

# How Weak Categorizers Based Upon Different Principles Strengthen Performance

VICTORIA S. UREN<sup>1,3</sup> AND THOMAS R. ADDIS<sup>2</sup>

<sup>1</sup>*Knowledge Media Institute, The Open University, Walton Hall, Milton Keynes, MK7 7AA, UK*

<sup>2</sup>*Division of Computer Science and Mathematics, University of Portsmouth, Portsmouth PO1 2EG, UK*

*Email: v.s.uren@open.ac.uk*

---

Combining the results of classifiers has shown much promise in machine learning generally. However, published work on combining text categorizers suggests that, for this particular application, improvements in performance are hard to attain. Explorative research using a simple voting system is presented and discussed in the light of a probabilistic model that was originally developed for safety critical software. It was found that typical categorization approaches produce predictions which are too similar for combining them to be effective since they tend to fail on the same records. Further experiments using two less orthodox categorizers are also presented which suggest that combining text categorizers can be successful, provided the essential element of ‘difference’ is considered.

*Received 6 July 2001; revised 26 February 2002*

---

## 1. INTRODUCTION

Combining the results of a number of individually trained classification systems to obtain a single, hopefully more accurate, classifier is a technique that has been extensively researched and shows considerable promise on many test sets, e.g. [1]. For methods, such as Bagging, a large number of classifiers are combined. These are typically produced by an ensemble of identical classifiers, which are often neural networks, trained on different randomly chosen sets of instances, e.g. [2, 3]. Alternatively, the predictions of a smaller number of different types of classifier that have been trained on the same data may be combined. Research on combining text categorizers has mainly taken the latter route. This may be because the relatively large numbers of features and data sets used for text prohibit the training of many classifiers.

How has this approach of combining a few categorizers fared? Hull *et al.* considered simple probability averaging strategies and more complex ways of combining the results of four text-filtering methods with different optimization and document representation methods [4]. It was found that the simple strategies could improve on the best categorizer only for ranking documents. They were less good at estimating probabilities and were consistently outperformed by the best single algorithm for a filtering application. The more complex strategies were less successful than the simple ones. Larkey and Croft report a consistent improvement in precision for linear combination of scores from pairs of classifiers in a medical domain and a greater improvement

for a three-way combination [5]. Recall only exceeds that of the better classifier for half the cases, but this is not unreasonable as it would generally be expected that gains in precision would come at the expense of recall. Li and Jain experimented with three different methods for combining the results of four typical classifiers: simple voting and two methods for selecting the classifier with the highest local accuracy for a problem [6]. They found that ‘*Combinations of multiple classifiers did not always improve the classification accuracy compared to the best individual classifier*’. Scott and Matwin report selected breakeven results from a simple voting system made up of rule based classifiers that used different text representations, words, stemmed words, noun phrases etc. [7]. These suggest that performance can be improved over the best single categorizer. Finally, Craven *et al.* tried combining votes from several variants of a naive Bayes classifier in a Web-based application. They report that the combined classifiers were not uniformly better than their constituents [8].

To summarize, combining several text categorizers has had mixed success. Some authors report improvements with combined systems, particularly for precision, but others report systems that give either marginal improvement or no improvement at all. Given the bias towards publishing positive rather than negative results, this suggests that combined systems are not producing the improvements in text categorization applications that one would expect from their success elsewhere. This raises an interesting question: what limits the payback from combining text categorizers?

Li and Jain [6] blamed their failure on ‘data dependence’, without being specific about what features of their data were

---

<sup>3</sup>Author to whom correspondence should be addressed.

causing the problem. However, text is known to present special difficulties because of its high dimensionality [9] and context dependence [10]. These affect the performance of single categorizers and might also affect combinations. Hull *et al.* identified correlations between classifiers as a cause. This seems reasonable: consider the case where a document contains no features that have a strong statistical link to its subject, but an interpretation of the context of the document would make it clear what the subject is. Any algorithm that ignores context and looks only at significant features will have trouble classifying this document correctly. Combining the results of several such algorithms will not magically introduce context sensitivity.

In this paper, we explore a theoretical approach to predicting the performance of combined systems and use its predictions to benchmark the real results of a number of combined systems. First we outline the theoretical framework used. Then we describe two sets of exploratory experiments. In the first experiment, the predictions of five typical categorizers are combined for two different datasets. In the second experiment we took the dataset for which the combined systems had least success and developed two less orthodox categorizers, each of which breaks from the typical categorization approach in some way. Further combined systems are produced which use these new systems with some success.

Our aim in these experiments was to start to clarify why combining text categorizers has not been very effective. Two simple models of combining present themselves: linear combining where the scores of a number of classifiers are combined and simple voting in which the decisions of the classifiers are taken as votes for or against the class. Frameworks exist to analyse both [11, 12]. Linear combining requires that the classifiers combined produce scores which are an accurate estimate of class probability (see [13] for a discussion of a categorizer that does this). If the categorizers are optimized to produce a clear yes or no decision for a particular class, they may be biased to produce scores which are either very high (close to 1) or very low (close to 0). Combining scores of this type would not be expected to give satisfactory results and should be avoided. Voting, on the other hand, allows results to be combined from any kind of categorizer. We wished to use a selection of existing binary categorizers, which might or might not give 'true' estimates of class membership. Consequently voting was better suited to our purpose.

## 2. PROBABILISTIC FRAMEWORK

The framework we have used to make predictions about the expected accuracy of a combined system given the accuracy of its components [12] was originally devised for the design of safety critical software. In safety critical applications, several different versions of a function may be written and run in parallel. The eventual result is determined by taking votes from the competing functions.

The framework has two parts. The first is the independent errors model. This allows us to make predictions of the

performance of majority voting systems assuming that all the systems produce errors randomly. This provides an ideal benchmark against which we can assess by how much combined systems fail to meet our expectations. The second part is the coincident errors model. This makes the assumption that certain records are more likely to cause categorizers to fail than others. This part of the framework provides a rationale for cases, such as text categorization, in which combined systems show limited success.

### 2.1. Independent errors model

Provided all the errors occur independently, the expected performance of a combined system using simple voting can be derived trivially from the multiplication law and the addition law of basic probability theory. 'Independence' in this case means that the chance of any of the categorizers making an erroneous prediction on a document is the same whichever document is being examined. Following the method of Eckhardt and Lee [12] and assuming independent errors and equal error rates for all the categorizers for a combined system with  $N$  categorizers, the expected error rate may be estimated from the binomial expansion

$$\sum_{k>N/2}^N \binom{N}{k} \theta^k (1-\theta)^{N-k} \quad (1)$$

where  $N$  is a positive odd number,  $\theta$  is the error rate and  $k$  is the possible numbers of erroneous votes that can result in a miscategorization (either two or three for a three vote system, three, four or five for a five vote system, etc.). Provided that the individual error rates of the components are less than 0.5, the error rate of a majority voting system with independent errors will be lower than the error rate of the best component. Furthermore, we would expect that the greater the value of  $N$  the more the error rate should fall.

Equation (1) assumes that all the categorizers have identical error rates. This can easily be adapted to the real situation in which the error rates for different categorizers vary. For example, in a three component combined system with error rates for the individual systems of  $\theta_1$ ,  $\theta_2$  and  $\theta_3$ , a record will be miscategorized if two or more categorizers make an error. The expected error rate of the combined system  $\theta_c$  is

$$\theta_c = \theta_1\theta_2\theta_3 + \theta_1\theta_2(1-\theta_3) + \theta_1\theta_3(1-\theta_2) + \theta_2\theta_3(1-\theta_1).$$

The independent errors model provides a benchmark against which we can make qualitative judgements of how close a given combination of categorizers comes to the performance that would be expected in an ideal world of independent categorizers.

### 2.2. Coincident errors model

The coincident errors model is derived directly from the independent errors model described by Equation (1). To apply it, we must understand the concept described by Eckhardt and Lee as coincidence intensity distribution [12]

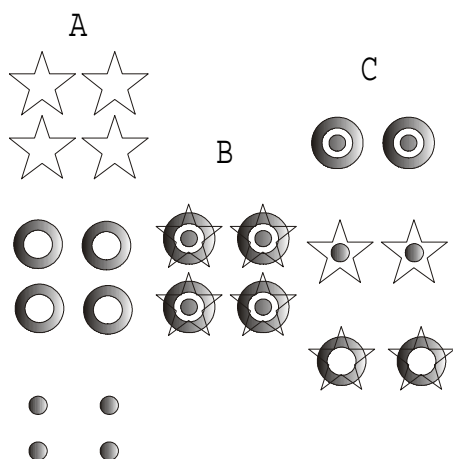


FIGURE 1. The effect of coincident errors in a three-vote system.

and by Hansen and Salamon as the classification difficulty distribution [14]. For the independent errors model it was assumed that any randomly chosen document would have the same probability of causing a categorizer error. The difficulty distribution takes account of the fact that some documents are more likely to cause an error than others. The difficulty distribution  $\mu$  describes the probability that a given document will cause a proportion  $\theta$  of the available categorizers to give an error. If this distribution is known the expected performance of the system assuming coincident errors can be calculated. Drawing on the work in [12], we can derive the following to predict the error rate:

$$\sum_{\theta=1/m}^{\theta=1} \sum_{k>N/2}^N \binom{N}{k} \theta^k (1-\theta)^{N-k} \mu(\theta) \delta\theta \quad (2)$$

where  $\theta$  is a measure of how hard the document is (the proportion of categorizers that will fail—equivalent to the error rate) and  $m$  is the number of categorizers used to derive the difficulty distribution, so that  $1/m$  is equivalent to the interval  $\delta\theta$ . It can be shown that if the difficulty distribution contains any region where the error rate  $\theta$  is equal to 0.5 the reduction in error rate predicted by Equation (2) will reach a limit beyond which adding further categorizers to the system will not improve performance. Furthermore, if any region has an error rate greater than 0.5, then a point will be reached beyond which adding further classifiers causes performance to fall and steadily worsen as more are added.

This is counterintuitive. It would be reasonable to expect that gathering more votes should do no harm, even if it did no good. To understand why this expectation is false we must understand how coincident errors can operate. Figure 1 illustrates how a majority voting system can produce higher error rates than its components. The symbols represent the errors produced by three different categorizers, star, ring and dot, each of which produces four errors on a dataset. These symbols were selected because each symbol can still be seen when any combination are overlaid. The dataset contains at least 12 documents, but those which do not cause errors have been omitted from the figure for the sake of clarity. Two or

more symbols overlaid represents a majority vote that would produce an error for a given record. Scenario A is the ideal. The errors are spread over twelve different records with the effect that no single record has more than one erroneous vote. Since the incorrect category has a minority in every case, A produces a perfect classification. In scenario B all three systems produce identical errors, with the result that the voting system produces the same set of errors as each of its components. Scenario C is the worst case. Here the errors from any one component system coincide with two errors each from the other two. The result is six records each with two erroneous votes, a majority. In scenario C the combined system produces half as many errors again as any component alone.

### 3. TOOLS AND MATERIALS

The theory outlined above indicates that most success will be obtained by combining categorizers that are different enough to avoid coincident errors. To conduct the experiments, we required a selection of text categorizers that might be assumed to be different from each other. Since at this stage there was no clear indication of what kinds of ‘difference’ could be expected to produce independent errors in practice, we selected categorizers that used a wide variety of algorithms and which, as far as possible, represented a reasonable cross section of the methods available. These ‘orthodox’ categorizers use a text representation based on stemmed words and a variety of machine learning methods. The algorithms are described below briefly to illustrate the differences between them.

#### 3.1. Categorizers

A unique letter code has been assigned to each categorizer to identify it in the text and tables.

##### Naive Bayes (B)

The Naive Bayes theorem provides one of the most widely used systems for categorization. It allows us to estimate the conditional probability that a given document belongs to class  $c$  given the words  $w$  that appear in it. This conditional probability can be calculated from the equation where we know the more easily obtained information; the probability of a class ( $c$ ), the probability of words and the probability of the words given a class:

$$P(c|w) = P(c) \times \frac{P(w|c)}{P(w)}$$

If we then assume that the words in the feature vector occur independently of the probability of the words given, the class can be estimated as the product of the individual conditional probabilities of all the words  $w_1 - w_d$ :

$$\prod_{j=1}^d P(w_j|c)$$

Clearly this independence assumption is false but still can produce quite effective results. Where it fails, for example, is in an artificial intelligence context—we know that the word ‘machine’ will not usually occur independently of the word ‘learning’. However it simplifies the problem of estimating  $P(w|c)$  considerably, particularly given the very large number of features typically involved in text classification problems. For this study, the implementation of Naive Bayes built at Carnegie Mellon was made available in the Rainbow package and used with its standard settings [15].

#### IBM Intelligent Miner for Text (I)

A commercial package produced by IBM was used as an example of an industry standard categorizer ([www.software.ibm.com/data/iminer/fortext/](http://www.software.ibm.com/data/iminer/fortext/)). The algorithm underlying this categorizer is not known. However, this should not present too much difficulty for us within the context of voting systems. Since we are only interested in the outcome of the classifier, we can treat it as a black box and assume that its mechanism is potentially distinct. The performance of this black box will tell us if this latter assumption is valid. The version used was an evaluation copy and the experimental work presented here was performed well within the evaluation period.

#### $k$ Nearest Neighbour (K)

$k$  Nearest Neighbour belongs to the class of so called ‘lazy’ learning methods. That is, instead of trying to form a theory about the class, in the form of a function or set of rules, it examines the examples in the training set to find those which are the most similar to the new document. It does this by calculating the distance to all the training examples. The classes of the  $k$  examples which are nearest to the new document in the feature space are examined. The majority class is assigned ( $k$  is usually an odd number to guarantee a majority).

A basic implementation of the algorithm was written for this study. The cosine measure was used to estimate the similarity  $S$  of documents  $x$  and  $y$ :

$$S_{xy} = \frac{\sum X_i Y_i}{\sqrt{\sum X_i^2 \times \sum Y_i^2}}$$

where  $Y_i$  is the weight of feature  $i$  in document  $y$  and  $X_i$  is the weight of feature  $i$  in document  $x$ . A simple word-based text representation with stemming of plurals was used, features that had some correlation with the class of interest were selected using the  $\chi^2$  measure.  $k$  was set to five following optimization trials over a range of  $k$  values.

#### Probabilistic Indexer (P)

Regression techniques find a function that describes the distribution of points in  $n$ -dimensional vector space that belong to a particular class. The function may then be used to predict class membership for new examples. This categorizer uses regression to find a function that

describes the distribution of documents in a space of features that have been selected and weighted using complex probabilistic strategies. The implementation used in this study was programmed at Carnegie Mellon and made available in the Rainbow package but the algorithm used was based on the work of Fuhr’s group at Darmstadt [16].

#### RIPPER (R)

RIPPER is an incremental rule learner designed to be efficient for text [17]; it can handle much larger feature sets than standard rule learners such as C4.5 rules. It finds one good rule that fits some of the positive examples, removes those positive examples from the training set and starts afresh to build another single good rule. The resulting rule set is a disjunction of conjunctions. RIPPER was programmed at AT&T Research.

Rules are very adaptable. Rule learners can be easily altered to exploit the structural aspects of language, for example using phrases as features [18] and exploiting information resources such as thesaurus knowledge [19]. Rules differ from algorithms such as Naive Bayes and  $k$  Nearest Neighbour in another important way. Rule learners carefully select a few important features to combine in each rule. The other algorithms base their predictions on a function calculated over a larger number of features. Valid arguments could be put forward to support either approach. However when it comes to combining systems the point of interest is that the approaches are *different*. As our aim is to maximize the difference between categorizers it is necessary to have at least one rule learner available.

### 3.2. Evaluation

Recall, precision, error rates and  $F_1$  are all in common use to assess how closely the indexing produced by the binary learning algorithm corresponds to that produced by human indexers [20].

The most fundamental measures are recall (the proportion of potential correct outcomes that are actually achieved) and precision (the proportion of the actual outcomes that are correct). In practice these trade off against each other. High recall can only be achieved at the expense of low precision and *vice versa*. The measures are listed below:  $a$  is the number of true positives,  $b$  the false negatives,  $c$  the false positives and  $d$  the true negatives. Where a ‘positive’ is an instance that has been assigned the class by the classifier, a ‘negative’ has been assigned ‘not-class’ and ‘true’ results are instances which have been classified correctly.

$$\text{recall} = \frac{a}{a+b} \quad \text{precision} = \frac{a}{a+c}$$

$$F_1 = \frac{2 \times \text{recall} \times \text{precision}}{\text{recall} + \text{precision}}$$

$$\text{error rate} = \frac{(b+c)}{(a+b+c+d)}$$

This study uses error rate because it is equivalent to the probability of failure  $\theta$ . Therefore, we can compare

predicted error rates to those actually achieved. As our aim is to examine the discrepancies between predictions and results, this suits our purpose well.

### 3.3. Datasets

The algorithms above were run on two datasets: the Reuters-21578 collection of newswires and a subset of Weldasearch, a bibliographic database. Reuters-21578 Distribution 1.0, hereafter referred to as 'Reuters', is a collection of newswire articles made available for research use by Reuters Ltd and Carnegie Group, Inc. (<http://www.research.att.com/~lewis/reuters21578.html>). Reuters has become a standard for text categorization research. Weldasearch is a commercial bibliographic database, produced by TWI<sup>4</sup> that indexes the literature on materials joining technology.

For each dataset the ten most highly populated classes were identified and studied in detail. We chose to look at individual classes because the probabilistic framework makes predictions on a class-by-class basis with each class having a different difficulty distribution. Full details of how the datasets were constructed are given in Appendix A.

## 4. FIVE TYPICAL CATEGORIZERS

In the first experiment, the results of the five categorizers were combined in their standard form in simple voting systems. We wished to see whether this could improve the categorization accuracy. The five categorizers can be combined in a total of ten three-vote systems and one five-vote system (four-vote systems were ignored for simplicity). The three-vote systems will be referred to by acronyms, using the assigned letter code, for example BKP is the system that combines Naive Bayes, k Nearest Neighbour and the Probabilistic Indexer. The five-vote system is simply called 'Five'.

For each of the datasets the ten classes studied were categorized using each possible combination. These 'actual' error rates were compared with 'predicted' error calculated from the error rates of the individual categorizers assuming that errors were independent. Tables 1 and 2 present the best results for each of the classes. Where two combinations tie both results are given.

It is immediately clear that the actual error rates produced by the voting systems for individual classes are consistently worse than the error rates predicted by the independent errors model. Additionally, if the errors were independent we would expect the Five categorizer system to be substantially better than the three component systems. In reality, Five is only one of the best systems in four out of 20 cases. Indeed in eight out of 20 cases a single categorizer is either the best system overall or is as good as the best combined system.

Perfect independence may not be critical. Pragmatic system builders will be more inclined to ask whether

TABLE 1. Best systems for the Reuters data.

Class	Best system/s	Actual error rate	Predicted error rate
Earn	BKR	0.020	0.002
Acq	Five	0.025	0.006
Money-fx	P	0.032	—
Grain	BKR	0.014	0.002
Crude	BIK	0.020	0.007
Trade	Five, BIK	0.018	0.000, 0.003
Interest	K, KPR, BKR	0.025	—, 0.004, 0.004
Ship	P	0.016	—
Wheat	BIR, IPR	0.000	0.000, 0.002
Corn	R	0.000	—

TABLE 2. Best systems for the Weldasearch data.

Class	Best system/s	Actual error rate	Predicted error rate
Steels	R	0.139	—
Mechanical props	BKR	0.167	0.102
Process conditions	Five	0.221	0.153
Arc welding	Five, KPR	0.101	0.033, 0.053
Welded joints	BKR	0.177	0.115
Microstructure	BKR	0.127	0.064
Composition	B	0.133	—
Strength	BKR	0.115	0.055
Defects	R	0.129	—
Patents	R	0.029	—

performance can be improved sufficiently to justify the extra effort required to run several categorizers instead of just one. The general impression given by the data is that the voting systems tend to perform about as well as the best component system of the group. This was tested by performing paired T tests, comparing each voting system with the component with the lowest mean error rate. This test establishes whether the two sets of performance results could have come from the same population; in this case whether the combined system is more effective than its best component (the 'best' component was taken to be the one with the lowest microaveraged error rate). Strictly speaking, the T test is only valid for a population with a normal distribution. However in reality it is robust to deviations from normality and may be used for information retrieval evaluations [21].

The T tests show that for the Weldasearch dataset, none of the combined systems improves on the best component. For the Reuters dataset, four combinations improve on the best component (significance greater than 5%). These are Five, KPR, BKR and BPR (see Table 3). The improvement in performance tends to be in the region of 10–20% of the mean error rate of the best component.

<sup>4</sup>TWI Ltd (The Welding Institute), Granta Park, Great Abington, Cambridge, CB1 6AL, UK.

**TABLE 3.** Mean error rates of each voting system, combinations that improve on their best component are in bold.

Categorizer	Reuters mean $F_1$	Weldasearch mean $F_1$
NBayes B	0.027 ± 0.003	0.160 ± 0.017
IBM I	0.182 ± 0.049	0.300 ± 0.026
kNN K	0.027 ± 0.004	0.185 ± 0.017
Prind P	0.041 ± 0.013	0.182 ± 0.022
Ripper R	0.036 ± 0.009	0.154 ± 0.020
Five	<b>0.021 ± 0.003</b>	0.145 ± 0.016
BIK	0.024 ± 0.003	0.164 ± 0.017
BIP	0.028 ± 0.006	0.167 ± 0.019
BIR	0.031 ± 0.007	0.146 ± 0.018
IKP	0.028 ± 0.006	0.173 ± 0.019
IKR	0.033 ± 0.007	0.157 ± 0.018
IPR	0.035 ± 0.008	0.158 ± 0.019
KPR	<b>0.023 ± 0.003</b>	0.143 ± 0.016
BKP	0.024 ± 0.002	0.160 ± 0.017
BKR	<b>0.022 ± 0.004</b>	0.140 ± 0.016
BPR	<b>0.022 ± 0.003</b>	0.153 ± 0.017

#### 4.1. Coincident errors

We have seen that the combined systems tested here do not perform as well as would be predicted using the independent errors model. Does the coincident errors model conform better to the results we have seen? In order to apply the coincident errors model we require an approximation of the difficulty distribution. The best difficulty distribution would be derived from the population itself rather than a sample. In practice, it is not possible to obtain this because we can never test all possible categorizers. A practical alternative is a discrete distribution derived from the largest number of classifiers available. For this experiment, we use the five classifiers that contribute to the five-vote system. The observable  $\theta$  values are 0, 1/5, ..., 5/5 etc. (i.e.  $m$  from Equation (2) is 5). The values of the difficulty distribution are the proportion of records for which 0, 1...5 of the categorizers make an incorrect classification. Difficulty distributions for the Reuters and Weldasearch classes are presented in Tables 4 and 5, respectively.

To get a prediction of the error rate using the independent errors model, the integral<sup>5</sup> of Equation (2) is calculated. For example for Earn with three categorizers in the voting systems the predicted error rate is

$$0.066(0.2^3 + 3(0.2^2 \cdot 0.8)) + 0.027(0.4^3 + 3(0.4^2 \cdot 0.6)) \dots$$

It is worth noting that all the categories have non-zero values for  $\theta = 0.6$ . We therefore expect that adding more categorizers to the voting systems will eventually cause performance to degrade rather than improve. Furthermore, many of the difficulty distributions have non-zero values for

$\theta = 0.8$  and  $\theta = 1.0$ . These represent groups of records for which adding more systems can never improve performance.

Performance was predicted for majority voting systems with 3–11 component categorizers using both the independent errors model and the coincident errors model. The results for a selection of classes are presented in Figure 2. These predictions assume that all component categorizers give the mean error rate. Therefore, they cannot be compared directly to the predictions and actual results in the previous section. However, they do demonstrate what could be expected, given the level of coincident errors that we have here, if even more effort were exerted to combine the results from more categorizers.

For all the classes the independent errors model predicts a steady improvement in performance as more categorizers are added to the voting system, with the error rate tending towards zero. Some classes, e.g. Wheat, actually hit zero errors with less than 11 systems. The coincident errors model gives a very different picture. At best, performance improves for a while then reaches a limit and starts to flatten off; classes such as Earn show this behaviour. At worst, adding more categorizers to the system produces a small but steady decline in performance. The Steels class is an example of this. The coincident errors model thus indicates that while combining systems in this way is not particularly damaging it does not reward the extra computation required to produce large numbers of categorizers.

The predictions fit well with the experimental results in the previous section. The independent errors model would have predicted that the best performance should always have been obtained with the five-vote system. In practice, the system that gave the best performance was often a three-vote system or even a single categorizer. The coincident errors model predicts limited performance improvement of this kind with error rate plots tending to a minimum and then flattening off, so that nothing is gained by adding more categorizers to the system.

The coincident errors model predictions also fit with the apparent intractability of the Weldasearch data. The plots of the coincident errors model for the ten Weldasearch classes are all very flat, suggesting that combining systems will not improve performance and, in practice, none of the combined systems was a significant improvement on its best component.

It seems that the systems selected here, which are reasonably typical of categorizers, are not sufficiently different to make really effective combined systems. Some improvement can be seen for the Reuters data, but there is no indication that working harder by adding more than five systems is going to improve performance any further. For the Weldasearch data there is no significant improvement. For this data, the levels of coincident errors appear too high for combining systems to produce any gains. High levels of coincident errors may indicate that there are underlying similarities between the predictions made by the categorizers.

<sup>5</sup>In fact, a discrete summation.

**TABLE 4.** Difficulty distributions for the Reuters classes.

Class	$\theta$					
	0.0	0.2	0.4	0.6	0.8	1.0
Earn	0.888	0.066	0.027	0.012	0.007	0.000
Acq	0.856	0.098	0.030	0.012	0.004	0.000
Money	0.921	0.031	0.019	0.012	0.011	0.005
Grain	0.951	0.028	0.011	0.008	0.001	0.002
Crude	0.948	0.025	0.008	0.003	0.004	0.011
Trade	0.971	0.007	0.007	0.003	0.003	0.009
Interest	0.958	0.009	0.013	0.011	0.006	0.004
Ship	0.933	0.046	0.004	0.003	0.010	0.004
Wheat	0.984	0.011	0.004	0.001	0.000	0.000
Corn	0.965	0.024	0.007	0.003	0.001	0.000

**TABLE 5.** Difficulty distributions for the Weldasearch classes.

Class	$\theta$					
	0.0	0.2	0.4	0.6	0.8	1.0
Steels	0.776	0.053	0.048	0.056	0.043	0.023
Mechanical props	0.755	0.054	0.058	0.058	0.052	0.022
Process conditions	0.706	0.062	0.068	0.060	0.070	0.034
Arc welding	0.834	0.052	0.043	0.037	0.021	0.013
Welded joints	0.768	0.044	0.051	0.047	0.054	0.035
Microstructure	0.811	0.042	0.048	0.047	0.032	0.020
Composition	0.837	0.032	0.031	0.036	0.040	0.024
Strength	0.819	0.047	0.047	0.043	0.026	0.018
Defects	0.802	0.053	0.028	0.041	0.049	0.028
Patents	0.934	0.027	0.010	0.010	0.015	0.004

## 5. TWO LESS ORTHODOX CATEGORIZERS

The first experiment has given indications that ‘difference’ of the kind needed to produce independent errors cannot be produced simply by using a variety of machine learning algorithms; there is some other process that needs to be considered to make a ‘difference’ workable. We hypothesize that this ‘difference’ must be created by stepping away from the underlying formal information processing mechanisms that constitute the usual range of categorizing/pattern recognition algorithms. In the second experiment, we explored this hypothesis by developing two new categorizers each of which ‘break the rules’ used to develop typical categorizers in some way.

We used RIPPER as the platform on which to build these unorthodox categorizers. RIPPER was one of the more successful standard categorizers, it featured in all of the successful combined systems for the Reuters data. Furthermore it was designed with intrinsic flexibility, which made it easy to adapt.

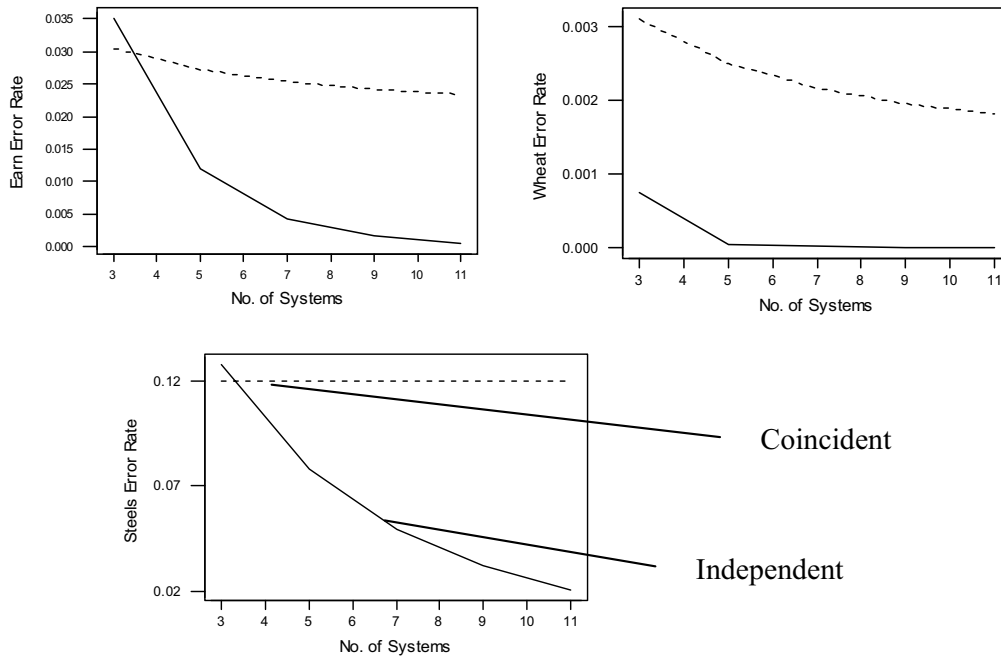
The test bed for these experiments was the Weldasearch database. None of the combined systems involving only the five standard categorizers could improve on their best component for this data. Therefore Weldasearch presented itself as a hard case.

An additional point of experimental design was to decide how many categorizers to include in each voting system. The predictions of the coincident errors model strongly suggest that unless the new categorizers are radically different most of the payback will come from combining three categorizers. Combining five is significantly more effort for very marginal gain. Therefore this experiment concentrates on three-vote systems.

### 5.1. Introducing difference in text representation

The standard categorization approach represents text as words, because experience suggests that they are the best available representation. Phrases, which would be expected to have more semantic content than isolated words, have been shown to perform less well than words, probably because they occur comparatively infrequently giving them poor statistical properties [22]. Since almost all categorizers use a word-based text representation it could be a source of similarity between them.

Changing the text representation is a candidate method for increasing the ‘difference’ between the systems that are to be combined. Research published by Scott and Matwin suggests that taking votes from versions of RIPPER that use different types of phrases with versions of RIPPER



**FIGURE 2.** Selected plots for classes Earn, Wheat and Steels, comparing the predictions of the independent (solid) and coincident (dashed) models for combinations of between three and eleven systems. Note, the  $x$ -axis varies.

using words can sometimes produce improvements over the best single classifier [7]. We take a different approach by representing text as  $n$ -grams: short strings of characters that ignore word boundaries. This is counter-intuitive—why would we want to reduce the amount of information available to a categorizer from word units, which have a recognizable semantic content, to  $n$ -grams which have none? To get improvements with combined systems we do not necessarily need to combine systems which are the best available. Rather, we need to combine systems which are good enough, ultimately systems with error rates of less than 0.5 and which are ‘different’ from each other. Combining a categorizer that uses phrases with one that uses words may work because more information has been made available. This introduces an additional factor which could confuse the interpretation of the results. Combining word-based categorizers with a categorizer that uses less information and is therefore weaker is a cleaner experiment; we can have more confidence that any improvement has come from the *difference* in the text representation rather than its *nature*.

In this experiment equipfrequent  $n$ -grams were deployed.  $n$ -grams are strings of consecutive letters which may be confined within words or bridge across words incorporating spaces. Bi-grams contain two characters, tri-grams three and so forth. They are produced by shifting an  $n$  character window across the text one letter at a time. In this way the phrase

*nests in my beard*

would produce the following set of bi-grams, where the

symbol ‘\*’ represents a space:

*ne, es, st, ts, s\*, \*i, in, n\*, \*m, my, y\*, \*b, be, ea, ar, rd.*

Taken over a whole text collection, this produces a more economical representation than looking at words since there are a fixed number of  $n$ -grams of a particular length; for example, the bi-gram *st* occurs in many more places than the word nests.

Equipfrequent  $n$ -grams are fragments of strings of different lengths which occur with approximately equal frequency in the text and produce attribute vectors which are as small and as densely populated as possible. For this work, a program was written to select equipfrequent  $n$ -grams using Lynch’s method [23].

$n$ -grams are a well-established tool in textual information systems. Their uses include text compression, spelling error detection, query expansion, personal name matching and processing of non-English texts. A review of this work is provided by Robertson and Willett [24].  $n$ -gram-based systems have not been a dominant line of enquiry in text categorization, because the use of this format reduces retrieval effectiveness [25]. However,  $n$ -grams are flexible. They can make sense of text containing spelling errors and are well suited to languages that are not amenable to the simple stemmers that can be used for English [26]. A number of promising  $n$ -gram-based categorizers have been reported in the literature including: an error tolerant categorizer for OCR documents [27]; Acquaintance, a language-independent system that could estimate the similarity between documents in English and Japanese [28] and was also tested on the TREC-4 problem [29]; and, more



**TABLE 6.** Best systems which include the equipfrequent  $n$ -gram categorizer for the Weldasearch data.

Class	Best system/s	Actual error rate	Predicted error rate
Steels	R	0.139	—
Mechanical props	XRK	0.157	0.096
Process conditions	XKP	0.227	0.184
Arc welding	P	0.118	—
Welded joints	XKB	0.178	0.121
Microstructure	XRK	0.113	0.056
Composition	XPB	0.130	0.059
Strength	XRB	0.112	0.045
Defects	XRB	0.124	0.061
Patents	XRP, XRB	0.015	0.003, 0.003

recently, Telltale, also aimed at OCR text, which claims to offer the searching of a gigabyte of text data on an ordinary PC in 45 s [25].

High dimensionality and the extremely skewed distribution of word frequencies are well-known problems for text categorization. To combat them, feature selection heuristics are commonly used which pick out words with mid-range frequencies. It might be reasonable to expect that equipfrequent  $n$ -grams, which eliminate these two problems, would perform well. In reality this is not quite true. The average performance of RIPPER using this text representation was worse than both standard RIPPER and naive Bayes. It seems that features with good frequency properties are not sufficient on their own to make a good categorizer. However, in theory combining systems with this new 'Xripper' categorizer ought to be profitable because all ten Weldasearch classes have error rates of less than 0.5 (the mean error rate of Xripper is  $0.176 \pm 0.024$ ).

The new Xripper (X) categorizer was combined with every available pair of the original five standard paradigm categorizers giving ten three-vote systems. The results are presented in Table 6.

## 5.2. Introducing 'difference' by using domain knowledge

The second approach to producing an unorthodox categorizer breaks from the machine learning approach altogether. Instead of using the rule learning function of RIPPER to generate rules automatically, rules were written by hand using both information gathered from a thinking-aloud exercise conducted as part of a related study [30] and information in the Weldasearch thesaurus. This is a method that has been rejected by most categorization researchers. The reason normally given is that writing an optimal rule set is very time consuming; writing the rules for the CONSTRUE system is reported to have taken 9.5 person years [31]. However, for this combined system no attempt was made to write the best possible rules. It was not necessary, all that was required was a 'good enough' rule set that had *not been produced by a machine learning system*. Instead a 'first draft' set of

sensible rules was used. Similar rules could be written by any competent indexer with very little training. Nonetheless, these 'semantic rules' were better on average than all the standard categorizers except RIPPER (the mean error rate of semantic rules is  $0.155 \pm 0.027$ ).

The semantic rules differ from RIPPER rules because the rules RIPPER makes are based on a probabilistic association between words and index terms, so that a typical RIPPER rule reads

*steels IF fabrication AND welded AND heat.*

These words are likely to appear in the context of articles on steels, most fabrication is carried out using steels, but they have no direct conceptual link to steels; a document on the fabrication of copper boilers could use exactly the same terms. The RIPPER rules are effective, but they do not necessarily make sense to a human expert. The semantic rules only incorporate words that mean steel in a much more direct way. For example AISI 316 is a commonly used type of stainless-steel, all the indexers who participated in the thinking-aloud experiment made the connection to steels as soon as they spotted it. Therefore we wrote the rule

*steels IF 316.*

Another rule for steels is context related. Most types of steel are magnetic and electric arcs can induce strong magnetic fields which may be a problem in welding. Any discussion of magnetic effects in a welding database is therefore likely to concern steels, resulting in the rule

*steels IF magnetic.*

Similar explanations could be produced for each of the rules we used, some would be complicated but most are very simple. This is what makes these rules semantic—they can be explained in the terms of the domain by tracing the connections between ideas. The rules produced by machine learning systems such as RIPPER can only be explained in terms of the information content of the system derived through training sets.

The semantic rules (S) categorizer was combined with every available pair of the original five standard paradigm categorizers giving ten three-vote systems. The results are presented in Table 7. In addition, semantic rules were combined with Xripper and each of the standard categorizers to give a total of five combined systems. The results for these combinations are presented in Table 8.

## 5.3. Results

Employing equipfrequent  $n$ -grams as the text representation effectively cripples RIPPER. The average error rate of Xripper is  $0.176 \pm 0.024$ , as compared to  $0.154 \pm 0.020$  for the standard version. Yet although it still does not approach the predictions of the independent errors model, we observe that when it is combined in simple voting systems *this crippled version can reduce error rates significantly*. This is

**TABLE 7.** Best systems which include semantic rules for the Weldasearch data.

Class	Best system/s	Actual error rate	Predicted error rate
Steels	SRK	0.132	0.070
Mechanical props	SKB	0.157	0.095
Process conditions	SKP	0.227	0.205
Arc welding	SKP	0.101	0.050
Welded joints	SRB	0.186	0.121
Microstructure	SRB	0.108	0.049
Composition	SPB	0.124	0.059
Strength	SRP	0.104	0.052
Defects	S	0.116	—
Patents	SRP, SRB	0.017	0.003, 0.003

**TABLE 8.** Best systems which include both the equipfrequent  $n$ -gram categorizer and semantic rules.

Class	Best system/s	Actual error rate	Predicted error rate
Steels	XSR	0.134	0.080
Mechanical props	XSP	0.144	0.098
Process conditions	P	0.253	—
Arc welding	XSP	0.113	0.054
Welded joints	B	0.187	—
Microstructure	XSB	0.107	0.049
Composition	B	0.133	—
Strength	XSB	0.104	0.042
Defects	S, XSI	0.116	0.097
Patents	XSB, XSP, XSR	0.019	0.003, 0.003, 0.002

something that the combinations of typical categorization systems tested in the first experiment were not capable of achieving for this dataset.

In total, there are just two classes where a single categorizer is better than any of the combined systems (see Table 6), compared to four when standard categorizers only were combined. Furthermore, paired T tests show that six of the new combined systems perform significantly better than their best component (5% level). These winning combinations are presented in Table 9. They show improvements in mean error rate of 5–15%.

For the first time for the Weldasearch data, combined systems are doing better than standard RIPPER, the best single categorizer available for these classes. We have the evidence to show that introducing less orthodox categorizers can improve the effectiveness of combined systems.

Examination of Table 7 shows that the combinations that have semantic rules as a component also fail to perform as well as the independent errors model would predict. However, a single component is more effective than any of the combined systems for only the one term 'Defects'. It is notable that the single categorizer that outperforms the combined systems in this case is semantic rules itself.

**TABLE 9.** Mean error rates of voting systems that include the equipfrequent  $n$ -gram categorizer, combinations that improve on their best component are in bold.

Categorizer	Mean $F_1$
Xripper X	0.176 ± 0.024
XRK	<b>0.145 ± 0.020</b>
XRP	<b>0.143 ± 0.019</b>
XRБ	<b>0.141 ± 0.019</b>
XRI	0.158 ± 0.022
XKP	<b>0.147 ± 0.016</b>
XKB	<b>0.144 ± 0.016</b>
XKI	0.165 ± 0.019
XPB	<b>0.150 ± 0.017</b>
XPI	0.165 ± 0.021
XBI	0.153 ± 0.018

**TABLE 10.** Mean error rates of voting systems that include the semantic rules categorizer, combinations that improve on their best component are in bold.

Categorizer	Mean $F_1$
SRules S	0.155 ± 0.027
SRK	<b>0.138 ± 0.020</b>
SRP	<b>0.136 ± 0.020</b>
SRB	<b>0.135 ± 0.020</b>
SRI	0.156 ± 0.025
SKP	0.143 ± 0.017
SKB	0.140 ± 0.017
SKI	0.160 ± 0.022
SPB	0.148 ± 0.017
SPI	0.160 ± 0.022
SBI	0.149 ± 0.022

Paired T tests confirm that three of the three-vote systems that include semantic rules are significantly better than their best single component (see Table 10), compared with six for the systems that include Xripper (Table 9). On the surface this looks as though semantic rules are less effective in combined systems than Xripper. Closer examination shows that where the semantic rules do produce improvements, the actual improvement in performance is greater than similar winning systems involving Xripper. Whereas Xripper produced improvements in the mean error rate with a maximum of 8.5% when RIPPER was the best component, the improvements for semantic rules with RIPPER as the best component are between 10 and 12%. This suggests that involving an element of domain knowledge in the system produces extra gains in effectiveness, as might be expected. This result is also coherent with our 'difference' hypothesis.

As for three-vote systems that contain both the new categorizers, the results presented in Table 8 show that the number of classes which give the best performance with a single categorizer has increased to four (from two for three-vote systems with Xripper, and just one for three-vote

**TABLE 11.** Mean error rates of voting systems that include both the Xripper and semantic rules categorizers, the combination that improves on its best component is in bold.

Categorizer	Mean $F_1$
XSB	$0.1389 + 0.020$
XSI	$0.158 \pm 0.025$
XSK	$0.144 \pm 0.021$
XSP	$0.140 \pm 0.020$
XSR	<b><math>0.141 \pm 0.021</math></b>

systems with semantic rules). Furthermore, only one system, that which combines semantic rules and Xripper with standard RIPPER, performs significantly better than its best component (see Table 11). This system has a mean error rate which is comparable to the best rate produced by any other voting system that uses Xripper (XRB), but it is not as good as the best three-vote system that uses semantic rules (SRB).

A limit appears to have been reached. Since performance is similar to the best system that uses Xripper and two standard categorizers, this may be a limit on the amount of ‘difference’ that can be achieved when combining an equifrequent  $n$ -gram representation with other systems. Alternatively, it may be a limit on the amount of difference that can be expected between rule-based systems, given that both semantic rules and Xripper use the RIPPER rules format. Xripper also uses RIPPER’s learning functions, albeit on a different text representation. This might be reasonably expected to introduce similarities in behaviour.

The experiments with Xripper and semantic rules in voting systems indicate that combining categorizers that use word-based text representations and machine learning algorithms with less orthodox ones can improve accuracy.

## 6. DISCUSSION AND CONCLUSIONS

We have seen that the actual accuracy achieved by combining text categorizers is considerably less than the accuracy that would be predicted if the errors produced by the systems were independent of each other. Their performance conforms more closely to the coincident errors model which assumes that some records are more likely to cause errors than others. This suggests that there may be fundamental similarities in the ways that different categorizers operate. We propose that this similarity can be traced to the underlying ‘information’ processing upon which the algorithms depend.

Categorization systems have limits on their performance that may be described by information theory [32]. They search for features whose values will help predict the categories of interest; in information theoretic terms, these minimize the conditional entropy. Some algorithms do a better job of distinguishing which features reduce the conditional entropy the most and, consequently, are more accurate. Obviously, not all algorithms use the entropy

measure to rate their success, but, ultimately, they can do one of two things: they either learn a function for a set of features or they select indicative features [15]. If the features available to the systems are similar, then, almost by definition, the machine learning algorithms that do best will be doing similar things and their performance will be limited by the distribution of the features across the classes of interest (the same distribution that would be used to calculate conditional entropy). Records in which the information given is misleading or inadequate will tend to cause errors whichever categorizer is used, thus producing an inevitable source of coincident errors when they are combined.

The best text categorizers tend to use the same kind of word-based text representation. Could coincident errors be reduced if a categorizer using a different kind of information as input was included in the combined system? We examined this possibility by developing a categorizer that used an unusual equifrequent  $n$ -gram text representation, Xripper. Although this categorizer was less accurate than its word-based equivalent, it did produce improvements in accuracy when included in a voting system. Apparently an  $n$ -gram representation can reduce the number of coincident errors when combined with word-based systems, despite the fact that the individual categorizer is weaker than its word-based equivalent.

The second unorthodox categorizer abandoned machine learning in favour of a knowledge based approach. This opened up the categorizer to new information sources. The knowledge encoded in the texts was supplemented by knowledge elicited from human indexers. The semantic rules produced were slightly less accurate than the equivalent rules produced by RIPPER. Once again, the less orthodox system produced improved accuracy when combined with other categorizers. The expertise required to write a rule set by hand is not available for every problem; few academic researchers have the privilege of working with domain experts on categorization problems. However, when that expertise *is* available it would certainly be unwise to ignore it. Combining systems presents an opportunity for getting the most out of rapidly developed knowledge-based systems by combining them with machine learning systems.

The orthodox approach to building a text categorizer using machine learning and a word-based text representation is highly effective when a single accurate categorizer is required. The problem is not that this approach is flawed. The problem is that machine learning algorithms all achieve their success in the same way: by learning a function using the correlation between features in the text and the subject. This means that, when the features remain unchanged, there is little to be gained from combining these systems because the errors they make tend to be the same. A better approach is to combine categorizers with others that break from the orthodoxy of stemmed words and machine learning, even if they are individually *weaker*.

It appears that, when combining text categorizers, success comes from selectively breaking the rules that have been developed to produce optimal individual categorizers and instead producing a collection of suboptimal categorizers

with different biases. This opens up an opportunity for a new generation of categorizers that employ imaginative methods. The combined systems will then provide a platform on which high-risk categorizer designs can exploit each others' strengths.

Finally, based on the results presented here, we can form some intuitions about what kinds of 'differences' between categorizers are likely to minimize coincident errors. Based on our results, combining categorizers that all use a word-based text representation and machine learning algorithms is not particularly profitable. A successful combination might include a mixture of systems that weight features, methods such as naive Bayes and k Nearest Neighbour and methods that select features. Ideally, the text representation used by these two systems should be different. The third component should exploit domain knowledge. The use of knowledge elicited from domain experts is the best available alternative to the function learning approach adopted by the current generation of categorizers. The work presented here demonstrates that the two approaches are not antagonistic, but can be successfully combined.

#### ACKNOWLEDGEMENTS

We thank TWI Ltd for access to the Weldasearch data and for the invaluable assistance of their indexers, IBM for supplying a trial copy of the Intelligent Miner package and Max Bramer and an anonymous reviewer for their comments on the experiments. This research was conducted while the first author was at Portsmouth University.

#### REFERENCES

- [1] Quinlan, J. R. (1996) Bagging, Boosting and C4.5. In *Proc. 13th Int. Conf. on Artificial Intelligence (AAAI96)*, Portland, OR, 4–8 August, pp. 725–730. AAAI Press/The MIT Press, Cambridge, MA.
- [2] Opitz, D. and Maclin, R. (1999) Popular ensemble methods: an empirical study. *J. Artificial Intell. Res.*, **11**, 169–198.
- [3] Breiman, L. (1996) Bagging predictors. *Machine Learning*, **24**, 123–140.
- [4] Hull, D. A., Pedersen, J. O. and Schützte, H. (1996) Method combination for document filtering. In *SIGIR 96. Proc. 19th Annual Int. ACM SIGIR Conf. on Research and Development in Information Retrieval*, Zurich, Switzerland, 18–22 August, pp. 279–287. Hartung-Gorre, Konstanz.
- [5] Larkey, L. S. and Croft, W. B. (1996) Combining classifiers in text categorization. In *SIGIR 96. Proc. 19th Annual Int. ACM SIGIR Conf. on Research and Development in Information Retrieval*, Zurich, Switzerland, 18–22 August, pp. 289–297. Hartung-Gorre, Konstanz.
- [6] Li, Y. H. and Jain, A. K. (1998) Classification of text documents. *Comp. J.*, **41**(8), 537–546.
- [7] Scott, S. and Matwin, S. (1999) Feature engineering for text classification. In *Machine Learning, Proc. 16th Int. Conf. (ICML'99)*, Bled, Slovenia, 27–30 June, pp. 379–388. Morgan Kaufmann, San Francisco, CA.
- [8] Craven, M., DiPasquo, D., Freitag, D., McCallum, A., Mitchell, T., Nigam, K. and Slattery, S. (2000) Learning to construct knowledge bases from the World Wide Web. *Artificial Intell.*, **118**, 69–113.
- [9] Sebastiani, F. (1999) A tutorial on automated text categorisation. In *Proc. ASAI-99, 1st Argentinian Symposium on Artificial Intell.*, Buenos Aires, 7–10 September, pp. 7–35.
- [10] Wiebe, J., Hirst, G. and Horton, D. (1996) Language use in context. *Commun. ACM*, **39**(1), 102–111.
- [11] Tumer, K. and Ghosh, J. (1999) Linear and order statistics combiners for pattern classification. *Combining Artificial Neural Networks*, pp. 127–162. Springer, New York.
- [12] Eckhardt, D. E. and Lee, L. D. (1985) A theoretical basis for the analysis of multiversion software subject to coincident errors. *IEEE Trans. Software Eng.*, **11**, 1511–1517.
- [13] Lewis, D. D. (1995) Evaluating and optimizing autonomous text classification systems. In *Proc. 18th Annual Int. ACM SIGIR Conf. on Research and Development in Information Retrieval*, Seattle, WA, 9–13 July, pp. 246–254. ACM Press, New York.
- [14] Hansen, L. K. and Salamon, P. (1990) Neural network ensembles. *IEEE Trans. Pattern Analysis Machine Intell.*, **12**(9), 993–1001.
- [15] Mitchell, T. M. (1997) *Machine Learning*. McGraw-Hill, New York.
- [16] Fuhr, N. and Buckley, C. (1991) A probabilistic learning approach for document indexing. *ACM Trans. Inform. Syst.*, **9**(3), 223–248.
- [17] Cohen, W. W. and Singer, Y. (1996) Context-sensitive learning methods for text categorization. In *Proc. 19th Annual Int. ACM-SIGIR Conf. on Research and Development in Information Retrieval*, Zurich, Switzerland, 18–22 August, pp. 307–316. Hartung-Gorre, Konstanz.
- [18] Finch, S. (1995) Partial orders for document representation: a new methodology for combining document features. In *Proc. 18th Annual Int. ACM SIGIR Conf. on Research and Development in Information Retrieval*, Seattle, WA, 9–13 July, pp. 264–72. ACM Press, New York.
- [19] Junker, M. and Abecker, A. (1997) Exploiting thesaurus knowledge in rule induction for text classification. In *Proc. RANLP-97 2nd Int. Conf. on Recent Advances in NLP*, Tzigrav Chark, Bulgaria, 11–13 September, pp. 202–207.
- [20] Yang, Y. (1999) An evaluation of statistical approaches to text categorization. *Inform. Retrieval*, **1**(1), 69–90.
- [21] Hull, D. (1993) Using statistical testing in the evaluation of retrieval experiments. In *SIGIR 93. Proc. 16th Annual Int. ACM SIGIR Conf. on Research and Development in Information Retrieval*, Pittsburgh, PA, 27 June–1 July, pp. 329–338. ACM, New York.
- [22] Lewis, D. D. (1992) An evaluation of phrasal and clustered representations on a text categorization task. In *Proc. 15th Annual Int. ACM SIGIR Conf. on Research and Development in Information Retrieval*, Copenhagen, Denmark, 21–24 June, pp. 37–50. ACM, New York.
- [23] Lynch, M. F., Petrie, J. H. and Snell, M. J. (1973) Analysis of the microstructure of titles in the Inspec data-base. *Inform. Storage Retrieval*, **9**, 331–337.
- [24] Robertson, A. M. and Willett, P. (1998) Applications of N-grams in textual information systems. *J. Documentation*, **54**(1), 48–69.
- [25] Miller, E., Shen, D., Liu, J. and Nicholas, C. (2000) Performance and scalability of a large-scale N-gram based information retrieval system. *J. Digital Inform.*, **1**(5) <http://jodi.ecs.soton.ac.uk/>

- [26] Ekmekcioglu, F. C., Lynch M. F., Robertson, A. M., Sembok, T. M. T. and Willett, P. (1996) Comparison of n-gram matching and stemming for term conflation in English, Malay and Turkish texts. *Text Technol.*, **6**(1), pp. 1–14.
- [27] Cavnar, W. B. and Trenkle, J. M. (1994) N-gram-based text categorization. In *Proc. 3rd Annual Symp. on Document Analysis and Information Retrieval (SDAIR-94)*, Las Vegas, NV, 11–13 April, pp. 161–175. UNLV Publications/Reprographics, Las Vegas, NV.
- [28] Damashek, M. (1995) Gauging similarity with n-grams: language-independent categorization of text. *Science*, **267**(10), pp. 843–848.
- [29] Huffman, S. (1996) Acquaintance: language-independent document categorization by N-grams. *The Fourth Text Retrieval Conference (TREC-4)*, Gaithersburg, Maryland, 1–3 November, pp. 359–371. NIST, Gaithersburg, MD.
- [30] Uren, V. S. (2000) An evaluation of text categorisation errors. In *Proc. Workshop on the Evaluation of Information Management Systems*, London, 15 September, pp. 79–87. Queen Mary and Westfield College, London.
- [31] Lewis, D. D. (1992) Feature selection and feature extraction for text categorization. In *Proc. Speech and Natural Language Workshop*, February, pp. 212–217. DARPA/Morgan Kaufmann, San Mateo.
- [32] Shannon, C. E. and Weaver W. (1949) *The Mathematical Theory of Communication*, University of Illinois Press, Urbana, IL.
- [33] Apte, C., Damerau, F. and Weiss, S. M. (1994) Automated learning of decision rules for text categorisation. *ACM Trans. Inform. Syst.*, **12**(3), 233–251.

## APPENDIX A. CONSTRUCTION OF THE DATASETS

This research was concerned with the core factors affecting performance and not with building a state of the art categorization system. Consequently a decision was made to sacrifice direct comparability with other systems in favour of using a somewhat smaller dataset, that would produce acceptable results while minimizing the time required for experiments. This is a continuation of our strategy of conducting minimal input explorative experiments. However, no published work on the minimum dataset size needed for text categorization is available. It was therefore necessary to establish empirically the number of records required to establish a reasonable performance.

### A.1. Reuters-21578

The Reuters-21578 Distribution 1.0 (Reuters) is a collection of newswires. Most researchers use the Modified Apte (ModApte) split based on that used by Apte and co-workers [33]. This uses 9603 documents in the training set and 3299 documents in the test set and omits all documents that have no topic set. Since we had chosen to only study the ten most populated classes we considered it necessary to reduce the size of the training set as such an abundance of data is unrealistic in practice.

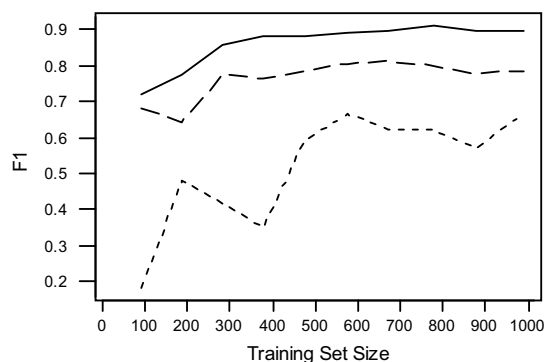


FIGURE A.1.  $F_1$  versus training set size for acq (solid), crude (dashed) and ship (dotted).

To this end, trials were run to determine the effect of gradually increasing the training set size. Distributed selections of test documents were made from the ModApte split maintaining the test set size at 559 documents and gradually increasing the training set size from approximately 100 to approximately 1000. The documents included in the training and test sets were sampled evenly from the main file rather than taking them all from the beginning of the file. This procedure was adopted to avoid temporal problems in which a particular class is represented mainly by wires about one news story. The naive Bayes algorithm was trained and tested using this data.

As Figure A.1 illustrates, when there are few training examples for a particular class, as for ‘ship’ in the early iterations,  $F_1$  is low and erratic. However once there are 10 examples of the class ship, which occurs with approximately 500 documents,  $F_1$  changes relatively little as more documents are added to the training data. By the time there are about 1000 training records in total the performances of these classes have all stabilized and adding further data does not substantially improve performance. A decision was therefore made to work with a sample of the ModApte split consisting of 988 training documents and 559 test documents. With this sample, none of the ten classes chosen for study had fewer than 15 records in the training set.

### A.2. Weldasearch

Bibliographic databases provide an invaluable resource for text categorization research because they contain many examples of text which have already been given subjects. For these experiments, a specialist engineering database was used. Weldasearch is a commercial bibliographic database on materials joining technology (welding, brazing, solid-state bonding, etc.). Typical content for a Weldasearch record is illustrated in Figure A.2.

This database has not previously been used in categorization experiments. However we consider it an acceptable experimental platform. First, there is the issue of quality. Weldasearch is produced by a relatively small team of indexers who deal with a relatively small range of topics.

ID: 158937
TITLE: Application of modified 9% chromium steels in power generation components
BODY: The effect of welding and cold bending on the long term properties of modified 9%Cr steel (P91/T91: approx. 0.1%C, 0.4%Mn, 9%Cr, 1%Mo, 0.2%V, 0.08%Nb) is reported. The behaviour of the parent material, weldments (produced by TIG, MMA and submerged arc welding), and short radius bends are investigated, including microstructural characteristics, mechanical properties, and creep strength (under constant stress). Comparison is made with the corresponding properties in the 12%Cr steel X20CrMoV121 (0.18%C, 0.52%Mn, 11.9%Cr, 0.92%Mo, 0.26%V). Advice is given on suitable design stress, and how to account for the reduced creep rupture strength resulting from welding and cold working. Material cost savings are reported for a case when T91/P91 material was used as a replacement for X20 material.
TOPICS: Reference lists; Bending; Submerged arc welding; Steels; Strength; Stress rupture strength; Tubes and pipes; Weld zone; Welded joints; Creep resisting materials; Gas shielded arc welding; GTA welding; Heat affected zone; High alloy steels; Arc welding; Mechanical properties; Metal working; Microstructure; MMA welding

FIGURE A.2. Example of typical content of a Weldasearch record.

All the abstracts are written to a house style and abstracts and index terms are edited so that all entries to the database are checked at least once. These factors produce a dataset that can be expected to be consistent. Second, it was possible to communicate with the indexers themselves and discover what processes they used to decide on class assignments. This made it possible to take the unusual route of building knowledge-based rule sets.

A subset of the database from record 143,000 to record 145,000 was used in most of the experiments. Even-numbered records were used for the training set and odd numbered records for the test set. These classes are more densely populated than those for the Reuters data. Given the results of the Reuters trials there is every reason to believe that this is a sufficiently large sample to categorize these classes.