Online Information Review

Dynamic and hierarchical classification of Web pages

Ben Choi
Assistant Professor in Computer Science, at the College of Engineering and Science, Louisiana Tech University, Ruston, Louisiana, USA
Xiaogang Peng
PhD student in Computational Analysis and Modelling, at the College of Engineering and Science, Louisiana Tech University, Ruston, Louisiana, USA

Online Information Review, Vol. 28 No. 2, 2004,

Dynamic and hierarchical classification of Web pages

Ben Choi and Xiaogang Peng

The authors

Ben Choi is an Assistant Professor in Computer Science and **Xiaogang Peng** is a PhD student in Computational Analysis and Modelling, both at the College of Engineering and Science, Louisiana Tech University, Ruston, Louisiana, USA.

Keywords

Internet, Learning, Classification, Information retrieval, Data handling

Abstract

Automatic classification of Web pages is an effective way to organise the vast amount of information and to assist in retrieving relevant information from the Internet. Although many automatic classification systems have been proposed, most of them ignore the conflict between the fixed number of categories and the growing number of Web pages being added into the systems. They also require searching through all existing categories to make any classification. This article proposes a dynamic and hierarchical classification system that is capable of adding new categories as required, organising the Web pages into a tree structure, and classifying Web pages by searching through only one path of the tree. The proposed single-path search technique reduces the search complexity from θ(n) to θ(log(n)). Test results show that the system improves the accuracy of classification by 6 percent in comparison to related systems. The dynamic-category expansion technique also achieves satisfying results for adding new categories into the system as required.

Electronic access

The Emerald Research Register for this journal is available at
www.emeraldinsight.com/researchregister

The current issue and full text archive of this journal is available at
www.emeraldinsight.com/1468-4527.htm

Introduction

The Internet is growing at a great speed but the documents in the Web are not logically organised which inevitably makes their manipulation and retrieval difficult. The need for mechanisms to assist in organising and locating relevant information is becoming more urgent. One of the solutions is provided by classified directories (Chekuri *et al.*, 1997). However, current systems, such as Yahoo! (2003) still require human labour in doing the classification. It is doubtful that manual classification is able to keep up with the growth of the Web. First, manual classification is slow and costly since it relies on skilled manpower. Second, consistency of categorisation is hard to maintain since different human experiences are involved. Finally, the task of defining the categories is difficult and subjective since new categories emerge continuously from many domains. Considering all these problems, the need for automatic classification becomes much more crucial.

Advantages of our new system

Our automatic classification system, unlike others, stresses the dynamic growth issue of the Internet. As the number of Web pages on the Internet increases continuously at great speed, it is impossible for a fixed category set to provide accurate classification. To address this problem, we proposed and implemented a dynamic expanding technique. Our dynamic-category expansion method has the functionality of adding new categories automatically to allow for the growth of the Internet.

Moreover, we proposed a single-path search algorithm that takes advantage of the hierarchical structure of the classification system and results in improving the classification accuracy and also in reducing the computational complexity. When classifying a Web page, our single-path algorithm searches a path from the root to a leaf of our classification tree. It increases the accuracy of classification by 6 percent and reduces the computational complexity from θ(n) to θ(log (n)) in typical cases, compared to other existing methods.

Organisation of the paper

This article is organised into seven sections. The next section provides related research in text learning and document classification. The third section provides an overview of our classification system. The fourth section provides details on how to build our initial hierarchical category

Refereed article received 30 September 2003
Accepted for publication 17 November 2003
This research was supported in part by the Centre for Entrepreneurship and Information Technology (CEnIT), Louisiana Tech University, Grant iCSe 200123.

Online Information Review
Volume 28 · Number 2 · 2004 · pp. 139-147
 · ISSN 1468-4527
DOI 10.1108/14684520410531673

Online Information Review
Volume 28 · Number 2 · 2004 · 139-147

tree. The fifth section shows how to classify a Web page using our single-path search technique and how to automatically add a new category into our category tree. The sixth section reports the performance analysis and the test results to validate our implemented classification system. The final section provides the conclusion and future research.

Related research

Our work relates to text learning and document classification. Text learning techniques are used to extract key information from documents, in our case, Web pages. The extracted information is used to represent a document or a category. We also review various document classification techniques that form the basis for our Web page classification.

To represent a document or a category in a concise way, text learning techniques are used to abstract key information from the documents. A simple but limited document representation is the bag-of-words technique (Koller and Sahami, 1998; Lang, 1995). To represent a given document, it simply extracts key words from the document and uses those words as the representation of that document. To make the representation concise, many common words (also called stop words), such as pronouns, conjunctions and articles, are not included in the representation.

Various derivatives of the bag-of-words technique have also been proposed. For example, Mladenic and Grobelnik (1998a, b) extend the bag-of-words to the bag-of-phrases, which was shown by Chan (1999) to be a better choice than using single words. Experiments show that a phrase consisting of two to three words is sufficient in most classification systems.

Another extension is to associate each phase (or term) with a weight that indicates the number of occurrences of the phase in a document (Salton and Buckley, 1987). To increase the accuracy of counting the occurrences, many forms of a word, such as plural or past tense, are considered the same as the original word, using a process called stemming. Each phrase together with its associated weight is considered as a feature of the document. All the extracted features of a document are grouped to form a vector called "feature vector" representation of the document.

One way to represent a category is by using the similar vector representation as described for a document. In this case, a set of training documents for a category is given. Text learning techniques extract the common terms among the documents and use those terms to form a vector representation of the category. One such technique is called term frequency inverse document frequency (TFIDF) (Salton and Buckley, 1987). TFIDF representation extends the feature vector concept further to account for the number of occurrences of a term in all training documents. It represents each category as a vector of terms that are abstracted from all training documents. For each training document Dj is represented by a vector Vj. Each element of the vector Vj is a product of the term frequency TF(Wi, Dj) and the inverse document frequency IDF (Wi). Where TF(Wi, Dj) is the number of occurrences of the term Wi in the document Dj. IDF(Wi) is the product of the total number of training documents K and the inverse of DF(Wi) that is the number of documents containing the term Wi. That is:

$$IDF(Wi) = \frac{K}{DF(Wi)}.$$

Log(K/DF(Wi)) is often used instead of simple product. A single vector is formed by combining all the vector Vj where j ranges 1 to K. Each element of the single vector is the average value of all the corresponding elements in Vj (for j from 1 to K).

Other more sophisticated techniques are available such as PrTFIDF (Joachims, 1997). Joachims extended the TFIDF representation into probabilistic setting by combining probabilistic techniques into the simple TFIDF. He proposed the PrTFIDF having the following formula:

$$P(d|c_j) = \sum_{w \in (d \cap c_j)} \frac{P(w|c_j) * P(c_j)}{\sum_{c \in C} P(w|c) * P(c)} * P(w|d), \quad (1)$$

where c and c_j are categories taken from a category set C, $P(d|c_j)$ is the probability for a document d given a category c_j, and $P(w|d)$ is the probability of a feature w given the document d. The PrTFIDF classifier optimises the parameter selection in TFIDF and reduces the error rate in five out of six reported experiments by 40 percent.

Once each category is represented by a vector and a document is also represented by a vector, the document is classified by comparing the vector of the document to the vector of each category. Dot product between vectors is usually used in the comparison. The result of the dot product is a value that is used to measure the similarity between the document and a category. The document is assigned to the category that results in the highest similarity among all the categories. Other more sophisticated classification algorithms and models were proposed including: multivariate regression models (Fuhr *et al.*, 1991; Schutze *et al.*, 1995), nearest neighbour classifiers (Yang and Pedersen, 1997), Bayesian classifiers (Lewis and Ringuette, 1994), decision tree (Lewis and Ringuette, 1994), Support Vector Machines (Dumais and Chen, 2000; Joachims, 1998), voted classification (Weiss *et al.*, 1999), and Page

Online Information Review
Volume 28 · Number 2 · 2004 · 139-147

and Link Page Classifier (Choi and Guo, 2003). Tree structures appear in all of these systems. Some proposed systems focus on classification algorithms to improve the accuracy of assigning documents to catalogues (Joachims, 1997), while others take the classification structure into account (Koller and Sahami, 1998). Our classifier goes even further by allowing the dynamic expansion of the classification structure as described in the following sections.

Our dynamic and hierarchical classification system

Our classification system consists of two stages. Stage one is used to prepare a special category tree that makes it possible for our single-path search algorithm. Stage two is used to classify a new page and to add a new category if needed.

Stage one of our system will generate an initial category tree. The first step is to choose a set of pre-defined categories. Next, text training methods are used to create an initial presentation for each category. All categories are arranged into hierarchical tree structure. The features in the initial presentation of each category are propagated upward from leaf nodes to the root node of the tree. This step insures a hierarchical arrangement of features and makes it possible for our single path search algorithm. To reduce computational time, some of the features are removed in the feature selection step. Each of these steps is described in detail in the following section.

After the special category tree is created, we are ready to classify a Web page. The first step is to create a representation for the Web page. This representation is compared with category representations that are located on a breadth-first search path from the root node to a leaf node of the category tree. The Web page is assigned to the category that has the highest similarity if no new category needed to be added. If certain conditions are met, then a new category is added and the new page is assigned to the new category. After the new page is assigned to a category, the features contained in the page are incorporated into the category. The newly added features are propagated upward from the current category all the way up the root node. This again ensures the hierarchical arrangement of features and makes it possible for our single path search algorithm. Each of these steps is described in detail after the following section.

Generating the initial category tree

Before any classifying can be done, we need to prepare a classification system. For our proposed system, the processing includes choosing pre-defined categories, organising the categories into a hierarchical tree structure, creating a representation for each of the categories, propagating the representation upward, thus creating a hierarchically arranged representation structure, and selecting parts of the representation that are relevant for classification. This stage consists of the following steps:

(1) getting a pre-defined category tree;
(2) creating a feature vector for each category;
(3) propagating features upward; and
(4) feature selection.

Getting a pre-defined category tree

Before any classification can be done, pre-defined categories need to be chosen. We choose to use the categories and category arrangement defined by Yahoo.com although other arrangements can also be used for our classification system. Yahoo! employs people to classify Web pages into categories that are organised in a hierarchical tree structure, such as arts/performing_arts/theatre/musicals, where "performing arts" is a sub-category "arts".

Creating a feature vector for each category

After we have chosen the pre-defined categories, we need to encode them in such a way that can be used by our classification system. We represent each category by a feature vector that consists of a list of features and their associated weights. Each feature for our system is a phrase consisting of one, two, or three consecutive words. In the process of extracting features from training documents, we ensure that a phrase cannot consist of words from two different sentences or from two different parts of text such as the title and the body.

To increase the accuracy of representing a category, we extract features from several sources, including categories and site listings provided by Yahoo.com, Web pages contained in the category, and the URLs of those Web pages. For each category, Yahoo.com provides a listing of sub-categories and/or a listing of sites, which is shown by Labrou and Finin (1999) to be one of the best sources for extracting features to represent a category. A Web page is divided into parts, such as title, first paragraph, and words enclosed by H1 HTML tags. Table I provides a list of the parts and their relative weights used in our system. Some sentences are considered more important for describing the context of the page, such as sentences containing phrases "the purpose of" or "the main aim of" (Mladenic, 1998a, b, c). Each of the sources of features (listed in Table I) is then processed by removing stop words and by performing word stemming.

The number of occurrences of a feature w in a category C is counted to form the term frequency $TF(w, C)$ and is normalised using the following formula:

Online Information Review
Volume 28 · Number 2 · 2004 · 139-147

Table I Source of feature and their relative weights

Source of features	Weight
First paragraph	2
Last paragraph	2
Important sentences proposed by Paice (1990)	3
⟨title⟩	4
⟨H1⟩	2
⟨H2⟩	1
⟨EM⟩	2
⟨Strong⟩	3
⟨Meta Name = "keywords" or "descriptions"⟩	3
URL	4

$$P(w|C) = \frac{1 + TF(w, C)}{|F| + \sum_{f \in F} TF(f, C)},$$

where F is the set of all the features in current category C and |F| is the number of elements in set F.

Propagating features upward

To make our single-path search algorithm possible, we need to create a hierarchically organised category structure. The feature information of a category is propagated upward from leaf nodes to the root node of our classification tree. Features of a category are propagated upward to its parent category. The root node of the category tree contains all features of all its child notes. In general, any node in the category tree contains all features of all its child notes. The propagated feature list is generated by adding the original feature list of the current node and all the propagated feature lists of the sub-categories and by assigning different weights. By propagating in this way, the feature list captures the characteristics of a sub-tree rooted in the current node rather than in an isolated category set.

As a feature is propagated upward, its weight is reduced. We use formulas (2) and (3) proposed by Mladenic (1998a, b, c) to calculate the new weights. The feature propagation formula is:

$$P(w|T) = \sum_{i=1}^{k} P(w|SubTi) * P(SubTi|T) + P(w|N) * P(N|T). \quad (2)$$

The weight of a feature given a tree T that is rooted at node N and has k sub-trees is calculated using formula (2), where $P(SubT_i|T)$ and P(N|T) are calculated using formula (3). $SubT_i$ is a sub-tree i of the given tree T. For a node without child notes, $P(SubT_i|T) = 0$ and $P(N|T) = 1$. Ex(node) is the number of Web pages contained in the current node while Ex(SubTi) is the number of Web pages contained in the sub-tree i. The formulas for weight reduction are:

$$P(N|T) = \frac{\ln(1 + Ex(Node))}{\sum_{i=1}^{k} \ln(1 + Ex(SubTi)) + \ln(1 + Ex(Node))},$$

$$P(SubTi|T) = \frac{\ln(1 + Ex(SubTi))}{\sum_{i=1}^{k} \ln(1 + Ex(SubTi)) + \ln(1 + Ex(Node))}. \quad (3)$$

Feature selection

It is clear that what really contributed in distinguishing between categories were those unique features belonging to the categories. Because of the feature propagation, these unique features will be weighted and propagated upwards and become the features of the parent category. In this case, by tracing these unique features it is easy to locate the correct category. Due to the uniqueness of the features, there is only one path in the tree for reaching the features. This phenomenon provides a foundation for our single path classification algorithm.

The goal of our feature selection is to compare the features in a node to its sibling nodes and try to distinguish the unique features. In order to do this, we take advantage of the feature propagation and use the features of the parent node as negative examples to determine a ranking. After propagation, each propagated feature probability is the weighted sum of the same feature probability from the sub-trees and the current node itself. We proposed formula (4) for determining the uniqueness ranking *R(w|node)* of a feature *w* given a *node*:

$$R(w|node) = \frac{P(w|node) * W(nodeAsSubtree)}{P(w|parentnode)}, \quad (4)$$

where *node* is the current node and *parentnode* is the parent of the current node. *W(nodeAsSubtree)* is the weight factor assigned to the current node when it is propagated to the parent. If a feature is unique in one node, it is the only source that can propagate to the parent feature list. In this case, we get the maximum value of the formula, which is 1. The unique features will be considered as key features that differentiate the node from its siblings and form the basis for our single path traversal algorithm.

Classifying a Web page

The concept of text classification requires the use of a classifier to assign values to a document and to each catalogue. Matching a document feature values to a category feature values, the

Online Information Review
Volume 28 · Number 2 · 2004 · 139-147

category with a global maximum value is considered the correct place to hold the document. Most automatic classification researchers have concentrated on global search algorithms, which treat all categories in a flat structure when trying to find the maximum. It follows that in order to find the category with the greatest value, it is necessary to compare all the categories.

For the tree structure that we have configured in the previous section, we claim that travelling one path is sufficient to achieve this goal. If there are n categories in the tree, the complexity of searching all categories is $\theta(n)$, but by using our single path algorithm it is merely $\theta(\log(n))$ for a balanced classification tree. This stage consists of the following steps:

(1) creating a feature list for a new Web page;
(2) breadth-first search for finding the maximum on a path;
(3) creating a new category if needed;
(4) merging the Web page information to the category; and
(5) updating the category hierarchy.

Creating a feature list for a new Web page

When classifying a Web page, the first step is to create a representation of the page. This includes assigning different weight to each sentence according to Table I, performing stemming, removing stopping words, and selecting N-gram terms. Then, the weighting probability for each feature is calculated. All the features are grouped together to form a feature vector for presenting the Web page as discussed previously.

Breadth-first search for finding the maximum on a path

The idea of the single path traversal is to eliminate the impact of other branches in the tree. After feature propagation, the propagated feature list of the parent node is a scaled summation of the propagated feature lists of all its child nodes. Thus, by checking the parent node we can know the information of its descendents. In our tree structure, all the features are propagated upwards with the ranking function of formula (2), so each category has unique features of its own to differentiate from its siblings. These two steps make the single path traversal possible.

The first step of the single path traversal is to discriminate sibling nodes in each level and find a correct path for the incoming Web page. In order to determine the discriminating probability for each node, we only consider the nodes at each level and apply the PrTFIDF formula (1) on the features with a ranking of 1, which indicate a unique feature. At each level, we chose the node with the maximum discriminating probability as the starting point for the next iteration. Recursively applying this rule creates a path from the root of the tree to one of the leaf nodes.

Then following this path we apply the PrTFIDF classifier again using all the features of the nodes belonging to the path, to get the actual probability for the page with categories within this path. By picking the node along the classification path with maximum actual probability value, we determine the candidate category for the page.

Creating a new category

In this section, we describe our proposed new dynamic tree expansion algorithm. As more and more documents are being put into a category set, the diversities of the documents make the existing category unable to guarantee classification accuracies. It is necessary for expanding the category tree to accommodate all the pages in order to yield accurate results. The problem of how to dynamically generate more sub-categories for the existing category set becomes evident.

Based on statistical results, we determine a set of criteria and check whether the criteria have been met for putting a page under the current category. If the criteria have not been met, we will create a new node for the page. As a result, the tree will be expanded by adding a new category. An updating algorithm is also introduced to incorporate the expansion information into the existing category set.

The criteria for creating new categories must be established. There are two types of expansion – deeper and wider – and two thresholds will be used to determine the criteria. The two thresholds B and D are obtained in the following ways. We take a set of sample pages from a category and calculate the probability of these pages relating to its category. We denote the resulting values for each category as S_i for i from 1 to the number of all categories n. Then, we compute the normal distribution of all sample S_i for $i = 1$ to n, and obtain the mean μ and the standard deviation d. We set $B = \mu - d$ and $D = 2d$.

We define the relation between a document and the category in which it should be placed as the "belongs to" relation. It is a basic assumption that the probability of a page given a category having the "belongs to" relation will have a maximum value. In expansion, where we set a lower bound for this relation, threshold B will be used to ensure the maximum characteristic of the relation.

A deeper expansion happens when the maximum actual probability value (obtained from single-path search described above) is smaller than threshold B. That is although the value is the maximum found, its corresponding category is not considered to be sufficiently suitable for the new page. A new sub-category

Online Information Review
Volume 28 · Number 2 · 2004 · 139-147

under the category is then created to hold the new page.

A wider expansion occurs in the following situation: the probability of the page being comparable to that of the candidate category (the one with the maximum actual probability value) is much bigger than the probability of the page and that of the candidate's child categories. The difference of probability represents the distinction between the page and the candidate's child nodes. Ignoring this distinction will cause the inconsistency of the "belongs to" relation between parent and children. When updating the category information after a page is put into a category, those features, which contribute to the difference, will also be incorporated into the category. This makes the relation between the category and the existing children more and more distinct. By creating the new category, we put the distinction to the siblings other than making the relation between parent and children far apart. Thus, the newly added features will only heavily affect the new created category. When propagating the new features to the candidate category, the weight factor is used to reduce the effect of the new feature. This reduces the inconsistent effect.

Based on these two cases, the threshold B ("belongs to" threshold) and the threshold D (difference threshold) are used as criteria to determine when expansion is needed. For deeper expansion, the probability for the new page (document *d*) given the candidate node is less than B, that is $P(d|c) < B$. In this case, we will create a new category. For wider expansion, the candidate note is not a leaf node of the tree. We need to check the probability of the page given each of the sub-categories of the candidate category $P(d|Sub_i)$. If the difference of $P(d|c)$ and the maximum number of the $P(d|Sub_i)$ is out of the range of threshold D, that is $P(d|c) - \text{Max}(P(d|Sub_i)) > D$, then the new page is considered to be substantially different from any of its sub-categories. In this case, we will create a new category.

Merging the Web page information to the category

When a page is assigned to a category, the feature vector of the page will be contributed to the category information. It is necessary to update the category information feature list to maintain the concordance, and the hierarchical structure and expansion of the vocabulary will inevitably change the characteristics of some of the categories.

After the page has been put into a category, the page is considered to be one part of the category, so it will contribute its own characteristics to the category. Both the page and category information are represented as feature vectors. Merging the page vectors into the category vectors is a solution for updating the category information. Since the page features are selected by a "well-grained text-learning method" (Peng and Choi, 2003) with those low frequency patterns removed, we assume that the extracted features are considered to be relevant to represent the page. We use the following formula for updating the category information:

$$P\text{“}(w|N) = \frac{|V| * P(w|N) + |Vpage| * P(w|Page)}{|V| + |Vpage|}, \quad (5)$$

where |V| represents the size of the category vector and the |Vpage| is the page vector size.

Updating the category hierarchy

After merging page vectors, the category information is changed, so the ranking and propagated feature probability should be recalculated. Since those changes only happen along the classification path, the updating only takes place along the single path. The process includes the following steps:

(1) Merge the feature vector of the page into the features vector of the nodes that the page is classified into.
(2) Recalculate the weight and feature probability of the nodes along the classification path by using formulas (2) and (3).
(3) Recalculate the ranking of the features list of the nodes along the classification path by using formula (4).

Test results and performance analysis

We designed experiments to test two aspects of our system: accuracy in dynamic expansion and performance of the single path traversal. The root of our experiment is set to Yahoo!/Science/Engineering/, one of the sub-categories of Yahoo!'s classification tree. We also chose the categories that strictly follow the levels of this root category; that is, we eliminated those categories that either go to another root category or do not follow the level structure. As we have noticed, many of the outgoing URLs in Yahoo! are not accessible. We chose Science/Engineering as the root because it is newly generated and covers 4,068 outgoing URLs as advertised. Our expectation is that it will somewhat reduce the number of outdated outgoing URLs and provide enough testing examples for our system.

Experiment settings

Considering processing time for testing purposes, we direct our program to obtain the category information of three levels in the Yahoo! "Science/Engineering" sub-tree. Labrou and Finin (1999) compared several different ways of describing the category information and

Online Information Review
Volume 28 · Number 2 · 2004 · 139-147

the Web page information. Their experiments demonstrated that the best way of describing category was entry summaries and entry titles; correspondingly the best choice of Web page description (called entry in their paper) was the entry summaries. Because of this, when generating the category information, we use the summaries that are generated by humans and already presented in Yahoo.com in addition to Web site contents.

The examples tested on our system are Web site contents whose URLs are taken from three levels of the Yahoo! sub-tree rooted in "Science/Engineering." Some non-text format pages associated with the URLs cannot yet be classified, for example, jpeg, swf, and gif files. We only use the URLs whose pages have more than 70 features after our well grain 3-gram feature extraction: these URLs are called good URLs.

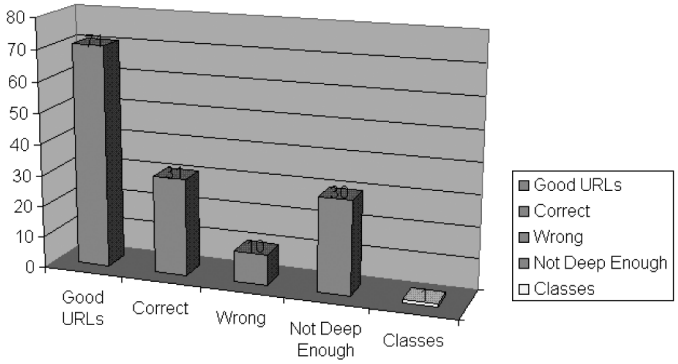
The two thresholds are generated at the machine-learning step using the statistic of the page-category probability. All the categories at the second and the third levels of "Science/Engineering", except the "Science/Engineering/organisations" category, are defined as existing categories. One third of the URLs that belong to those categories are taken to calculate the two thresholds based on the ways that we have described already.

Expansion tests

Our expansion experiments are designed to test two kinds of accuracies: deeper test – testing the accuracy for deeper expansions (see results in Figure 1), and wider test – testing the accuracy for wider expansions, (see results in Figure 2).

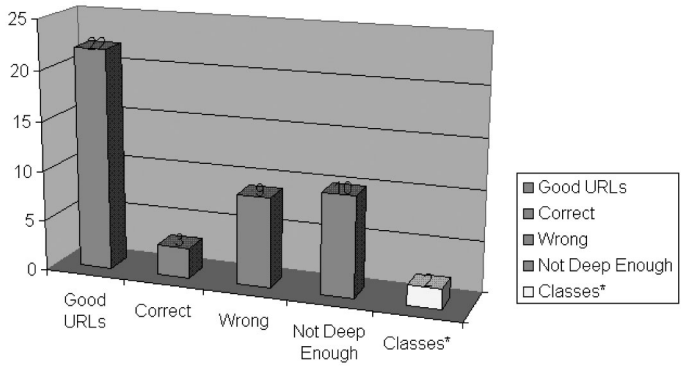
All of the URLs that have the "belong to" relation of the category "Science/Engineering/organisation" are considered to be a "new" group and used to test the wider expansion. In this experiment, the number in the "correct" column means how many pages are classified to an expanded category that is under the selected

Figure 1 Accuracies for deeper expansion



	Good URLs	Correct	Wrong	Not Deep Enough	Classes
■	71	31	10	30	1

Figure 2 Accuracies for wider expansion



	Good URLs	Correct	Wrong	Not Deep Enough	Classes*
■	22	3	9	10	2

"Science/Engineering/organisation". If the page is classified in a category that contains the category where this page comes from, it is called "Not Deep Enough." The column "Classes" keeps the number of the newly expanded category. Since all the URLs are taken from a same category, we are expecting the number to be 1. In the deeper test, the testing URLs come from "Science/Engineering/Civil_Engineering/Institutes/." Similarly to the wider test, we have same settings for the experiment results. Experiment results are provided in Figures 1 and 2.

Testing our single-path algorithm

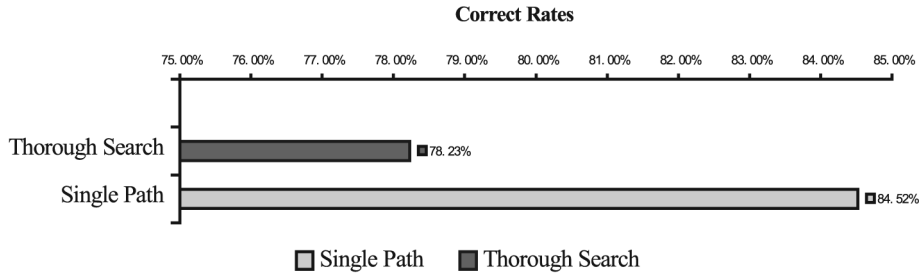
In order to test the accuracy of our single path algorithm, we compare the classification results of the single path algorithm to the one that searches all categories, the latter of which is used in most existing systems. Using 33 percent of all the Web pages in the Yahoo! "Science/Engineering" tree as test Web pages, we compare the two algorithms for accuracy and the effectiveness. The results are shown in the list below and summarised in Figure 3:

(1) *Both algorithms produce same classification results:*
   - Correct results: 333.
   - Wrong results: 6.

(2) *Algorithms produce different classification results:*
   - Correct results by single path algorithm: 58.
   - Correct results by thorough search algorithm: 30.
   - Wrong results by both algorithm: 37.

From the results, we can see that the two algorithms have the same results in more than 73 percent of all testing cases. Surprisingly, even when we have ignored most of the branches of the tree, our single path classification still has an accuracy of 84.26 percent, which even outperforms by about 6 percent the accuracy

Figure 3 Classification accuracy



that was achieved by the thorough search algorithm (Figure 3).

Performance analysis of our single-path classification algorithm

Test results show that our single-path classification algorithm improves the classification accuracy compared to algorithms that search through all categories. At first look these results were surprising. After careful analysis, we concluded that the improvement in accuracy is due to the additional information provided by our hierarchically originated categories. The other algorithms that search through all categories do not have the additional information since they treat all categories equally and have no hierarchy.

Our single-path classification algorithm searches for the most appreciated category along a path from the root to a leaf of our classification tree. It has computational complexity of log (n) for a typical classification tree having n categories. This is a large improvement over other algorithms that require searching through all n categories. However, we need to carefully address the problem of a large number of features accumulated at the upper parts of the tree, which is the result of propagating features upward from the leaves to the root node. This problem is related to the problem of high dimensionality and is usually addressed by feature selection (Mladenic and Grobelnik, 1998a, b).

Besides using feature selection, we also utilise a hash table for addressing the problem of high dimensionality. We store features of each category in a hash table. When performing classification, we compare features of a Web page to the features in the hash tables. The number of features in a Web page is usually much smaller than that of a category. Thus, we greatly reduce the number of comparisons needed for our classification. Our performance results confirmed our analysis. In fact, we have two implementations: one using link lists to store the features and the other using hash tables. As we expected, the one using hash tables outperforms the one using link lists.

Conclusion and future research

In this article we describe an approach that utilises class hierarchies for improving Web page classification. Our classification system reduces computational complexity and improves classification accuracy compared to other systems that search through all categories. We also proposed an approach to dynamically create new categories to accommodate the ever growing number of Web pages on the Internet.

Although our tests show that our proposed system improves classification accuracy by 6 percent over similar systems that search through all categories, further testing with a larger dataset is needed to establish better comparisons. Moreover, further research is needed to create better methods for dynamically creating new categories.

Our work relates to the future development of a Semantic Web. The goal is to organise the vast amount of information on the Internet in a meaningful way. Automatic classification of Web pages plays a major role in this endeavour. Hierarchical structure and dynamic growing mechanisms can be considered as bases of a structured Internet. To achieve the goal much future research remains to be done.

References

Chan, P.K. (1999), "A non-invasive learning approach to building Web user profiles", *KDD-99 Workshop on Web Usage Analysis and User Profiling, 5th International Conference on Knowledge Discovery and Data Mining, San Diego, CA*, pp. 7-12.

Chekuri, C., Goldwasser, M.H., Raghavan, P. and Upfal, E. (1997), "WebSearch using automatic classification", *Proceedings of the 6th International World Wide Web Conference, Santa Clara, CA, April.*

Choi, B. and Guo, Q. (2003), "Applying semantic links for classifying Web pages", *Developments in Applied Artificial Intelligence, IEA/AIE 2003, Lecture Notes in Artificial Intelligence*, Vol. 2718, Springer-Verlag, Heidelberg, pp. 148-53.

Dumais, S. and Chen, H. (2000), "Hierarchical classification of Web content", in Belkin, N.J., Ingwersen, P. and Leong, M.-K. (Eds), *Proceedings of SIGIR-00, 23rd ACM International Conference on Research and Development in Information Retrieval, Athens*, ACM Press, New York, NY, pp. 256-63.

Fuhr, N., Hartmanna, S., Lustig, G., Schwantner, M. and Tzeras, K. (1991), "A rule-based multi-stage indexing system for large subject fields", in Lichnerowicz, A. (Ed.), *Proceedings of RIAO-91, 3rd International Conference "Recherche d'Information Assistee par Ordinateur", Barcelona*, Elsevier Science Publishers, Amsterdam, pp. 606-23.

Joachims, T. (1997), "A probabilistic analysis of the Rocchio algorithm with TFIDF for text categorisation", in Fisher, D.H. (Ed.), *Proceedings of 101 ICML-97, 14th International Conference on Machine Learning, Nashville, TN*, Morgan Kaufmann Publishers, San Francisco, CA, pp. 143-51.

Joachims, T. (1998), "Text categorisation with support vector machines: learning with many relevant features", *Proceedings of the 10th European Conference on Machine Learning (ECML)*, Lecture Notes in Computer Science, Number 1398, Springer Verlag, Heidelberg, pp. 137-42.

Koller, D. and Sahami, M. (1998), "Hierarchically classifying documents using very few words", in Fisher, D.H. (Ed.), *Proceedings of 101 ICML-97,14th International Conference on Machine Learning, Nashville, TN*, Morgan Kaufmann Publishers, San Francisco, CA, pp. 170-8.

Labrou, Y. and Finin, T. (1999), "Yahoo! as an ontology – using Yahoo! categories to describe documents", *CIKM "99, Proceedings of the 8th International Conference on Knowledge and Information Management*, ACM Press, New York, NY, pp. 180-7.

Lang, K. (1995), "Newsweeder: learning to filter news", in Prieditis, A. and Russell, S. (Eds), *Proceedings of the 12th International Conference on Machine Learning (ML-95)*, Morgan Kaufmann, San Mateo, CA, pp. 331-9.

Lewis, D.D. and Ringuette, M. (1994), "A comparison of two learning algorithms for text categorisation", *3rd Annual Symposium on Document Analysis and Information Retrieval (SDAIR'94)*, pp. 81-93.

Mladenic, D. (1998a), "Feature subset selection in text-learning", *Proceedings of the 10th European Conference on Machine Learning ECML98*.

Mladenic, D. (1998b), "Machine Learning on non-homogeneous, distributed text data", PhD thesis, University of Ljubljana, Ljubljana.

Mladenic, D. (1998c), "Turning Yahoo! into an automatic Web-page classifier", *Proceedings of the 13th European Conference on Artificial Intelligence ECAI'98*, pp. 473-4.

Mladenic, D. and Grobelnik, M. (1998a), "Feature selection for classification based on text hierarchy", Working Notes of Learning from Text and the Web, Conference on Automated Learning and Discovery.

Mladenic, D. and Grobelnik, M. (1998b), "Word sequences as features in text-learning", *Proceedings of ERK-98, the 7th Electro-technical and Computer Science Conference*, pp. 145-8.

Paice, C.D. (1990), "Constructing literature abstracts by computer: techniques and prospects", *Information Processing & Management*, Vol. 26 No. 1, pp. 171-86.

Peng, X. and Choi, B. (2003), "Automatic Web page classification in a dynamic and hierarchical way", *IEEE International Conference on Data Mining*, pp. 386-93.

Salton, G. and Buckley, C. (1987), "Term weighting approaches in automatic text retrieval", Technical Report, COR-87-881, Department of Computer Science, Cornell University, Ithaca, NY, November.

Schutze, H., Hull, D. and Pedersen, O.J. (1995), "A comparison of classifiers and document representations for the routing problem", in Fox, E.A., Ingwersen, P. and Fidel, R. (Eds), *Proceedings of the 18th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, ACM Press, New York, NY, pp. 229-37.

Weiss, S.M., Apte, C., Damerau, F., Johnson, D.E., Oles, F.J., Goets, T. and Hamp, T. (1999), "Maximising text-mining performance", *IEEE Intelligent Systems*, Vol. 14 No. 4, pp. 63-9.

Yahoo! (2003), available at: www.yahoo.com

Yang, Y. and Pedersen, O.J. (1997), "A comparative study of feature selection in text categorisation", *Proceedings of the 5th International Conference on Machine Learning ICML97*, Morgan Kauffman, San Mateo, CA, pp. 412-20.