 313–323. Prentice Hall.

Sahami, M. (1996) "Learning limited dependence Bayesian classifiers." *Proc Int Conf on Knowledge Discovery and Data Mining*, pp. 335–338. AAAI Press.

Salton, G. (1989) *Automatic text processing.* Addison-Wesley, Reading, Massachusetts.

Teahan, W.J. (1998) "Modelling English text." Ph.D. Thesis, University of Waikato, New Zealand.

Witten, I.H., Bray, Z., Mahoui, M. and Teahan, W.J. (1999) "Text mining: a new frontier for lossless compression." *Proc Data Compression Conference*, pp. 198–207. IEEE Press, Los Alamitos, CA.

Witten, I.H. and Frank, E. (2000) *Data mining: Practical machine learning tools and techniques with Java implementations.* Morgan Kaufmann, San Francisco, CA.

Yang, Y. (1994) "Expert network: Effective and efficient learning from human decisions in text categorization and retrieval." *Proc ACM SIGIR*, pp. 13–22.

Yang, Y. (1999) "An evaluation of statistical approaches to text categorization." *Information Retrieval*, Vol. 1, No. 1, pp. 69–90.

Yang, Y. and Pederson, J.O. (1997) "A comparative study on feature selection in text categorization." *Proc Int Conf on Machine Learning*, pp. 412-420.