Hierarchical Classification for Multiple, Distributed Web Databases

Hui Yang and Minjie Zhang
School of Information Technology & Computer Science
University of Wollongong
Wollongong, NSW 2522, Australia
hy92@uow.edu.au minjie@uow.edu.au

Abstract
The proliferation of online information resources increases the importance of effective and efficient distributed searching. This 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 web 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 here demonstrate that such a classification approach not only effectively reduces the class search space, but also helps to significantly improve accuracy on classification performance.

Keywords: 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 services such as Yahoo, AltaVista, web users still feel frustrated in profitably utilized such large amount 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, an automatic and robust method of partitioning multiple, distributed web databases based on their subject content into classification schemes, which is also called database classification, will be helpful for efficient and fruitful database selection.

There exist quite a number of special – purpose databases which focus on documents in confined subject domains such as IEEE, ACM digital library databases on the Internet, while there also exist 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, the use of a hierarchy can decompose the classification task into a set of smaller tasks, which only correspond to small splits in the hierarchical tree. Therefore, it makes the accomplishment of the classification work more effective and efficient.

Recently several researchers have investigated the use of hierarchies for text classification and have gotten encouraging achievements [7]. Our work differs from earlier work in the following important aspects. First, during the construction of the structured hierarchy of topics, unlike other hierarchical classification methods which only treat the classes (categories) in the hierarchy as a simple vertical parent - child relationship between different levels, our hierarchical structure considers the horizontal logical relationships among the classes at the same level, which can better express the correlation of the classes. Second, we use a variation of probabilistic Bayesian model based on Naïve Bayes learning techniques [2] for automatic database classification. This model takes the special characteristics of database 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 for the databases. Our experimental results reported here have demonstrated that our hierarchical classification methods can significantly improve the performance of database classification.

2 Probabilistic Framework for Hierarchical Classification
In this paper, we 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.

2.1 Hierarchical structure of topics for databases
Hierarchical structure of topics has long been used in special – purpose collection of documents. More recently, several large-scale Internet search engines such as Yahoo, Infoseek have also adopted such hierarchies to manage the World Wide Web so as to conveniently guide users to the appropriate topic which they are interested in. It is sensible for us 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.

Now, we first describe the structured hierarchy of topics. It is a topic directory which a hierarchical architecture of 3 layers shown as Figure 1. The tree hierarchy contains nodes at different levels indicating the topics of interests to the users. It is easy to see that topics in the hierarchical topic structure are ordered from more general broad topics (top) to more narrow ones (bottom), and the leaf nodes point to specific and unambiguous subjects.

The hierarchical structure treats the topics as a collection of topic sets, each of which only consists of a small set of topics. As we see from Figure 1, each node (except root node and leaf nodes) in the hierarchy actually acts as two roles: it is both 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 all parent nodes, each node has a classifier that distinguishes databases about one class from databases about another class. Since the classifier at a node need only to focus on a small set of topics rather than the overall topics, it is possible to make the classification more accurate. Therefore, we can say, 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 accurate as well.

2.2 Bayesian Classifiers

2.2.1 Construction of Naive Bayes Classifiers

A training set of labelled feature vectors is used to induce a classification model. This model is then used to predict the class label for a set of previously unseen data instance (e.g. web databases). Optimistically, the feature vector will fully determine the appropriate classification. However, as we know, 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. So we use a probability distribution to model the classification function. Formally, for each assignment of values \( f_i \) to \( F \), we have a probability \( P(c_i | f_i) \) in the possible cluster \( C \), where \( c_i \in C \), and \( C \) is a set of classes denoted as \( C = \{c_1, c_2, \ldots, c_M\} \); \( f_i \in F \), and \( F \) is a set of features described as \( F = \{f_1, f_2, \ldots, f_N\} \); \( P(c_i | f_i) \) is the probabilistic distributions for each feature \( f_i (f_i \in F) \) in class \( c_i \). This ensures that in the class network, each feature will be taken into account for the classification. A learning algorithm utilizes the feature information obtained from the training set using the LSI method [1] to construct a classifier.

During the 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 allow additional edges between the classes, which capture logical correlation among the classes shown as Figure 2. The basic intuition underlying our approach is that it is common that one document contains the content of multiple topics rather than a single one. Thus, the relationships of these topics are closer than 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). Then, a line is drawn between them denoting “AND” connection. Note that the process of adding these edges may involve in a heuristic search on a hierarchical structure.
Like most agglomerative clustering methods [7], our hierarchical clustering algorithm constructs a cluster hierarchy from the bottom to the top by merging some subject-relevant classes at a time. At the beginning, just the topics of the leaf level in the hierarchy associate with labelled training documents as the classes. For each leaf-level topic (class), it consists of a set of the most important features with the parameters \( \Pr(f_i | c_k) \).

Then the last-second-level topics are used as class labels. Note that every node in the last-second-level 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. Here, we will only present our method for constructing the classifiers at the last-second level. The constructions of the classifiers at other higher levels are implemented in the similar way.

**Definition 1.** Cluster \( C \) is a superclass of a number of classes presented as \( C = \{c_1, c_2, \ldots, c_u\} \), where \( c_i \) (\( 1 \leq i \leq M \)) is a class of cluster \( C \). \( F^C \) is the feature space for the cluster \( C \), which is described as \( F^C = \{f_1, f_2, \ldots, f_y\} \), where \( F^C \) is the feature space set of all the classes, namely, \( F^C = F_1 \cup F_2 \cup \ldots \cup F_u \), \( F_k \) corresponds to the feature space of class \( c_i \), and \( f_i (1 \leq i \leq N) \) is a distinct feature vector in this feature space \( F^C \).

**Definition 2.** For the given cluster \( C \), \( R(c_j,c_i) \) is a logic connection function for the classes, \( c_j \) and \( c_i \) (\( 1 \leq j, k \leq M, j \neq k \)).

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

\[
\begin{align*}
&1. \text{ If } \forall f_j \in F^C (1 \leq j \leq M, j \neq k), f_i \notin F_j, \text{ then } \\
&\quad \Pr(f_i | C) = \Pr(f_i | c_k) \quad (1) \\
&2. \text{ If } \exists f_j \in F^C (1 \leq j \leq M, j \neq k), f_i \notin F_j \cap F_k, \text{ then } \\
&\quad \Pr(f_i | C) = \frac{\max(\Pr(f_i | c_j), \Pr(f_i | c_k)) \cdot R(c_j, c_k)}{\max(\Pr(f_i | c_j), \Pr(f_i | c_k))} \text{ otherwise} \quad (2)
\end{align*}
\]

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

\[
\Pr(C) = \frac{1 + \sum_{c_i} \Pr(c_i)}{M + N} \quad (3)
\]

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 the cluster \( C \) is the combination of all the feature space of all the classes, the size of the feature will be possibly very huge. Therefore 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 \( \Pr(f_i | C) \) in that such features are generally not able to improve the classification accuracy. Finally, for each last-second-level class, we construct a separate classifier on the appropriate reduced feature set.

### 2.3 Hierarchical Classification Strategies

#### 2.3.1 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, which can be described as \( C_i = \{c_1, c_2, \ldots, c_M\} ; D_i \) 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 database \( S_i \) has \( s \) distinct terms, namely, \( T_i = \{t_1, t_2, \ldots, t_s\} \). Each term in the database can be represented as a 2-dimension vector \( \{t_j, w_j\} \) (\( 1 \leq j \leq s \)), where \( t_j \) is the term (word) occurring in 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, which can be got by the famous function tf*idf [6].

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. At first, the database \( S_i \) is classified in this hierarchy by filtering the irrelevant subject domains at the first level with the root classifier. Then the hierarchy scheme sends the database down to 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.
At each level (except the root level), once one class is chosen by the classifier as the most likely topic for the database \( S_i \), it will be automatically added into the class set \( C \) of database \( S_i \). Clearly, the database that belongs to a class (a leaf node in the hierarchy tree) is also assumed to belong to each of nodes (classes) along the path to the root.

2.3.2 Category Search Strategy

In each level in the hierarchy, the category search strategy will be executed with the following steps:

1. Firstly, calculate the posterior probability \( \Pr(c_i \mid S_j) \) of class \( c_i \) for the database \( S_j \), where \( c_i \) is a class of the chosen cluster \( C \) in the higher level.
2. Rank all the classes in the chosen cluster \( C \) based on the posterior probability \( \Pr(c_i \mid S_j) \).
3. Assign the most likely categories to the database \( S_j \) 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 likeness for the database \( S_j \) is the posterior probability \( \Pr(c_i \mid S_j) \). Given the parameters, \( \Pr(f_j \mid c_i) \) and \( \Pr(c_i) \), the posterior probability \( \Pr(c_i \mid S_j) \) can be determined by Bayes’ rule and the occurrence frequency of features of the class \( c_i \) in the database \( S_j \):

\[
\Pr(c_i \mid S_j) = \frac{\Pr(c_i) \Pr(S_j \mid c_i)}{\Pr(S_j)}
\]

\[
\Pr(c_i) = \frac{\Pr(c_i) \sum_{t \in S_j} \Pr(t \mid c_i) w_t}{\sum_{c_j} \Pr(c_j) \sum_{t \in S_j} \Pr(t \mid c_j) w_t}
\]

where, if the term \( t_i \) occurs in the feature space \( F^c \) of class \( c_i \), that is, \( t_i = f_j \) (\( f_j \in F^c \)), \( \Pr(t_i \mid c_i) \) in fact equates to \( \Pr(f_j \mid c_i) \); otherwise, \( \Pr(t_i \mid c_i) = 0 \); \( w_t \) is the word weight of the term \( t_i \) (recall Definition 4).

2.3.3 Category Assignment Strategy

In Step 2 and 3 at Subsection 2.3.2, it is usual to make the ranking \( C \) based on the posterior probability \( \Pr(c_i \mid S_j) \).

According to the category ranking, one or more categories are assigned to the database \( S_j \) using the categories assignment strategy. Although many category assignment methods have been proposed [5] such as the probability threshold method which assigns all the categories with the likeliness value over a predefined threshold \( \tau \). One problem of the above method is that due to the difference of the number of classes in each level of a hierarchy tree, a simple \( \tau \) cannot correctly reflect the proper class number in different level.

In this research, we propose a new approach to select the “winning” classes in the cluster \( C \). Consider such a scenario where for some web databases, especially in those large-scale general-purpose web databases, the distributions 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 the categories as many as possible. This possibility-window method is extended from the possibility-threshold method. The window is defined as follows:

\[
\frac{\Pr\text{\tiny win}(C \mid S_j)}{\Pr(c_i \mid S_j)} \leq 1 + \epsilon \quad (1 \leq k \leq M)
\]

\[
\frac{\Pr(c_i \mid S_j)}{\Pr\text{\tiny win}(C \mid S_j)} \geq 1 - \epsilon
\]

where \( \Pr\text{\tiny win}(C \mid S_j) \) is the maximum of the posterior probabilities of all the classes in the cluster \( C \), and \( \epsilon \) is a window parameter. As long as the posterior probability of category \( c_i \) satisfies all of the above conditions, category \( c_i \) will be chosen as the appropriate category for the database \( S_j \).

3 Experimental Evaluation Measures

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 [3], and web data set, which are articles downloaded from some online newsgroups and web sites. Although those topic categories have not been organized with a hierarchical structure, we still use the label information to construct a 3-layer hierarchy.

The effectiveness of database classification can be measured as recall \( R \) and precision \( P \). To combine precision and recall into one number, \( E \)-Measure is commonly used:

\[
E = 1 - \frac{2}{1 + \frac{1}{P} + \frac{1}{R}}
\]

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”, \( E = 0 \).

4 Experiment Results and Discussions

4.1 Accuracy for Different Category Assignment methods

First, we compare different category assignment methods with probability-window, probability-threshold in term of selection accuracy. It is interesting to see how the variety of window parameter \( \epsilon \) and threshold parameter \( \tau \) affect
the classification performance. To see it more clearly, we plot the accuracy curve in Figure 3.

In order to allow a fair comparison, we ran experiments over the Reuters databases with the varying values of parameters $\varepsilon$ and $\tau$ ranging from 0.2 to 1, respectively. As we note, for high value of window parameter $\varepsilon$, the classification scheme will have the databases assigned to more leaf nodes (topics), which might lead to high recall. But, as mentioned previously, it arises the shortcoming of low precision. That these characteristics might affect the classification performance is not surprising, and the same would be expected for the probability-threshold 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 $\varepsilon$ can have a large impact on classification accuracy. When $\varepsilon$ varies the range between 0.5 and 0.8, the results are statistically significant improvements over the accuracy on the original testing dataset. By contrast, the probability-threshold virtually incapable of taking advantage of the hierarchical structure.

![Figure 3: The average E-Measure value of the category assignment methods. (a) probability-window ($\varepsilon$ ranges from 0.2 to 1). (b) probability-threshold ($\tau$ ranges from 0.2 to 0.8)](image)

4.2 The Effect of Logical Correlations between the Classes.

We are 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 the selection effectiveness. The classification results in Figure 4 show that when the right logical relationships are used for classification, classification performance increases, by 9.75% on average. It suggests that the common features occurring in the related classes affect classification effectiveness to some extent. We note that the method of adding logical relationship between the classes significantly outperforms the non-logical method regardless of the number of features used.

5 Conclusions

The work described in this paper explores the use of hierarchical structure in order to facilitate 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. Moreover, we present a new category assignment strategy that can assign more “winning” topic categories into the database, thus outperforming other two common assignment approaches on classification effectiveness. There are a number of research directions for future work. For example, we wish to construct more accuracy models by using an optimal Bayesian network learning algorithm, which allow us to obtain further advantages in efficiency in the hierarchical approach.

6 Acknowledgement

This research was supported by a large grant from the Australian Research Council under contract DP0211282.

7 References