Managing content with automatic document classification

Rafael A. Calvo(1), Jae-Moon Lee (1,2) and Xiaobo Li (1)
(1) School of Electrical and Information Engineering -
University of Sydney, Australia -
(2) School of Computer Engineering - Hansung University, Korea
http://www.weg.ee.usyd.edu.au
rafa@ee.usyd.edu.au, jmlee@hansung.ac.kr, xiaobo@ee.usyd.edu.au

Abstract

News articles and web directories represent some of the most popular and commonly accessed content on the web. Information designers normally define categories that model these knowledge domains (i.e. news topics or web categories) and domain experts assign documents to these categories. This paper describes how machine learning and automatic document classification techniques can be used for managing large numbers of news articles, or web page descriptions, lightening the load on domain experts. In this paper we use two datasets, one with with more than 800,000 Reuters news stories and another with over 41,000 web sites, and classify them using a Naive Bayes algorithm, into predefined categories. We discuss the different parameters and design decisions that normally appear when building automatic classifiers, including, stemming, stopwords, thresholding, amount of data and approaches for improving performance using the structure in XML documents. The methodology developed would enable web based applications or workflow systems to manage information more efficiently, i.e. by assigning documents to topics automatically or assisting humans in the process of doing so.

1 Introduction

Information overload (in which individuals are faced with an oversupply of content) could become the road rage in this new millennium. In order for this content to become useful information that can empower users, rather than frustrate or confuse them, we need novel ways of delivering only what is needed, at the right time and in the right format. News stories are the most important source of up-to-date information about the world around us, and represent some of the most often updated, and highest quality content on the web. Web page descriptions and directories are more stable than news stories, but the amount of pages needed to be indexed grows much faster than human indexers can process. Therefore, it is most important to develop ways to process news, web page descriptions and other forms of text efficiently. In this paper we have studied machine learning models, and applied them to the automatic classification of the two types of content.

News story articles are written by reporters from all over the world. These reporters often work for news agencies such as the Associated Press and Reuters. These agencies collect the news, edit them and sell bundles of articles to the periodicals accessed by web users (e.g. news.yahoo.com, Sydney Morning Herald, etc). It is important for both the agencies and the periodicals to have an organized well-managed stream of news. News is normally classified according to taxonomies that are relevant to readers, (e.g. politics, Iraq or Oil). This classification can be very difficult because it requires human expertise to spot relationships between the taxonomy and the documents. Even the experts do not agree on what should go where and inter-indexer consistency in classification shows considerable variation [7].

The web is also a great resource for all types of information, but it needs to be organized in order to be useful. Hundreds of information designers and domain experts have built and maintain directories such as Yahoo!, and thousands of volunteers have built others such as the Open Directory Project (used by web portals like Google and Lycos). Building and maintaining catalogs is a time consuming and expensive task, particularly in those applications with a large number of elements and categories as is the case of web-directories.

Automatic classification techniques use algorithms that learn from these human classifications, so they can only do as well as the human training data provided. In addition, different algorithms can learn different types of patterns in the data. In order to compare the classification performance of different algorithms, researchers have a set of standarised benchmarks, with a particular dataset, and a well defined task. The most popular classification benchmark during the late nineties was a Reuters collection called Reuters-21578 (based on Reuters-22173) with 21578 documents, that had to be classified in about 100 different categories [1]. This benchmark is still used currently to compare the performance of different algorithms but the challenges now lie in moving towards larger scale document classification. In 2002, Reuters released a new research dataset with over 800,000 documents that we discuss in this paper.

There are several Machine Learning (ML) algorithms that have been successfully used in the past [1, 8, 11]. They include Neural Networks, Naive Bayes, Support Vector Machines (SVM) and k-Nearest Neighbours (kNN). Each of these methods has their advantages and limitations on classification performance and scalability. The choice of algorithm will depend on the application, and the amount of data to be used. In web applications, efficiency is of particular importance, since the large number of users and amount of data can make some algorithms unfeasible.

Section 2 of this paper describes two datasets: 1. News: the Reuters RCV1 news stories collection, focusing on the challenges offered by its size, structure and the richness of its XML structure; 2. Web: the Open Directory Project, a major web pages directory where we focus on techniques such as stemming, stopwords and thresholding that can be used to improve performance and reduce the computational complexity of training the classifiers. Section 3 describes the Naïve Bayes method used in this paper, section 4 describes a classification framework [9] that simplifies the integration of automatic classifiers in web-based applications. Section 5 describes the performance results for the news stories classifier and section 6 the results on the catalog. Together, the results in these two collections show how machines learning techniques can be used to manage text of different types. Section 7 concludes.

2 Sample applications: managing news and web directories

The Reuters RCV1 Corpus [7] consists of all English news stories (806,791) published by Reuters in the period between 20/8/1996 and 19/8/1997. The news is stored as files in XML format, using a News ML Document Type definition (DTD). NewsML is an open standard being developed by the International Press and Telecommunications Council (IPTC). The news is written by approximately 2000 reporters and then classified by Reuters specialists in a number of ways. The classified news articles are then syndicated by websites such as news.yahoo.com and news.google.com or periodicals like the Sydney Morning Herald that may or may not have a website.

Due to seasonal variations, the number of stories per day is not a constant. In addition, on weekdays there are an average of 2,880 stories per day compared to 480 on weekends. Approximately 3.7Gb is required for the storage of the uncompressed XML files.

The NewsML schema contains metadata produced by human indexers about themes and categories. When two humans index a document differently they create inter-indexer variations. These can be measured using a correction rate C=(NC/NE)*100, where NE is the number of stories indexed by an editor and NC is the number of times an editor has been corrected by a second editor. Normally untrained editors will have a higher C than more experts ones, but even when they are all experienced correction rates of 10% are common. In the RCV1 collection there are correction rates of up to 77%. Since ML algorithms learn from examples -classifications done by humans- correction rates are an important limiting factor to their performance. Performance measures in classification systems are really a measure of how much they correlate to the human classifiers.

RCV1 data is stored in XML documents providing the metadata information normally required by news agencies and periodicals who need to deliver the stories to end users. NewsML defines a rich schema with entities for title, headline, text, copyright and several types of classification. For our experiments we have used all three available classifications (topic, country and industry) as a single task.

Web applications will increasingly exploit this type of metadata. As it has been discussed by researchers studying the concept of the semantic web, the next revolution in the Internet will come when web applications have access to structured collections of information, and set of inference rules that can be used to perform automated reasoning. This project aims at producing such applications.

For the second evaluation we chose the ODP data set because of: its large size (3.8 million websites classified in 460,000 categories), its flexible licensing (free) and the impact that any improvement would have in very large user base (i.e. Google users) and the the largest volunteer community in a single online collaboration project (almost 60,000 editors). The shear size of the dataset makes the evaluation of new algorithms extremely difficult, and since the goal of this project was not to improve scalability we decided to use a subset.

Our subset is the `Top/Computer/Internet` subcategory that has 41,498 web sites, belonging to 1,300 categories and with 122 that belong to more than one category. The categories are structured in a 7-level hierarchy, with the first one having 32 subcategories and the second one 253. We performed two experiments, one using the two level hierarchy in a to train a novel hierarchical Naïve Bayes algorithm [6], and one that uses the 253 categories with the Naïve Bayes algorithm described earlier, and the data in a flat structure. In the experiment described here, all the documents bellow level 2 are treated as part of one of their parent (level 2) categories. We excluded the 5 level 1 and 12 level 2 categories without documents and the 652 documents that belong to the root category (level 1). The distribution of documents is extremely skewed, with a few categories having a very large number of documents and other having very few. In fact, 77 out of 241 level 2 categories have only one document, but only one level 1 category has less than 10. The ODP dataset we used was originally stored in Resource Description Framework (RDF) format. The dataset is made of a title, description and URL for each web page in the catalog. The title and description is what we call "document" in this paper.

3 Automatic classification with Naïve Bayes

This section introduces Naïve Bayes, one of the automatic document classification technique used in this paper. The K Nearest Neighbors (kNN) algorithm also used in the Reuters RCV1 benchmark is not described, for the sake of brevity, but this is a well known and documented technique. A recent review by Sebastiani [8] provides a more detailed description of this emerging field.

Naïve Bayes is a well known statistical method and has been successfully applied to classification tasks [8, 4]. The different forms of Naïve Bayes are based on Bayes theorem for computing the conditional probability that given a document represented by d it belongs to category c:

$$P(c|d) = \frac{P(d|c)P(c)}{P(d)} \tag{1}$$

The most probable category is given by:

$$\operatorname*{ArgMax}_{c_j \in \mathcal{C}} P(c_j|d_i) = \operatorname*{ArgMax}_{c_j \in \mathcal{C}} \frac{P(d_i|c_j)P(c_j)}{P(d_i)} \tag{2}$$

The estimation of $P(d_i|c_j)$ is difficult since the number of possible vectors d_i is too high. This difficulty is overcome by using the naïve assumption that any two coordinates of the document vector are statistically independent. Using this assumption, the most probable category c_j can be estimated.

$P(c_j)$ is estimated from the number of documents in the training set that belongs to c_j. To estimate $P(d_i|c_j)$, we use the terms of d_i: $t_{i1}t_{i2}\dots t_{ik}$ and $\operatorname{ArgMax}_{c_j \in \mathcal{C}} P(c_j|d_i)$ is then estimated as:

$$\operatorname*{ArgMax}_{c_j \in \mathcal{C}} P(c_j|d_i) \approx \operatorname*{ArgMax}_{c_j \in \mathcal{C}} \prod_{k=1}^{n} P(t_{ik}|c_j)P(c_j) \tag{3}$$

Taking T as the total number of distinct terms in the training set, and using Laplace smoothing the definition of probabilities for $P(t_{ik}|c_j)P(c_j)$, the conditional probablility for the terms can be written as:

$$P(t_{ik}|c_j) = \frac{1 + TF((t_{ik}|c_j)P(c_j)}{|T| + \sum_{s=1}^{N(c_j)} TF(t_s, c_j)} \tag{4}$$

The optimum category can be chosen by:

$$C_{\text{best}} = \operatorname*{ArgMin}_{c_j \epsilon C} \log \frac{DF(c_j)}{|D|} + \sum_{k=1}^{n} \log P(t_{ik}|c_j) \tag{5}$$

This last expression is the one used in our implementation of Naïve Bayes.

Previous studies [11] have shown that Naïve Bayes is accurate and fast compared to other algorithms. In [6] we extended the above method so it can utilize the hierarchical structure of the data.

4 Classifier Design and Implementation issues

4.1 Reusable engineering with a Document Classification Framework

The end goal of our project is to develop classification tools that can be integrated into web applications or other types of information systems seamlessly. As the field of document classification progresses, this integration effort becomes more important so designing the classification software for reusability of design and implementation is of great importance.

We have implemented the Naïve Bayes classifier (and others not described here) using an Object Oriented Application Framework [3] for document classification [9]. The framework has been designed to increase reusability and offers

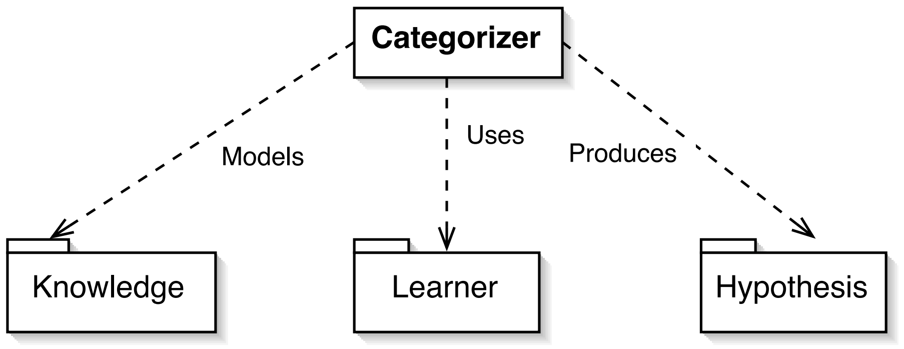

Figure 1: Packages in AI::Categorizer

a structured way to extend it. Figure 1 shows the overall structure of the framework. The *Knowledge* package includes the `KnowledgeSet` class that represents the set of documents, the set of categories, and the many-to-many mappings between them. The *Learner* package is made of the `Learner` class and subclasses as shown in the UML diagram of Figure 2. The abstract `Learner` class provides an interface to train on a set of pre-categorized documents. The result of asking a `Learner` to categorize a previously unseen document is a `Hypothesis` object that make up the *Hypothesis* package. These object may be queried for reporting information such as which categories were assigned, which was the single most appropriate category, what scores were assigned to each category, etc.

The Naïve Bayes and other machine learning algorithms are implemented by subclassing the `Learner` abstract class and by creating two concrete methods: `train()` and `categorize()` that represent the training and test phase respectively, and `get_scores()` and `create_model()` that are called by them internally.

4.2 Feature Selection

Feature selection methods are used to reduce the dimensionality of the feature vectors. This dimensionality reduction reduces computational cost and often increases the classification accuracy. Several feature selection methods are used for text classification including: document frequency (DF), information gain (IG), mutual information (MI), a χ^2-test (CHI), and term strength (TS) [12].

DF is given by the number of documents in which each a term occurs, DF is the simplest method with the lowest computational cost, and high effectiveness so we chose it for the experiments described here.

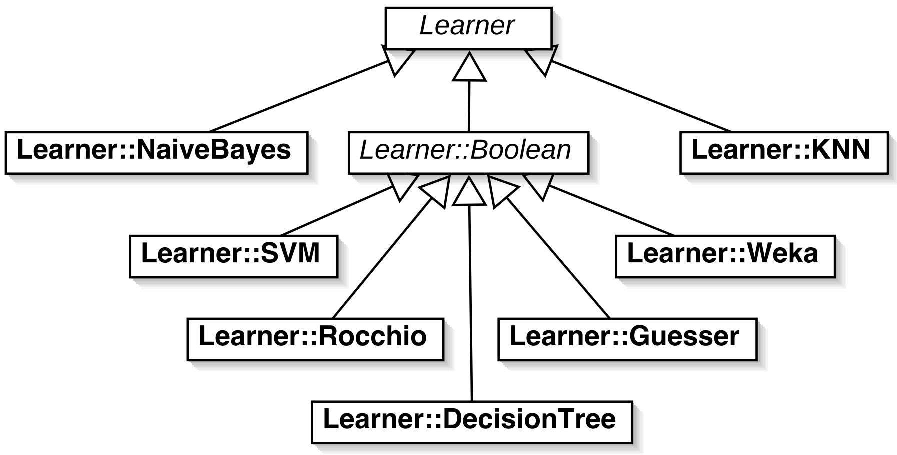

Figure 2: UML diagram for the AI::Categorizer framework

4.3 Thresholding Strategy

Three thresholding strategies are commonly used in the field of text categorization [10]. The thresholds are tuned during training by optimizing the classifiers' performance on the documents in the cross-validation set. The trained classifiers are then used over the documents of the test set. The three thresholding strategies of *Scut*, *Rcut* and *Pcut* are defined by Yang [10] as follows:

- *Scut* - the name stands for score-based classification. For categories $c_i \in C$, given a document d_j, function $\Phi i : D \rightarrow [0,1]$ returns a categorization status value [8], this value is between 0 and 1, which scores the percentage of evidence of the fact $d_j \in c_i$. For each category c_i, we can set a value of threshold τ_i such that, if $\Phi i(d_j) \geq \tau_i$, category $c_i \in C$ is interpreted as $True$. This means c_i is the selected category for document d_j. If $\Phi i(d_j) < \tau_i$, category $c_i \in C$ is interpreted as $False$, so c_i is discarded for document d_j. We can set a single appropriate global threshold for all the category memberships, or we can have per-category local threshold strategy in which a different τ_i is chosen for each different c_i. This is one of the thresholding strategies evaluated in the results section.

- *Rcut* - this approach is a rank-based classification. Given a document d_j, function $\Phi i : D \rightarrow [0,1]$ returns a score value for each category c_i. We then rank the score of each category c_i in descending order, in which we choose the k (an interger value between 1 and m) top-ranking categories to be assigned to each document d_j. When a document is assigned to one and only one category, $k = 1$ this is a commonly used technique. This the second thresholding strategy evaluated in the results section.

	Human Expert	
Machine Classifier	YES	NO
YES	a_j	b_j
NO	c_j	d_j

Table 1: Contingency table for class j

- *Pcut* - the name stands for proportion-based classification. For category $c_i \in C$, given a document d_j, function $\Phi i : D \to [0, 1]$ returns a score value for each document d_j. We rank the score of each document d_j in descending order, in which we choose the k_j top-ranking documents to be assigned to each category c_i, where $k_j = P(c_i) \times x \times m$, m is the number of categories, $P(c_i)$ is the prior probability that document d_j belongs to category c_i, x is the number between 0 and n to be adjusted in a similar way as tuning k for *Rcut* [5].

In the experiments described below we have used SCut and RCut.

4.4 Performance measures

Table 1 describes the possible outcomes of a binary classifier. The “assigned” YES/NO results refer to the classifier output and the “correct” YES/NO refers to the ODP assigned categories. A perfect classifier would have a value of 0 for b_j and c_j.

Using Table 1 we define the three performance measures common in the document categorization literature:

$$Recall = R = \begin{cases} \frac{a}{a+c} & \text{if } a + c > 0 \\ 0 & \text{otherwise} \end{cases}$$

$$Precision = P = \begin{cases} \frac{a}{a+b} & \text{if } a + b > 0 \\ 0 & \text{otherwise} \end{cases}$$

$$F_1 = \frac{2PR}{P + R}$$

The first two measures contain information about whether classification errors are dominated by false positives or false negatives. The trade-off between recall and precision can often be controlled by setting a classifier’s parameters. Both measures should typically be used to describe the overall performance, as neither is particularly informative by itself. The third measure F_1 is an average of R and P.

When dealing with multiple classes there are two possible ways of averaging these measures, *macro-averaging* and *micro-averaging*. In macro-averaging, one contingency table per class is used, then performance measures are computed on each of them and averaged. In micro-averaging only one contingency table is used; an average of all the classes is computed for each cell and the performance measures are obtained therein. The macro-average weights equally all the classes, regardless of how many documents they contain. The micro-average weights equally all the documents, thus biasing toward the performance on common classes.

5 The Reuters classifier

We have tested the extended framework on the Reuters RCV1 data in order to:

1. Assess the feasibility of automatic document classification on large-scale management of news stories.
2. Measure the classification performance of Naive Bayes and kNN classifiers on this new corpus, and find ways to improve it.

The data was selected from the original RCV1 dataset using a simple strategy where we randomly chose 80% of the total amount of dates as training set, and 20% as test set. With this approach the total number of news on each set is not an exact percentage since there are different number of stories on each day. This sampling strategy has the disadvantage of producing test sets that might have somewhat different statistics, for example a particular date (i.e. Bush's election or September 11) could start a new type of documents that did not appear in the training set.

In the two tasks evaluated (Reuters and ODP datasets), we have used Naïve Bayes and an additional algorithm that we considered interesting for comparison. For the Reuters classifier we used kNN. Table 2 shows the classification results for the Naïve Bayes and the kNN classifier described in [2]. We can see that the classification performance is better than for the Naïve Bayes classifier. This is particularly true with the recall measures. The computational performance of kNN is an obstacle and we have not performed tests for all the subsets. KNN is not optimum for web applications, such as the one we discuss in this paper, since all the processing is performed at test time (there is no training), this means that in a real application all the computation would be performed when the news arrive and need to be redirected. kNN showed to have much better recall performance (miR=0.55) compared to Naive Bayes (miR=0.19). This means kNN is able to assign more documents to some category, and this might be of great importance in some applications. Precision (the number of correct classifications) was somewhat degraded by using kNN, but not in a considerable amount.

The second goal of the experiments was to see if the newsML structure could be used to improve performance and find guidelines on how to do it. We expected that giving more weight to more important attributes of the newsML schema (i.e. the title) would be beneficial. The content of title element is similar to the headline. In the experiments we have fixed the weighting factor of headlines and text elements to 1, and we tried several weighting factors for

Method	maR	maP	maF1	miR	miP	miF1
Naïve Bayes	0.327	0.860	0.430	0.540	0.950	0.690
kNN	0.456	0.657	0.487	0.260	0.773	0.389

Table 2: Accuracy measures for Naïve Bayes and kNN on RCV1

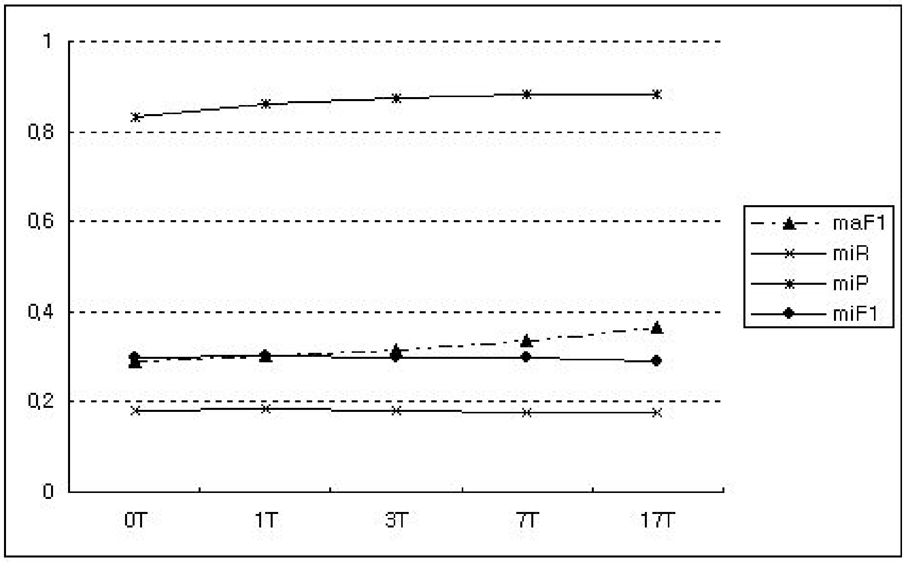

Figure 3: Naïve Bayes performance on the RCV1 dataset

title, T = 0, 1, 3, 7, 17. The weighting factor T means we give T times the frequency count of the word in the document. Figure 3 shows the performance results for the weighting factors, and indicates that micro-precision (miP) of 80-90% are achievable. We must remember that: first, due to the inter-indexer variations (humans do not always agree on how to classify a document), 100% precision is not possible. Second, and that these results only show the correlation with the human classifier, this means we assume humans are always right, and this is not always the case, in fact machines might be doing it better.

Analyzing the performances for different weighting schemes we find that the optimum T seems to be around 3. We can also see that increasing the number of training days from 10 to 50 improved the miP but does not seem to improve the miF1 measure in a considerable amount.

6 The ODP classifier

We have evaluated the Naïve Bayes classifier using the performance measures described earlier. A number of stop-words without intrinsic semantic value (mostly prepositions and articles) were removed from these documents. We also applied Porter's stemming algorithm and only left the stem of each term reducing the total number of distinct terms, and therefore the size of the vector representations to be used. We finally represented the distinct terms of each document as vectors using the Term Frequency / Inverse Document Frequency (TF/IDF) weighting scheme as described in [8]. We compare here the performance of the Naïve Bayes algorithm and a a novel hierarchical Naïve bayes model [6] that exploits the intrinsic hierarchy of the documents in a directory.

The skewed distribution, commonly found in most collections produces lower macro-averaged performance measures. Interestingly, the hierarchical models are better at handling these distributions, probably because despite being skewed, higher level categories normally have enough training documents. This result to our knowledge, has not been discussed by other authors, although it could compensate for the fact we discussed earlier, that errors carried from higher to lower levels in the hierarchy can not be recovered.

6.1 Accuracy

Table 3 sumarizes the results in [6] showing the average results for several experiments using the Naïve Bayes classifier described earlier and a variant that exploits the hierarchical structure of the data. The results for hierarchical classification algorithms reported in the literature, have been mixed, with a few reports of performance improvements for specific datasets. In the case of our ODP data we can see that the $macroF_1$ measure is better for the hierarchical classifier but not so the $microF_1$.

	Flat NB	Hierarchical NB
maR	0.3675	0.4079
maP	0.5086	0.5166
maF_1	0.4017	0.4218
miR	0.7469	0.7032
miP	0.8798	0.8969
miF_1	0.8079	0.7883

Table 3: Performance results averaged for 6 different partitioning of the dataset using flat and hierarchical Naïve Bayes and Scut thresholding strategy

6.2 Thresholding strategy

We evaluated the Scut and Rcut thresholding strategies described earlier. The results for Scut are shown on Table 3 and for Rcut in Table 4.

k	maR	maP	maF1	miR	miP	miF1
1	0.457	0.405	0.367	0.763	0.652	0.7029
2	0.512	0.260	0.249	0.840	0.270	0.408
3	0.560	0.223	0.2171	0.865	0.154	0.2615
4	0.579	0.198	0.194	0.878	0.103	0.1844
5	0.586	0.184	0.178	0.884	0.073	0.136

Table 4: Accuracy results for RCut strategy with different values of k on the hierarchical classifier.

The Rcut thresholding strategy was performed choosing $k = \{1, 2, 3, 4, 5\}$. In Table 4 we can see that the best performance is for $k = 1$. We believe this is because the average of categories which each document is assigned to is approximately 1. When k increases, the recall of the classifier increases, but the precision decreases more rapidly, causing the decrease of the overall $F1$ performance. We also note that the best performance of the Rcut strategy is considerably worse than the Scut thresholding strategy.

6.3 Time performance

The experiments were performed on an Intel Pentium 4 CPU (2.40GHz) with 512 Megabytes of RAM and running Redhat Linux 8.0. The time cost results are given on table 5. We trained the classifiers of parent nodes with all the data in the leaf nodes, this approach increases the computational expense of training the classifier, but makes it more accurate. Classifying new documents is faster on the hierarchical model. It is interesting to note, that in order to improve accuracy we train the level one category classifiers with documents that belong (and have been used for training) in level 2 categories. This procedure makes the training more computationally expensive. In a related project we are looking at how large scale classification tasks can be parallelised and therefore make the classification algorithms more scalable.

Flat Naïve Bayes	
Pre-processing phase	122 Seconds
Training phase	9 seconds
Categorizing phase	106 seconds
Total	237 seconds
Hierarchical Naïve Bayes	
Pre-processing phase	122 Seconds
Training Phase	558 seconds
Categorizing Phase	43 seconds
Total	763 seconds

Table 5: The time efficiency between flat NB and hierarchical NB

7 Conclusions

Generally, information designers define categories that model knowledge domains for a particular application. They define these categories so that the content can be easily understood and located by users (i.e. news topics or web categories). Normally, domain experts will then assign specific documents to these categories.

In this paper we have experimentally investigated automatic classifiers on two datasets: one with news stories from Reuters and the other a web directory with web page descriptions from the Open Directory Project (ODP). A Naïve Bayes classifier was trained, and then used to classify new documents. The results were compared to a kNN (for the Reuters dataset) and to a hierarchical classifier (for the ODP dataset). All classifiers were implemented in an Object Oriented Classification Framework so that they can be integrated into web applications and business workflows.

News stories are normally classified manually, but with this type of techniques automatic or assisted classification can be used to reduce the cost and time involved in managing, syndicating and delivering news stories. The classification performance shown by these experiments is extremely promising and would allow real web applications to use this type of functionality. We obtained 0.9 miP, meaning that 9 out of 10 stories were correctly classified. The miR and therefore the miF1 are somewhat lower because some classes contain few examples and are harder to classify. Naive Bayes classifiers produce lower quality classification but seem to be better suited for applications were classification needs to be performed at real time, kNN instead produces better classification but places to much load on classification time since there is no training.

The same happens with web page directories, particularly the ODP directory used in this paper, where nowadays many human hours are required to update and maintain the information.

Several parameters must be optimized in an automatic classifier, and we have described our use of stemming and stop-words, changes in the amount of training data and thresholding strategies, and techniques for exploiting the structure within an XML document or the hierarchical structure within a collection. In the Reuters set we studied the impact of the amount of data used to train the classifier. In the ODP application we compared two different thresholding strategies: Rcut and Scut and showed that in the ODP application Scut gives a better performance.

The time performance evaluations discussed give an indication of the type of applications that can be targeted with state of the art algorithms and hardware.

The results and previous literature show that the choices to be made for an optimum classifier (i.e algorithm, use of stemming or stopwords, etc) will be different for each dataset and application.

Acknowledgements

JML and RAC acknowledge the Australian Research Council and Hansung University for their financial support. XL and RAC acknowledge the support of the Capital Markets Collaborative Research Centre.

References

[1] Rafael A. Calvo and H. A. Ceccatto. Intelligent document classification. *Intelligent Data Analysis*, 4(5), 2000.

[2] Rafael A. Calvo and Jae-Moon Lee. Coping with the news: the machine learning way. In A. Treloar and A. Ellis, editors, *Proceedings of Ausweb 2003 Conference*, Gold Coast, 2003.

[3] Mohamed Fayad and Douglas C. Schmidt, editors. *Building Application Frameworks*. John Wiley & Sons, 1999.

[4] David D. Lewis. Naive (Bayes) at forty: The independence assumption in information retrieval. In Claire Nédellec and Céline Rouveirol, editors, *Proceedings of ECML-98, 10th European Conference on Machine Learning*, number 1398, pages 4–15, Chemnitz, DE, 1998. Springer Verlag, Heidelberg, DE.

[5] David D. Lewis and Mark Ringuette. A comparison of two learning algorithms for text categorization. In *Proceedings of SDAIR-94, 3rd Annual Symposium on Document Analysis and Information Retrieval*, Nevada, Las Vegas, 1994.

[6] Xiaobo Li and Rafael A. Calvo. Hierarchical document classification using naive bayes. In *8th Australasian Document Computing Symposium*, CSIRO, Canberra, Australia, 15th December 2003.

[7] Tony G. Rose, Mark Stevenson, and Miles Whitehead. The reuters corpus volume 1 - from yesterday's news to tomorrow's language resources. In *3rd International Conference on Language Resources and Evaluation*, page 7, May 2002.

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

[9] Ken Williams and Rafael A. Calvo. A framework for text categorization. In The University of Sydney, editor, *7th Australasian Document Computing Symposium*, Syndey, Australia, 2002.

[10] Yiming Yang. A study on thresholding strategies for text categorization. In *SIGIR*, 2001.

[11] Yiming Yang and X. Liu. A re-examination of text categorization methods. In *22nd Annual International SIGIR*, pages 42–49, Berkley, August 1999.

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