Hierarchical Classification for Multiple, Distributed Web Databases

Hui Yang and Minjie Zhang*
University of Wollongong, Wollongong, NSW 2522, AUSTRALIA

Abstract

The proliferation of online information resources increases the importance of effective and efficient distributed searching. Our research aims to provide an alternative hierarchical categorization and search capability based on a Bayesian network learning algorithm. Our proposed approach, which is grounded on automatic textual analysis of subject content of online web databases, attempts to address the database selection problem by first classifying web databases into a hierarchy of topic categories. The experimental results reported demonstrate that such a classification approach not only effectively reduces the class search space, but also helps to significantly improve the accuracy of classification performance.

Key Words: Hierarchical classification, Bayesian classifiers, multiple web databases.

1 Introduction

As the Internet has rapidly proliferated over the past decades, especially World Wide Web (WWW), web users have witnessed an explosion in the availability of online information from distributed web databases. Despite the usefulness of various search engines such as Yahoo and AltaVista, web users still feel frustrated in profitably utilizing such large amounts of information. One potential problem of current dilemmas is the *database selection* problem; that is, how to optimally select a number of databases from such a vast information source, which are most likely to provide the useful information with respect to a given user query. We believe that an automatic and robust method called database classification, which partitions multiple, distributed web databases based on their subject contents into classification schemes, will be helpful for efficient and fruitful database selection.

Classification of textual documents as a useful technique for information retrieval has long attracted significant attention from information science researchers [20]. Since web databases usually consist of hundreds of thousands textual documents, text classification techniques can easily be applied to database classification.

There are quite a number of special–purpose databases which focus on documents in confined subject domains such as IEEE and ACM digital library databases on the Internet, while there are also some large-scale general–purpose databases which cover wide-ranging web contents of various topic categories. The above characteristics of web databases make it feasible to organize and manage web databases in a structured hierarchy of topics. We believe that the use of a hierarchy can decompose the classification task into a set of smaller tasks, each of which only corresponds to a small split in the hierarchical tree. Such a hierarchical structure makes accomplishment of the classification work more effective and efficient.

Several researchers have recently investigated the use of hierarchies for text classification and have received encouraging achievements [12, 19, 25]. Our work differs from earlier work in the following important aspects. First, as we know, most of hierarchical classification methods only treat the classes (categories) in the hierarchy as a simple vertical parent-child relationship between different levels. But, in practice, the relationships between classes in the hierarchy are usually more complicate such as the horizontal relationship between classes. Hence, to better express the correlation of the classes, we consider the horizontally logical relationship among the classes at the same level during the construction of the topic hierarchy. Second, we use a variation of a probabilistic Bayesian model based on Naive Bayes learning techniques [16] for automatic database classification. This model takes the special characteristics of databases into account, thus making itself more applicable to database classification rather than text document classification. Third, we propose a new category assignment strategy called possibility-window, which allows more appropriate categories to be chosen regarding the content of the databases. Our experimental results reported in this paper have demonstrated that our hierarchical classification methods can significantly improve the performance of database classification.

The rest of our paper is structured as follows. In Section 2 we present an overview of text classification for information retrieval. In Section 3 we discuss the specific technique we use for database classification. We focus on a probabilistic model that provides a framework for the construction of a set of classifiers for a category-based search. In Sections 4 and 5

[*] School of Information Technology & Computer Science. E-mail: hy92@uow.edu.au, minjie@uow.edu.au.

we provide our experimental methodology and a variety of experimental results supporting our approach. Conclusions and future work are provided in Section 6.

2 Related Work

With the exponential growth of digital libraries, and online databases on the Internet, system-aided classification techniques, which are used to organize and manage these emerging large–scale information sources, have recently become more and more promising and appealing to information science researchers. While work in text database classification is relatively new, a substantial body of research which looks at text document classification, has been occurring for decades.

Although manual classification might produce good quality category assignments, it is hampered by the bottlenecks inherent in the manual way such as expensive costs (i.e., human participation) and the lack of scalability. Obviously, in many ways, machine learning techniques provide suitable solutions to solve the above problems. In recent studies a number of machine-learning techniques have been proposed to address the text classification problem. Among them exists mainly two types of learning techniques: supervised learning and unsupervised learning. Although they all depend on some labeled training data for category models to learn, the main difference of unsupervised learning from supervised learning is that in unsupervised learning once category models are trained, new data instances can be added with little or no outside interference (e.g., human efforts).

A wide range of research has been devoted to text-based classification algorithms based on supervised learning techniques, including the linear classification algorithm [17] and decision tree or rule induction [1, 24]. In addition, there has been some work on unsupervised learning in textual analysis application [2, 10]. Moreover, some comparative studies were found [3, 8] which discovered various clustering methods perform differently under different sets of conditions.

Due to good performance on clustering and learning abilities [14], neural network techniques have recently been studied for text classification application by researchers. There exists some neural network clustering algorithms for text classification, which are based either on supervised or unsupervised learning. Schutze, et al. [24] present a two-layer back-propagation-like neural network for document learning. Chen, et al. [4] adopt a variation of the hopfield network for concept classification of electronic brainstorming comments. In their recent work [5] they have also developed a multilayed Kohonen self-organizing feature map (SOFM) to categorize Internet homepages.

Although text database classification is a relatively new research area, a number of research efforts have still been devoted to it. Gauch, et al. [11] manually construct query probes to facilitate the classification of text databases. In [15] Ipeirotis, et al. introduce an automatic database classification method based on the number of matches that each query probe generates at the databases. The formation of queries comes from a rule-based document classification. In [19] a concept hierarchy is constructed for text database categorization. Each concept description is treated as a query that is submitted to the database. The documents retrieved from the database are used to calculate the similarity between the concept and the database.

3 Probabilistic Framework for Hierarchical Classification

For a moderate number of databases (e.g., hundreds of databases), a “flattened” class space with each class (topic) for every leaf in the hierarchy is suitable. We can train a single classifier so that each database is precisely categorized into one of the possible basic classes. Unfortunately, when the number of databases is very large, such as thousands or tens of thousands, this simplistic approach will be hampered by computational costs. This is because using a single flattened classifier, the similarity estimation of the most relevant topic for each database could be time-consuming in the case of a large number of databases. For this reason, we introduce a hierarchical structure of topics, which allows us only to focus on the relevant topic area beginning from the root level. As a result, the topic number of probability estimation can be drastically reduced. Comparatively speaking, the hierarchical approach obtains significant efficiency gains over the standard flat approach.

In this paper we have chosen to focus on probabilistic methods used for hierarchical classification. The probabilistic model provides an efficient means for producing a set of classifiers used in the hierarchy of topics. In this section we give a brief overview of the probabilistic model and its application to database classification.

3.1 Hierarchical Structure of Topics for Databases

Hierarchical structure of topics has long been used in special purpose collection of documents [13]. More recently, several large-scale Internet search engines such as Yahoo and Infoseek have also adopted such hierarchies to manage the World Wide Web to conveniently guide users to their appropriate topic of interest. It is sensible to utilize the existing well-known category hierarchies such as Yahoo to build our own hierarchical structure since they are widespread and familiar to web users.

We first describe the structured hierarchy of topics. Figure 1 shows a topic directory which is a hierarchical architecture of three layers. The tree hierarchy contains nodes at different levels indicating topics of interests to users. It is easy to see that topics in the hierarchical structure are ordered from general broad topics (top) to more narrow ones (bottom), and the leaf nodes point to specific and unambiguous subjects. Topics close to each other in the hierarchy typically have a lot more in common with each other than topics far apart. Nodes “Database” and “Platform” are close in the content of subject, but they are quite different from the nodes “Video” and “Image” which belong to another parent node “Multimedia”.

The hierarchical structure treats the topics as a collection of topic sets, each of which consists only of a small set of topics. As we see from Figure 1, each node (except root node and leaf nodes) in the hierarchy actually have two roles, a parent node and a child node. On the one hand when it is used as a parent node, in practice, it is a cluster which corresponds to a topic set. On the other hand, when it is used as a child node, it is only a single topic (class). For example, for node "computers", when it is a child node for the root node, it only points to the topic "Computers", while when it is a parent node for node "software", it actually corresponds to the topic set {"software", "hardware", … , "Internet", "multimedia"}.

For all parent nodes, each node has a classifier that distinguishes one class from a set of classes. Since the classifier at a node only needs to focus on a small set of topics rather than overall topics, it is possible to make the classification more accurate. Therefore, such a hierarchical topic structure actually decomposes the classification task into a set of simpler subtasks, which can be solved much more efficiently and, hopefully, more accurately as well.

3.2 Bayesian Classifiers

3.2.1 Feature Selection. In order to distinguish the appropriate class from a set of classes in the hierarchical classification scheme, a set of features that have enough discriminating power are needed for the classifier. For example, in text classification, the features are the words that strongly associate with one specific category. Theoretically, the more the features represent a class, the more useful it is for the classifier to distinguish the classes. Unfortunately, it is impractical to simply implement the above idea as described, since the computation cost is exponential in the number of the features. Thus there is a fundamental trade-off between a large feature space and classification accuracy. However, the issue of how best to select optimal features for the classes will be studied in our future research and will not be reported in this paper. Here we employ a feature selection technique called Latent Semantic Indexing (LSI) reported in [6, 7] to produce a smaller subset of the most important features for the class with minimal loss in accuracy.

LSI serves as a means of data compression that represents features and documents by a low-dimensional linear combination of orthogonal indexing variables. It can best capture the important information contained in a large number of terms with a much smaller number of factors. In particular, it is useful for eliminating the redundancy in word features that is due to term dependence.

3.2.2 Construction of Naive Bayes Classifiers. A training set of labelled feature vectors are used to induce a classification model. This model is then used to predict the class label for a set of previously unseen data instances (e.g., web databases). Optimistically, the feature vectors will fully determine the appropriate topic class. As we know however, the training data, which is simply a sample from the underlying population of relevant documents about the class, may not adequately characterize its true distribution since the training set provides only a rough approximation. Thus we use a probability distribution to model the classification function. Formally, for each feature f_i in the feature space *F*, we have a probability $\Pr(f_i | c_k)$ in the possible cluster *C*, where $c_k \in C$, and *C* is a set of classes denoted as $C = \{c_1, c_2, \cdots, c_M\}$; $f_i \in F$, and *F* is a set of features described as $F = \{f_1, f_2, \cdots, f_N\}$; $\Pr(f_i | c_k)$ is the probabilistic distributions for each feature f_i ($f_i \in F$) in class c_k. This ensures that in the class network each feature will be taken into account for the classification. A learning algorithm utilizes the feature information that is obtained from the training set using the *LSI* method to construct a classifier.

Recent work in supervised learning has shown that a simple Bayesian classifier with the assumption of independence among features is competitive in textual classification [17, 26]. A Bayesian network [21] provides a compacted representation of probabilistic distributions over the training data. A Bayesian classifier is simply a Bayesian network applied to a classification domain.

During construction of the classifiers in a hierarchical scheme, we limit our attention to a set of Naive Bayes network structures. Unlike other ordinary Naive Bayes structures, we add horizontal lines between some classes in the same level to common that one document contains the content of multiple topics rather than a single one. Thus, the relationships of these

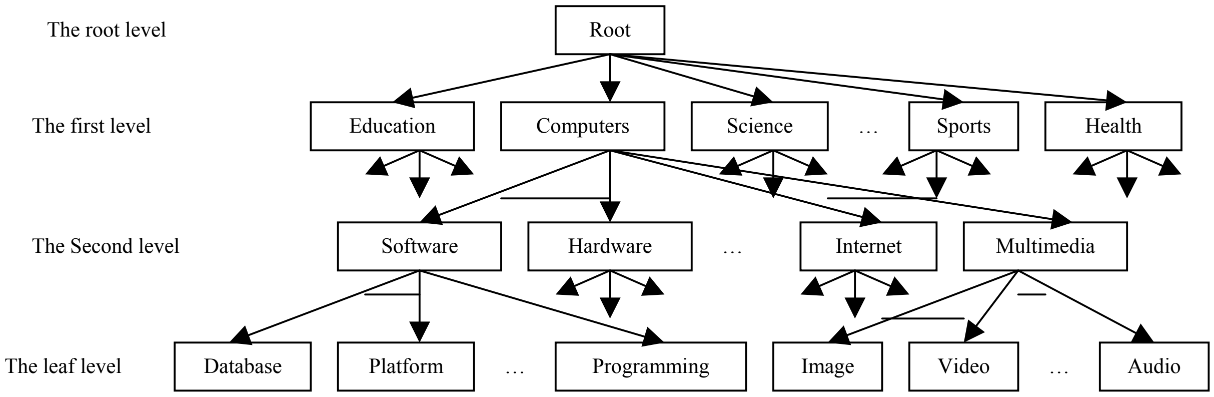

Figure 1: A Small Fraction of the Topic Hierarchical Structure

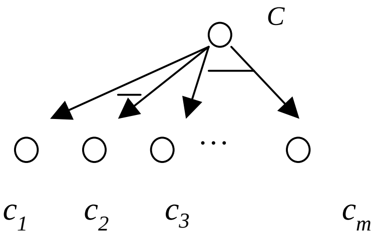

Figure 2: The structure of the Naive Bayes Network for a classifier

topics are closer than those of other topics. For example, for the cluster "Computers", classes "Software" and "Hardware" usually have more cohesive relationship than class "Internet" or class "Multimedia" (see Figure 1). A line is then drawn between them denoting "*AND*" connection. Note that the process of adding these edges may involve a heuristic search on a hierarchical structure.

Like most agglomerative clustering methods [12, 25], our hierarchical clustering algorithm constructs a cluster hierarchy from the bottom to the top by merging some subject-relevant classes one at a time. At the beginning, just the topics of the leaf level in the hierarchy associate with labeled training documents as the classes. For each leaf-level topic (class), a separate round of optimal feature selection is employed by the ***LSI*** method to obtain the most indicative features for this topic. Note that this feature selection is done starting from the original feature set which has been preprocessed by stop-word removal and stemming. Stop-words are first removed because they occur very frequently in the documents such as the words: 'the', 'a', 'we', 'of', 'but'. These frequent words are not helpful to the search results, and require a lot of processing power. In addition, many words have different variations. These variations have the same or similar meaning(s) such as the words "stepped", "steps" and "stepping". Due to different spellings, they cannot be directly matched to each other. Using Porter's stemming algorithm [22], different variations of the same word can be converted to the same word stem which is useful during the classification.

With the framework of multinominal Naive Bayes text classification [16], the classifier can parameterize each class separately with the prior probability for each class c_k in the leaf level, $\Pr(c_k)$ and the probabilistic distributions for each feature f_i ($f_i \in F$) given the class c_k, $\Pr(f_i \mid c_k)$. Thus, each leaf-level topic, in practice, consists of a set of the most important features with the parameters $\Pr(f_i \mid c_k)$.

The last-second-level topics are then used as class labels. Note that every node in the last-second-level topic has only a subset of the total class labels in the leaf level. As we have observed, the most informative features at the lower-level child class are likely to be particularly useful for its high-level parent class. We will only present our method for constructing the classifiers at the last-second level. The construction of the classifiers at other higher levels are implemented in a similar way.

Definition 1. Cluster *C* is a superclass of a number of classes presented as $C = \{c_1, c_2, \cdots, c_M\}$, where c_k $(1 \le k \le M)$ is a class of cluster *C*. F^C is the feature space for the cluster *C*, which is described as $F^C = \{f_1, \ f_2, \ \cdots, \ f_N\}$, where F^C is the feature space set of all the classes, namely, $F^C = F_1 \cup F_2 \cup \cdots \cup F_M$, F_k corresponds to the feature space of class c_k, and $f_i \, (1 \le i \le N)$ is a distinct feature vector in this feature space F^C.

Definition 2. For the given cluster *C*, $R(c_j, c_k)$ is a logic connection function for the classes, c_j and $c_k \, (1 \le j, \ k \le M, \ j \ne k)$.

Assume that for the given cluster *C*, the probabilistic distribution for each feature f_i to each class c_k, $\Pr(f_i \mid c_k)$, and the prior probability for each class c_k, $\Pr(c_k)$, are known. We can easily calculate the probabilistic distribution for each feature f_i to the cluster *C*, $\Pr(f_i \mid C)$. For a feature vector f_i, assume that f_i exists in the feature space F_k of a certain class c_k, where $c_k \in C$.

1) If $\forall \ F_j \in F^C (1 \le j \le M, \ j \ne k)$, $f_i \notin F_j$, then

$$\Pr(f_i \mid C) = \Pr(f_i \mid c_k) \tag{1}$$

2) If $\exists \ F_j \in F^C (1 \le j \le M, \ j \ne k)$, $f_i \in F_j \cap F_k$, then

$$\Pr(f_i \mid C) = \begin{cases} Min(\Pr(f_i \mid c_j), \Pr(f_i \mid c_k)) & R(c_j, c_k) = AND \\ Max(\Pr(f_i \mid c_j), \Pr(f_i \mid c_k)) & otherwise \end{cases} \tag{2}$$

The prior probability for the cluster *C*, $\Pr(C)$ will be

$$\Pr(C) = \frac{1 + \sum_{c_k \in C} \Pr(c_k)}{M + N} \tag{3}$$

where *N* indicates the number of classes in the level in which the cluster *C* lies, and *M* is the number of classes in the cluster *C*.

Since the feature space F^C for cluster *C* is the combination of all the feature space of all the classes, the size of the feature will possibly be very large, so it may result in the time-consuming problem of classification computation for a classifier. To solve the above problem, we will eliminate those features with too small $\Pr(f_i \mid C)$ in that such features are generally not able to improve classification accuracy. Finally, for each last-second-level class, we construct a separate classifier with the appropriate reduced-feature set.

For example, assume that the cluster *C* "software" consists of three classes (see Figure 3), c_1 = "Database", c_2 =

"Platform", and c_3= "Programming", respectively. Among them, the logic connection of the classes, c_1 and c_2: $R(c_1,c_2)$= "*AND*". For each class, the prior probabilities for each class are, $\Pr(c_1)$=0.3, $\Pr(c_2)$=0.4, and $\Pr(c_3)$=0.3. Thus, according to Equation 3, the prior probabilities for cluster *C*: $\Pr(C)=\frac{1+(0.3+0.4+0.3)}{3+10}=0.2$ (assume that the number of classes of the level in which the cluster *C* belongs is 10).

The probabilistic distributions for the feature f_1, "Windows", to each class are, $\Pr(f_1 \mid c_1)$=0.4, $\Pr(f_1 \mid c_2)$=0.5, and $\Pr(f_1 \mid c_3)$=0.45. According to Equation 2 and the logic relationship among the classes, the probabilistic distribution for the feature f_1 to the cluster *C*, $\Pr(f_1 \mid C)=(0.4 \cap 0.5) \cup 0.45 = 0.45$. Since the feature f_1, "Oracle" only appears in the class c_1, the probabilistic distribution for the feature f_2 to the cluster *C*, $\Pr(f_2 \mid C)=\Pr(f_2 \mid c_2)=0.3$. Finally, we can obtain the feature space F^C and the probabilistic distributions for each feature in F^C shown as Table 1.

3.3 Hierarchical Classification Strategies

3.3.1 Hierarchical Classifying Web Databases. At first, the definition of a web database, S_i, is described as follows.

Definition 3. A web database S_i is a 3-tuple, $S_i =< C_i, D_i, T_i >$, where C_i is the set of subject domain (topic) categories that the documents in database S_i come from; D_i is the set of documents that database S_i contains, and T_i is the set of distinct terms that occur in database S_i.

Definition 4. Suppose the database S_i has *s* distinct terms, namely, $T_i = \{t_1,\ t_2,\ \cdots,\ t_s\}$. Each term in the database can be represented as a two-dimension vector $\{t_j, w_j\}$ $(1 \le j \le s)$, where t_j is the term (word) occurring in the database S_i, and w_j is the weight (importance) of the term t_j due to its term frequency (i.e., the number of occurrence) in the database.

Definition 5. Consider that there exists a number of topic categories in database S_i, which can be described as $C_i = \{c_1,\ c_2,\ \cdots,\ c_p\}$.

Definition 6. Suppose that database S_i consists of a set of documents which can be represented as a vector: $D_i = \{d_1,\ d_2,\ \cdots,\ d_t\}$, where *t* is the number of documents in the database S_i. For each document d_i $1 \le i \le r$, d_i can be similarly described as $d_i = \{t_{i1},\ t_{i2},\ \cdots,\ t_{is}\}$, and each term in document d_i is also a two-dimension vector $\{t_{ij},\ w_{ij}\}$ $(1 \le j \le s)$, w_{ij} is the weight (important) of the term t_{ij}.

Note that for the sake of expressive convenience, t_j in Definition 4 and t_{ij} in Definition 5, in practice, refer to the same term. The only difference between them is the weight of the term, w_j and w_{ij}:

$$w_j = tf^*{}_j \times icf_j \tag{4}$$

$$w_{ij} = tf_j \times idf_j \tag{5}$$

where tf_j^* is the term frequency of the term occurring in the database S_i, icf_j is the database frequency (the number of databases that have the term) in the test database collection. tf_j

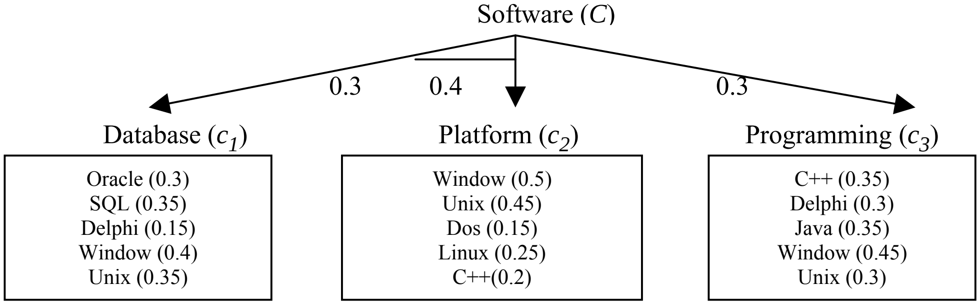

Figure 3: The Nave Bayes structure of the classifier for cluster "software"

Table 1: The feature space F^C and the probability distribution for each feature in F^C for the cluster *C*

The Feature Space F^C	Oracle	SQL	Delphi	Windows	Unix	Dos	Linux	C++	Java
$\Pr(f_i \mid C)$	0.3	0.35	0.3	0.45	0.3	0.15	0.25	0.35	0.35

is the term frequency of the term occurring in the document d_i, *idf* the database frequency (the number of databases that have the term) in database S_i. All the weights are normalized with the cosine function [23], which can range from 0 and 1.

Note that in order to reduce the term space in the database and improve classification accuracy, these terms have been preprocessed by removing stop-words, stemming, and eliminating rarely appearing words.

Once the classifiers in the hierarchical classification scheme are built, we can start the work of assigning a web database S_i with the appropriate topics (categories) into the hierarchy. In practice, the optimization process for category search in probabilistic category hierarchy can be implemented using a heuristic search technique called *best first search* to find the "*best*" candidate (candidates) over the category space of each level in this hierarchy. At first, instead of taking the first node at the first level to classify the database S_i, the hierarchy classification scheme first chooses the *best* node (topic) with the biggest similarity score using the root classifier. Then the hierarchy scheme further sends the database down to the children nodes associated with the chosen first-level category (topic). Apply the same category search strategy until one or more categories at the leaf level are chosen for the database (detail on the search will be described in subsections 3.3.2 and 3.3.3).

At each level, except the root level, after one class is chosen by the classifier as the most likely topic for the database S_i, it will automatically be added into the class set C_i of database S_i (recall Definition 5). Clearly, the database that belongs to a class (a leaf node in the hierarchy tree) is also assumed to belong to each of the nodes (classes) along the path to the root. For example, if class "database" is chosen by the classification scheme as a appropriate topic for database S_i, then classes "computers" and "software" must be in the class set C_i of database S_i (see Figure 1).

Note that one of the advantages of our approach is that the classification mechanism can be utilized in parallel hardware. When one or more categories in one level are chosen by the classifier, the classification processes at the lower level can proceed in parallel in different subsets of classes rooted from the chosen parent nodes. Therefore, this method improves the classification efficiency.

3.3.2 Category Search Strategy. In each level of the hierarchy, the category search strategy will be executed by the following steps:

1) First calculate the posterior probability $\Pr(c_k \mid S_i)$ of class c_k for the database S_i, where c_k is a class of the chosen cluster C in the higher level.
2) Then Rank all the classes in the chosen cluster C based on the posterior probability $\Pr(c_k \mid S_i)$.
3) Assign the most likely categories to the database S_i employing the category assignment strategy.

Step 1 searches a set of categories in the chosen cluster C with the classifier constructed by Naive Bayes network. The measure of likeliness for the database S_i is the posterior probability $\Pr(c_k \mid S_i)$. Given the parameters, $\Pr(f_i \mid c_k)$ and $\Pr(c_k)$, the posterior probability $\Pr(c_k \mid S_i)$ can be determined by Bayes' rule and the occurrence frequency of features of the class c_k in the database S_i:

$$\Pr(c_k \mid S_i) = \frac{\Pr(c_k)\Pr(S_i \mid c_k)}{\Pr(S_i)}$$

$$= \frac{\Pr(c_k)\sum_{t_i \in T_i} \Pr(t_i \mid c_k) w_i}{\sum_{c_s \in C}\left(\Pr(c_s)\sum_{t_i \in T_I} \Pr(t_i \mid c_s) w_i\right)} \tag{6}$$

where, if the term t_i occurs in the feature space F_k of class c_k, that is, $t_i = f_j \quad (f_j \in F_k)$, $\Pr(t_i \mid c_k)$ in fact equates to $\Pr(f_j \mid c_k)$; otherwise $\Pr(t_i \mid c_k) = 0$; w_i is the word weight of the term t_i (recall Definition 4).

3.3.3 Category Assignment Strategy. In Steps 2 and 3, it is usual to make the ranking C based on the posterior probability $\Pr(c_k \mid S_i)$. According to the category ranking, one or more categories are assigned to the database S_i using the categories assignment strategy. Many category assignment methods have been proposed [19]. For example, the famous top-*K* method chooses the top *K* categories as the most likely categories, and the probability-threshold method assigns all the categories with the likeness value over a predefined threshold τ. However, one problem of the above two methods is that due to the difference of the number of classes in each level of a hierarchy tree, a simple *K* or τ can not correctly reflect the proper class number in different level.

In this paper we propose a new approach to selecting the "winning" classes in cluster *C*. Consider such a scenario where for some web databases, especially large-scale, general-purpose web databases, the distribution of the documents concerning various topics are usually quite different. When one or several categories do not fully reflect the factual categories in the database, we use a window to capture as many categories as possible. This possibility-window method is extended from the possibility-threshold method. The window is defined as follows:

$$\frac{\Pr_{\max}(C \mid S_i)}{\Pr(c_k \mid S_i)} \le 1 + \varepsilon \qquad (1 \le k \le M) \tag{7}$$

$$\frac{\Pr(c_k \mid S_i)}{\Pr_{\max}(C \mid S_i)} \geq 1-\varepsilon \qquad (8)$$

where $\Pr_{\max}(C \mid S_i)$ is the maximum of the posterior probabilities of all the classes in cluster *C*, and ε is the parameter of window size. As long as the posterior probability of category c_k satisfies all of the above conditions, category c_k will be chosen as the appropriate category for the database S_i.

The size of the window for selection of categories which are considered relevant to the content of the database depends greatly on the window parameter ε. Using huge windows means expensive computation but small ones discard possible valuable categories. Therefore, ε is a sensible parameter for classification accuracy. We ran a number of experiments to determine its appropriate value in Section 5.

4 Experiment

4.1 Testbed

To evaluate our database classification techniques, we first need to obtain hierarchical classified text data. Our testbed was based on artificial data set which is the Reuters 21578 Distribution 1.0 [18].

The Reuters 21578 distribution 1.0 data set. This data set consists of 21578 articles, each of which has been manually labelled with one or more categories. Although topic categories have not been organized with a hierarchical structure, we still use the label information to construct a 3-layer hierarchy. In the hierarchy, 4 categories at *level-1* roughly correspond to economy, currency, commodity, and energy, and there are 5, 5, 6 and 2 topics, respectively, as the topics of *level-2* (see Table 2). Ninety-three topic categories are extracted from 135 financial topic categories as the topics of the leaf level. Except for the vertical parent-child connection of different levels, we added some horizontal logic "AND" relationships among the classes at the same level. In order to get the most representative feature space for each topic category at the leaf level, for those categories which have the logic relationship with other same-level categories, we intentionally select some articles which are labelled with two correlative topic tags as the training data for these topics.

Tables 3 and 4 show some statistics of Reuters dataset.

Database classification is made up of two phases: training phase and test phase. To guarantee enough labelled training documents for Naive Bayes learning, a large set of documents with known category labels are used to build an initial probabilistic Bayesian model, and another set of data is used to identify optimal model parameters. Test data is used to hierarchically classify new databases.

4.2 Experimental Setup

To evaluate the database classification performance of our proposed classification methods, we considered a number of variations which are shown as follows:

Table 2: 3-level clusters of Reuters categories

ECONOMY	
Level 2	Nation, Person, Money, Price, Corporate
Leaf Level	bop, gnp, lei, jobs, income, interest, money-supply, money-fx, housing, install-debt, cpu, earn, reserves, cpi, wpi, retail, ipi, inventories, trade, earn, acq
CURRENCY	
Level 2	Ocean, America, Europe, Asia, Africa
Leaf Level	dir, can, saudriyal, yen, rand, dfl, lit, nkr, stg, dmk
COMMODITY	
Level 2	Agriculture, Metallurgy, Non-Metal Industry, Food, Farming
Leaf Level	barley, grain, corn, wheat, cotton, rice, iron-steel, platinum, copper, nickel, cold, lead, silver, strategic-metal, tin, zinc, hog, carcass, livestock, pork-belly, l-cattle, wool, lumber, plywood, alum, rubber, palladium, cocoa, coffee, coconut, groundnut, meal-deed, oilseed, orange, potato, sugar, tea, veg-oil, copra-cake, oat, palm-oil, palmkernel, rape-oil, rapeseed, sorghum, soy-meal, soy-oil, soybean, sun-oil, sunseed, tapioca, ship
ENERGY	
Level 2	Oil, Gas
Leaf Level	crude, heat, fuel, gas, pet-chem, jet, naphtha, nat-gas, propane, ship

Table 3: Statistics about the training dataset for Reuters databases

	Reuter Dataset Training
Number of documents	2850
Number of classes	90
Mean relevant documents per class	32
Mean words per document	176

Table 4: Statistics about the test dataset for Reuters databases

	Reuter Dataset Test (Small-Scale databases)	Reuter Dataset Test (Large-Scale databases)
Number of databases	45	24
Documents per database	50~150	1000~2500
Mean classes per database	6.5	38
Mean words per database	13347	402124

1) The effect of adding logic correlations among the classes at the same level.
2) The comparison of three different category assignment strategies (probability-window, probability-threshold,

and Top-K).

3) A hierarchy model vs a flat model.
4) The effects of various factors in database structure on classification performance.

   a. The similarity measurement methods employed.
   b. Database size.
   c. The distribution of database clusters.

We conducted experiments with the above variations, and evaluated the impact these aspects had on the final classification results.

4.3 Evaluation Measurement

The effectiveness of database classification can be measured as recall *R* and precision *P*, which can be calculated by the following equations:

$$R = \frac{\text{the number of categories that are correctly assigned to databases}}{\text{The number of categories that should be assigned to the database}} \quad (9)$$

$$P = \frac{\text{the number of categories that are correctly assigned to database}}{\text{The number of categories that are assigned to the database}} \quad (10)$$

To combine precision and recall into one number, *E* Measure is commonly used:

$$E = 1 - \frac{2}{\frac{1}{P} + \frac{1}{R}} \quad (11)$$

where the value of *E* varies from 0 to 1, and is inversely related to selection performance. When P=R=1, selection performance is “perfect”and E=0.

For example, assume that database S_i has the documents of five topic categories (see Figure 1), D_i={computer, software, database, Internet, network}. There are six topic categories which are chosen by the database classification method. They are “computer”, “software”, “database”, “platform”, “programming”, and “science”, respectively. Only three categories in the returned category list are correct. The precision will then be 0.5 (*P*=3/6=0.5), and the recall will be 0.6 (*R*=3/5=0.6). Finally, we get the database classification effectiveness with *E*=0.454.

It should be noted that an “ideal” selection result is *R*=*P*=1. Unfortunately, precision and recall sometimes are antinomy to each other. When recall goes up, precision usually (though not always) goes down. Thus there exists the possible trade-off between recall and precision.

The above evaluation methods will be used to estimate the database classification effectiveness in our experiments reported in Section 5.

5 Experimental Results and Discussions

5.1 Accuracy for Different Category Assignment Methods

First we compare different category assignment methods with probability-window, probability-threshold, and Top-K in terms of selection accuracy. It is interesting to see how the variety of window parameter ε, threshold parameter τ, and class parameter *K* affects the classification performance. To see more clearly, we plot the accuracy curve in Figures 4 and 5.

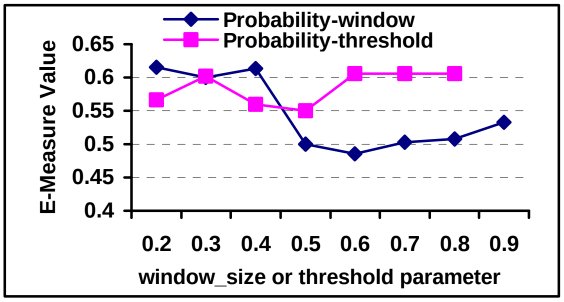

Figure 4: The average e-measure value of the category assignment methods. (a) probability-window (ε ranges from 0.2 to 1); (b) probability-threshold (τ ranges from 0.2 to 0.8)

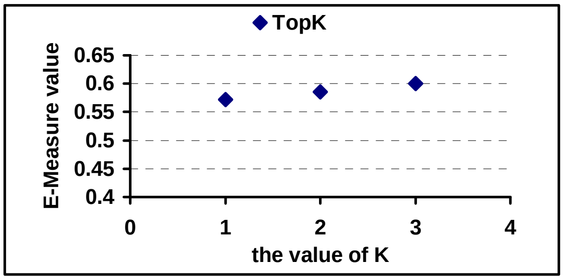

Figure 5: The average e-measure value of the Top-K category assignment methods

In order to allow a fair comparison, we ran experiments over the Reuters databases with varying values of parameters ε and τ ranging from 0.2 to 1, and *K* being 1, 2, 3, respectively. As noted, for high value of window parameter ε, the classification scheme will have the databases assigned to more leaf nodes (topics), which may lead to high recall. But, as mentioned previously, it raises the shortcoming of low precision. These characteristics might affect the classification performance and the same would be expected for the probability-threshold and the Top-K methods. Therefore, the selection of the proper value for these parameters will be focused on the important tradeoff between precision and recall, namely, low E-Measure value.

As seen in the accuracy results for Reuters testing dataset using our probability-window method, selection of the appropriate window parameter ε can have a large impact on classification accuracy. When ε varies the range between 0.5 and 0.8, the results are significantly improved over the accuracy on the original testing dataset. It is understandable that the topic distributions in the database are uneven. Some topics have hundreds of documents in the database, but some topics only have a few or tens of documents. The irregular distributions make the size of the probability window broaden in order to cover the topics with a few documents within the window.

By contrast, the probability-threshold and the Top-K seem virtually incapable of taking advantage of the hierarchical structure. In very few cases did we observe little improvement in the accuracy for the Top-K method. We conjectured that the possible reason for this drawback is that the Top-K method is too simple to completely reflect the complicated relationships between the classes in the hierarchy.

5.2 The Effect of Logical Correlations Between the Classes.

We were also interested in the impact of adding logical relationships between the classes in the hierarchy on classification performance. As would be expected, the logical relationships between the classes are useful indicators of class relevance, which can help improve selection effectiveness. The classification results in Figure 6 show that when the right logical relationships are used for classification, classification performance increases by an average of 9.75 percent. This suggests that it is possible that the common features occurring in the related classes affect classification effectiveness to some extent. We note that the method of adding a logical relationship between the classes significantly outperforms the non-logical method regardless of the number of features used. For the common feature vectors in the logic related classes, we used the minimum function (recall Equation 2) to decrease their impact on the classification. Experimental results proved our opinion that these features are not the best indicators to disinguish the classes, since they co-occur in one or more classes.

During the experiments we employed an aggressive feature selection method to reduce the feature space to 50, 100, 150 and 200 features in order to observe the impact of the number of features on the classification performance. As noted, when feature selection was aggressively employed versus the case where all topic category features were used, the performance gains were particularly noticeable, with a statistically significant improvement. The greatest improvement was obtained in the range between 100~200 features for both logic and non-logic cases. By contrast, in both cases, beginning from the point of around 2000 features, 80 percent of the feature space, the performance starts to deteriorate in both cases. We conjecture that the reason for this shortcoming arises from the fact that ideal feature space should cover all the most representative features of the class, but the real-world feature space only approximates it. It is so-called bias-variance of classification. The small size of features helps lessen the suffering of the error associated with the variance, while the use of too many features makes the classification very inaccurate, sometimes even resulting in poor overall performance.

The results in Figure 6 show that an initial improvement occurs as the number of features is increased, and then a decrease in the accuracy as the number of features is increased too much. It is surprising that the greatest improvement can be obtained when 100 features are chosen for classification. One possible explanation is that the most distinguished features for the class could be contained in the top 100 features. For this reason, selecting a larger number of features for classification does not provide a great benefit, and may even degrade performance.

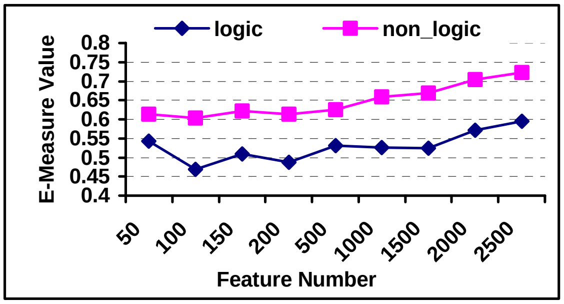

Figure 6: The average e-measure value for the effect of adding logic correlation between the classes

5.3 The Accuracy of a Hierarchical Model and Flat Model on Classification Performance

To compare the difference between the hierarchy and the flat classification models, we began with an equivalent number of features (e.g., 100 features in each class) for each classifier. As can be seen in Figure 7, in some cases the hierarchical model is slightly better (window parameter $\varepsilon \in (0.5, 0.7)$), while in others (window parameter $\varepsilon \leq 0.5$ or $\varepsilon \geq 0.7$) the flat approach wins. However, when an appropriate window parameter ε is chosen (e.g., $\varepsilon \in (0.6, 0.7)$), there is no apparent difference in overall classification performance. This may be due to the same feature spaces that we use for all the classes in both the hierarchical model and flat model. It suggests that classification performance is not the main reason we choose the hierarchical structure over the flat one.

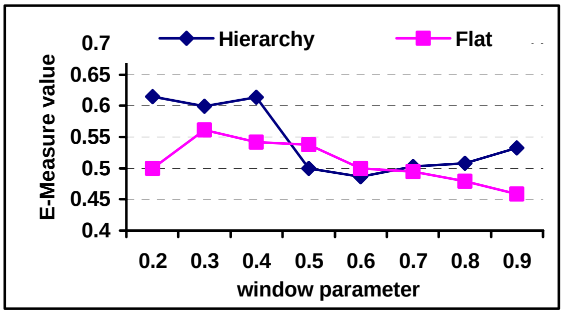

Figure 7: The average e-measure for the hierarchy and flat models

As far as computational expense and running time, the hierarchy approach shows promise for scaling to larger topic domains. Since the classification in the hierarchy scheme is a filter method, the hierarchical category search algorithm makes dramatic reductions in the class search space and consequently shortens running time as well. In Table 5 it can be seen that, in a hierarchical structure with 91 topic categories, the searched topics are 25.375 categories on average using the hierarchical model, which is only 28.2 percent of the whole search space used by the flat model. Besides, the running time for classification is 12 CPU seconds due to parallel computation, 34.2 percent of the time used for the flat model. In terms of training time for constructing a hierarchy of Naive Bayes classifiers, the running time with 2,850 training documents was 1 hour 23 minutes in comparison with 4 hours 56 minutes for the flat approach. The above advantages indicate that the hierarchical model is generally more robust in database classification.

Table 5: Classification statistics of the hierarchy and flat models

	Hierarchy Model	Flat Model
Topic Categories Searched	25.375	91
Running Time in Testing	12 (second)	35 (second)
Running Time in Training	1hour 23 minutes	4 hours 56 minutes

5.4 The Effects of Various Factors in Database Structure on Classification Performance

Considering the special characteristics of database classification are different from text document classification, we also investigated various factors (e.g., the similarity measurement methods employed, database size and distribution of database clusters) in database structure that affect classification performance.

5.4.1 The Similarity Measurement Methods Employed. Since each database is composed of documents with various topic categories, there are two similarity measure methods for database classification: *term-based* method and *document-based* method. In the *term-based* method the measure of similarity for the database S_i with respect to the class c_k (recall Equation 6) is based on the occurrence frequency of features of the class c_k in the database S_i, while in the *document-based* method, the similarity of the database S_i is based on the number of documents belonging to the class c_k. In practice, the *document-based* method is the variation of text document classification. We need to first classify each document in the database. The posterior probability $\Pr(c_k \mid d_i)$ can be computed by

$$\Pr(c_k \mid d_i) = \frac{\Pr(c_k)\Pr(d_i \mid c_k)}{\Pr(d_i)}$$

$$= \frac{\Pr(c_k)\sum_{t_i \in T_i}\Pr(t_i \mid c_k)w_{ij}}{\sum_{c_s \in C}\left(\Pr(c_s)\sum_{t_i \in T_I}\Pr(t_i|c_s)w_{ij}\right)} \qquad (12)$$

where, if the term t_i occurs in the feature space F^C of class c_k, that is, $t_i = f_j$ $(f_j \in F^C)$, $\Pr(t_i \mid c_k)$ in fact equates to $\Pr(f_j \mid c_k)$; otherwise $\Pr(t_i \mid c_k) = 0$; w_{ij} is the word weight of the term t_i in the document d_i (recall Definition 6). Therefore, the number of topic categories in the database S_i is equal to the number of topic categories of the documents in database S_i.

During our experiment we ran each method with various feature sets. The performance variations are shown in Figure 8. It was easily noted that the *term-based* method performed better than the *document-based* method, with an average of 5.05 percent performance improvement. The best-performance feature-number point for both methods was the 100-feature point, with the E-Measure values 0.497 and 0.527, respectively. This implicates that, for huge documents (note that in the *term-based* method, the database S_i is in fact assumed as a single large document with hundreds of thousands of words), Bayesian classifiers still have similar, or even better, classification performance than ordinary documents.

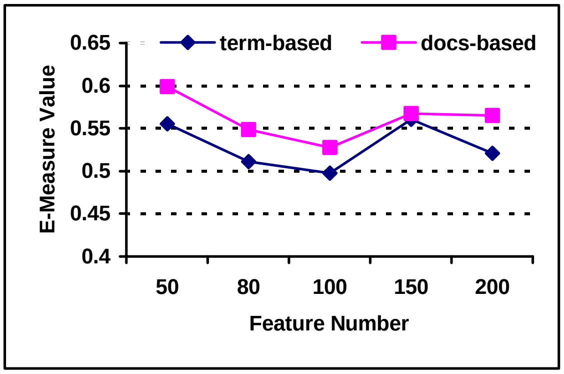

Figure 8: The average e-measure for different similarity measurement methods employed

As for time cost and resource consumption, the *term-based* method is far better than the *document-based* method since the database S_i is only regarded as a single large document. The rough estimation of running time for the *term-based* method is about 15.2 seconds in comparison to 315.5 seconds for the *document-based* method. According to the above observations, we deduce that the *term-based* method provides a more applicable and effective measurement for database classification.

5.4.2 The Effect of Database Size on Classification Performance. Database size is one of the important factors for database structure. We ran the experiments with various-size databases (see Table 4 for more information about the test databases).

The first thing we noticed in Figure 9 was the difference of the best-performance feature number for various-size databases. The best-performance feature number increases with the database size. A possible reason is that large-scale databases contain a large number of documents with more topic categories (recall Table 4). It means that the classifiers in the hierarchy need more features to better distinguish various classes in the database.

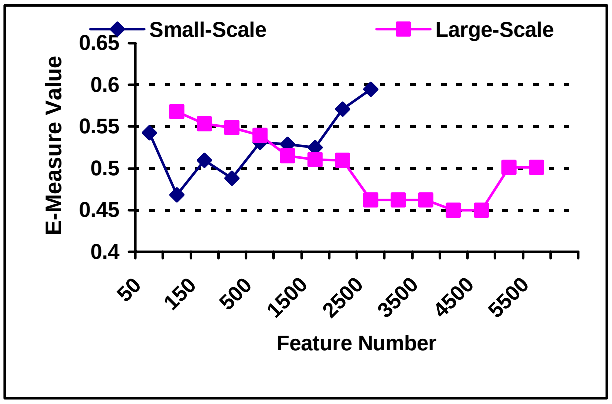

Figure 9: The average e-measure for the database size

The second thing we noticed, as described in Section 5.2, is the increase of the feature number tends to improve classification performance, but there are limits to that trend. In fact, beyond a certain point, the use of more features will lead to a diminished performance. Similar experimental results can be observed here. In both small-scale and large-scale database cases, performances seem to approach an arch. For small-scale databases, the best-performance feature number point is usually located at the range between 80-150 features, while the best-performance feature number for large-scale databases is at the range between 2,500-4,500 features.

5.4.3 The Effect of the Distribution of Database Clusters. As mentioned previously, in the real world there exists two different types of databases, special-purpose databases and general-purpose databases. The main distinction between them is the distribution of subject domains in the databases. To determine the extent to which the distribution of topic categories affects classification performance, we conducted a number of experiments shown in Table 6 on these two types of large-scale test databases.

In Figure 10 we noted that Bayesian classifiers perform better in general-purpose databases than in special-purpose databases in terms of capturing the distinctions between relevance and irrelevance for available topics. In the range between 3,500-4,500 features, the performances for both types of databases are very close. However, in other range of features, the classification accuracy for general-purpose databases significantly outperformed that for special-purpose databases. In some feature ranges (e.g., the range between 1,000-2,000 features), the performance gap even reaches 31.2 percent.

Table 6: Statistics about the test dataset for Reuters databases

	Reuter Dataset Test (Special-Purpose databases)	Reuter Dataset Test (General-Purpose databases)
Number of databases	12	12
The documents per database	1,000~2,500	1,000~2,500
Mean classes per database	7	38
Mean words per database	201,062	201,062

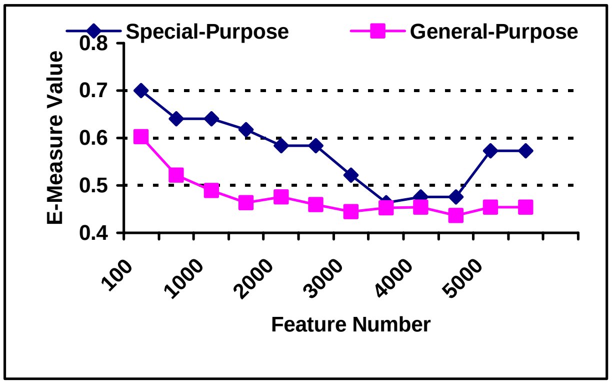

Figure 10: The average e-measure for the distribution of database clusters

A primary cause of this phenomenon is the problem of classification errors in the hierarchy. In Section 3.2.1 we discussed that, due to the trade-off between computation cost and classification accuracy, the feature space F of a class c_k only contains those most representative features for class c_k. The information loss in the feature space results in classification errors in the classification process. As described in Section 3.2.2, the features space F^C for the cluster C is the combination of the feature spaces of a subset of child classes in the lower level (recall Definition 1). Classification error for cluster C is also the accumulation of all classification errors of the child classes. When classification error of a class reaches beyond-error tolerance, the classification performance will rapidly degrade. Since general-purpose databases contain the documents with more topic domains than special-purpose databases, they provide more category room, which helps alleviate to some extent the problem of classification errors in the hierarchy.

6 Conclusions and Future Work

With the proliferation of online information resources on the Internet, the problem of database selection has become a challenging issue for searching multiple, distributed web databases. The work described in our paper explores the use of hierarchical structure in order to resolve the difficulty. Within a probabilistic framework, our hierarchical classification approach takes the logical relationships between the classes in the hierarchy and the special characteristics of databases into consideration. The experimental results have proved that this approach is effective and drastically reduces search space, while also helping to improve accuracy in many cases. Moreover, we presented a new category assignment strategy that could assign more "winning" topic categories into the database thus outperforming two other common assignment approaches on classification effectiveness.

Our future work on this research topic includes the following tasks. First, we wish to construct more accuracy models by using an optimal Bayesian network learning algorithm, which will allow us to obtain further advantages in efficiency in the hierarchical approach. Second, we will further investigate the work on feature selection, especially how to effectively discriminate the features of topic categories at higher levels in the hierarchy. In this paper we simply combine the feature spaces of child classes at lower levels, which we expect to improve in future work. More importantly, we intend to apply this approach to a wider variety of text datasets. Our present work is conducted only on small hierarchies of topics extracted from the Reuters datasets. We hope that these techniques can also effectively tackle complex problems in a vast web environment.

Acknowledgement

This research was supported by a large grant from the Australian Research Council under contract DP0211282. We would also like to thank the anonymous reviewers for many valuable comments and suggestions that have improved the quality of this paper.

References

[1] C. Apte, F. Damerau, and S. M. Weiss, "Automated Learning of Decision Rules Text Categorization," *ACM Transactions on Information System*, 12(3):231-251, July, 1994.

[2] R. A. Botafogo, "Cluster Analysis for Hypertext Systems," Proc. Of the 16th Annual International ACM/SIGIR Conference on Research and Development in Information Retrieval, Pittsburgh, PA, pp. 116-125, 1993.

[3] R. Burgin, "The Retrieval Effectiveness of Five Clustering Algorithms as a Function of Indexing Exhaustiveness," *Journal of the American Society for Information Science*, 46(8):562-572, 1985.

[4] H. Chen, P. Hsu, R. Orwig, L. Hoopes, and J. F. Nunamaker, "Automatic Concept Classification of Text from Electronic Meetings," *Communication of the ACM*, 37(10):56-73, October, 1994.

[5] H. Chen, C. Schuffels, and R. Orwig, "Internet Categorization and Search: A Self-Organizing Approach," *Journal of Visual Communication and Image Representation*, 7(1):88-102, 1996.

[6] S. Deerwester, S. Dumais, G. Furnas, T. Landauer, and R. Harshman, "Indexing by Latent Semantic Analysis," *Journal of the American Society for Information Science*, 41(6):391-407, 1990.

[7] S. T. Dumais, "Latent Semantic Indexing (LSI) and TREC-2)," Proc. Of the Second Text Retrieval Conference (TREC-2), Gaithersberg, Maryland, pp. 105-115, 1993.

[8] A. EI-Hamdwchi, P. Willett, "Comparison of Hierarchical Agglomerative Clustering Methods for Document Retrieval Systems," *Journal of Information Science*, 13:361-365, 1989.

[9] B. Everitt, *Cluster Analysis*, Second Edition, Heinemann Educational Books, London, England, 1980.

[10] P. H. Fisher, "Knowledge Acquisition via Incremental Conceptual Clustering," *Machine Learning*, 2:139-172, 1987.

[11] S. Gauch, G, Wang and M. Gomez, "Profusion: Intelligent Fusion from Multiple, Distributed Search Engines," *Journal of Universal Computer Science,* 2(9)637-649, Sept. 1996.

[12] A. Griffiths, L. A. Robinson, and P. Willett, "Hierarchic Agglomerative Clustering Methods for Automatic Document Classification," *Journal of Documentation*, 40(3):175-205, 1984.

[13] W. R. Hersh, C. Buckley, T. J. Lenone and D. H. Hickam, "OHSUMED: An Interactive Retrieval Evaluation and New Large Test Collection for Research", Proc. the 17th Annual International ACM/SIGIR Conference on Research and Development in Information Retrieval, Dublin, Ireland, pp. 192-201, 1994.

[14] W. Huang and R. Lippman, "Network Net and Conventional Classifier," Proc. IEEE Conference on Neural Information Processing System-Natural and Synthetic, Boulder, CO, November 1987.

[15] P. G. Ipeirotis, L. Gravano, M. Sahami, "Probe, Count, and Classify: Categorizing Hidden Web Databases," Proc. of the 2001 ACM SIGMOD International Conference on Management of Data, Santa Barbara, California, United States, pp. 67 – 78, 2001.

[16] D. D. Lewis, "Naive Bayes at Forty: The Independence Assumption in Information Retrieval," Proc. of EDML-98, Chemnitz, Germany, 1998.

[17] D. D. Lewis, R. E. Schapire, J. P. Callen, and R. Papka, "Training algorithms for linear text classifiers," Proc. of the 19th Annual International ACM/SIGIR Conference

on Research and Development in Information Retrieval, Zurich, Switzerland, pp. 298-306, 1996.

[18] D. D. Lewis, Reuter-21578 Text Categorization Test Collection Distribution 10, available at http://www.research.att.com/~lewis.

[19] W. Meng, W. Wang, H. Sun, and C. Yu, "Concept Hierarchy Based Text Database Categorization," *Journal of Knowledge and Information Systems*, 4(2):132-150, 2002.

[20] G. W. Milligan and M. C. Cooper, "Methodology Review: Clustering Methods," *Applied Psychological Measurement*, 11:329-354, 1987.

[21] J. Pearl, *Probabilistic Reasoning in Intelligent Systems*, Morgan Kaufmann, 1988.

[22] M. Porter, "An Algorithm for Suffix Stripping," *Program*, 14(3):130-137, 1980.

[23] G. Salton and M. McGill, *Introduction of Modern Information Retrieval*, McGraw-Hill, 1983.

[24] H. Schutze, D. A. Hull, and J. O. Pedersen, "A Comparison of Classifiers and Documents Representation for the Routing Problem," Proc. of the 18th Annual International ACM/SIGIR Conference on Research and Development in Information Retrieval, Washington, pp. 229-237, Seattle, 1995.

[25] P. Willette, "Recent Trends in Hierarchic Document Clustering: A Critical Review," J*ournal of Information Processing & Management*, 24(5):577-597, 1988.

[26] H. Yang and M. Zhang, "Query Expansion with Naive Bayes for Searching Distributed Collections," Proc. of the 11th International Conference on Intelligent Systems, Boston, MA, pp. 17-23, 2002.

Hui Yang is currently a Ph.D. student at the University of Wollongong, Australia. She received her Bachelor's degree in Electronics from Huazhong University of Science and Technology, China, and her Master's degree in Computer Science from the University of Wollongong in 2001. Her current research interests are in the areas of distributed information retrieval, intelligent information system and data mining.

Minjie Zhang is an Associate Professor of Computer Science and Software Engineering at the University of Wollongong, Australia. She received her Bachelor of Science degree from Fudan University, China in 1982 and her Ph.D. degree from the University of New England, Australia in 1996. She is a member of IEEE, IEEE Computer Society, and ISCA. She is the author or co-author of over 40 research papers. Her research interests include distributed artificial intelligence, distributed information retrieval, and agent-based software engineering.