Tree-Based Method for Classifying Websites Using Extended Hidden Markov Models

Majid Yazdani, Milad Eftekhar, and Hassan Abolhassani

Web Intelligence Laboratory, Computer Engineering Department,
Sharif University of Technology, Tehran, Iran
{yazdani,eftekhar}@ce.sharif.edu
abolhassani@sharif.edu

Abstract. One important problem proposed recently in the field of web mining is website classification problem. The complexity together with the necessity to have accurate and fast algorithms yield to many attempts in this field, but there is a long way to solve these problems efficiently, yet. The importance of the problem encouraged us to work on a new approach as a solution. We use the content of web pages together with the link structure between them to improve the accuracy of results. In this work we use Naïve-bayes models for each predefined webpage class and an extended version of Hidden Markov Model is used as website class models. A few sample websites are adopted as seeds to calculate models' parameters. For classifying the websites we represent them with tree structures and we modify the Viterbi algorithm to evaluate the probability of generating these tree structures by every website model. Because of the large amount of pages in a website, we use a sampling technique that not only reduces the running time of the algorithm but also improves the accuracy of the classification process. At the end of this paper, we provide some experimental results which show the performance of our algorithm compared to the previous ones.

Key words: Website classification, Extended Hidden Markov Model, Extended viterbi algorithm, Naïve-Bayes approach, Class models.

1 Introduction

With the dramatically increasing number of sites and the huge size of the web which is in the order of hundreds of terabytes [5] and also with the wide variety of user groups with their different interests, probing for sites of specific interests in order to solve users' problems is really difficult. On the other hand, almost predominant section of the information which exists in the web is not practical for many users and this portion might interfere the results which are retrieved by users' queries. It is apparent that searching in the tiny relevant portion can provide us better information and lead us to more interesting sites and places on a specific topic.

At this time, there are a few directory services like DMOZ [3] and Yahoo [11] which provide us by useful information in several topics. However, as they

are constructed, managed and updated manually, most of the time they have incomplete old information. Not only webpages changes fast, but also linkage information and access records are updated day by day. These quick changes together with the extensive amount of information in the web necessitate automated subject-specific website classification.

This paper proposes an effective new method for website classification. Here, we use the content of pages together with the link structure of them to obtain more accuracy and better results in classification. Also among different models for representing websites, we choose tree structure for its efficiency. Tree model is useful because it displays the link structure of a given website clearly.

Before we begin to talk about ways of website classification, it is better to explain the problem more formally. Given a set of site classes C and a new website S consisting of a set of pages P, the task of website classification is to determine the element of C which best categorizes the site S based on a set of examples of preclassified websites. In other words, the task is to find a class C that website S is more likely to be its member.

The remaining of this paper is organized as follows. Related works are discussed in Section 2. Section 3 introduces some necessary terminologies. In section 4, we describe the models which we use for representing websites and website classes. In Section 5, we explain our method for classifying websites together with the extended version of Viterbi algorithm. Section 6 is about learning and talks about what the method do in learning phase. Section 7 contains sampling techniques. A performance study is reported in Section 8. Finally, Section 9 concludes the paper and discusses future works.

2 Related Works

Text classification has been an active area of research for many years. A significant number of these methods have been applied to classification of web pages but there was no special attention to hyperlinks. Apparently, a collection of web pages with a specific hyperlink structure conveys more information than a collection of text documents. A robust statistical model and a relaxation labeling technique are presented in [2] to use the class label and text of neighboring in addition to text of the page for hypertext and web page categorization. Categorization by context is proposed in [1], which instead of depending on a document alone, extracts useful information for classifying the document from the context of the page in which the hyperlink to the document exists. Empirical evidence is provided in [9] which shows that good web-page summaries generated by human editors can indeed improve the performance of web-page classification algorithms.

On the other hand, website classification has not been researched widely; the basic approach is superpage-based system that is proposed in [8]. Pierre discussed several issues related to automated text classification of Web sites, and described a superpage-based system for automatically classifying Web sites into industrial categories. In this work, we just generate a single feature vector counting the

frequency of terms over all HTML-pages of the whole site, i.e. we represent a web site as a single "superpage". Two new more natural and more expressive representation of web sites has been introduced in [6] in which, every webpage is assigned a topic out of a predefined set of topics. In the first one, Feature Vector of Topic Frequencies, each considered topic defines a dimension of the feature space. For each topic, the feature values represent the number of pages within the site having that particular topic. In the second one, Website Tree, the website is represented with a labeled tree and the k-order Markov tree classifier is employed for site categorization. The main difference between our work and this work is that in the latter, the topic (class) of each page is independently identified with a classifier and without considering other pages in the website tree, and then the topic of pages tree is used to compute the website class, but this independent topic identification will lower the accuracy of responses. Whereas in our work we calculate the website class without explicitly assigning a topic to each page and the topics of pages will be hidden to us. In [7] a website is represented with a set of feature vectors of terms. By choosing this representation, effort spent on deriving topic frequency vectors will be avoided. kNN-classification is employed to classify a given website according to training database of websites with known classes. In [10] the website structure is represented by a two layered tree: a DOM tree for each page and a page tree for the website. Two context models are used to characterize the dependencies between nodes. The Hidden Markov tree is used as the statistical model of page trees and DOM trees.

Other works have been done based on Hidden Markov Model classification for sequential data. In [4] a new approach is presented for classifying multipage documents by naïve bayes HMM, in which the input of system is a sequence of documents and each document is a bag of words which is represented by naïve-bayes. However, to the best of our knowledge, there is no work on extending HMM for classifying data represented as tree.

3 Necessary definitions for labels and page models

In this Paper, we explain a new method to classify websites more accurately. Before formally describing our approach, we first introduce the following definitions.

Definition 1. *(The Set of webSite Class Label : SL) SL is the set of labels which can be assigned to websites as class labels, in other words, members of SL are category domains. This set can be obtained by human experts or by using websites like DMOZ and Yahoo!.*

An example of website class label set is given in example 1.

Example 1. The set SL = {Arts, Business, Computers, Games, Health, Home, Kids and Teens, News, Recreation, Reference, Regional, Science, Shopping, Society, Sports, World} of category domains is retrieved from DMOZ[3].

Definition 2. *(The Set of webPage Class Labels : PL) For each label sl in SL, there is a set of known labels which can be assigned to individual pages of a website in that specified category. This labels can be seen as pages' topics.*

We provide an example of webpage class label set in example 2.

Example 2. Assume we choose category "Shopping" $\in$ SL. A typical PL set for it can be as follows:
PL = {home page, product page, customer review page, master card page, $\cdots$}

To continue, it is necessary to construct a model for website classes which are members of the SL set. But before, we have to provide a model for describing page classes.

There are many choices to use as page class models like Naïve-Bayesian approach and Hidden Markov models. We prefer the former due to its simplicity. Thus we adopt Naïve-Bayes model for webpage classes in this paper.

4 Using an extended Hidden Markov Model to model websites and classes

Given the above definitions, we can construct a new model for website classes, now. Here, we extend Hidden Markov Model in order to satisfy the criteria of content and link structure within pages which we talked about it before.

Definition 3. *(web Site Class Model : SCM) For each category domain sl $\in$ SL, the model m which corresponds to sl is a directed graph $G(V, E)$ in which V ,G's vertex set, is a set of webpage class models and for every two states $v_i, v_j \in V$ there is two directed edges $\{(i.j), (j, i)\} \in E$ that show the probability of moving from one to another. Also there is a loop for every state which shows the probability of transition between two pages of that class.*

In this model, the state-transition probability matrix A is an $n * n$ matrix in which n is the cardinality of SL and $a_{ij} = P(c_{t+1} = j | c_t = i),\ 0 \leq i, j \leq n$, which shows the probability of transition from state i to state j. Also the emission probability e is: $e_j(p) = P(p_t = p | c_t = j)$.

$e_j(p)$ represents the probability of generating page p by webpage class model j. You can see an example of a typical website class model in figure 1. Our extended Hidden Markov Model is different from the standard HMM. First, in our model we can move from one state to multiple states in the next time stamp. As an example, You can consider a typical product page which has multiple links to other product pages and other page types like homepages, user comment pages, contact information pages and etc. Second main difference is about inputs which are not sequences. In these models websites which are described by trees of pages are input entries. As a result, we have a hybrid of Naïve-Bayes HMM model for all of websites classes. As we mentioned before, we can model website classes by a hierarchial HMM if we use HMM instead of Naïve-Bayes model for describing webpage classes. Regarding above explanations, the input of website

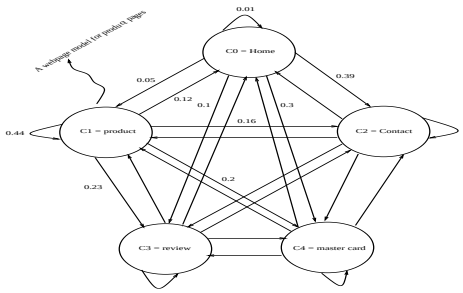

Fig. 1. A sample website class model with five states (each state is itself a webpage class model)

class models are websites which are modeled by trees. We describe website models in a more formal way, as follows.

Definition 4. *(web Site Model : SM) for each Website ws, the ws's model is a page tree $T_{ws} = (V, E)$ in which V is a subset of ws's pages. The root of T_{ws} is the homepage of the website and there is a directed edge between two vertices (pages) $v_i, v_j \in V$ if and only if v_i is the parent of v_j. In other words the page p_j which corresponds to vertex v_j is one of the children of the page p_i.*

It is necessary to consider that for constructing this model, we have to crawl the website by Breath-First-Search technique and ignore the links to pages which are visited before. It is obvious that each page appears in the tree according to its smallest path from the homepage of the website graph. However the algorithm may generate different trees for a website graph in the case of having two smallest paths to a page in that graph. As a result, various tree models can be generated by different crawling policies. Thus, in that case, we will have different models for a specific website.

Example 3. Figure 2 presents a typical website graph which contains 7 pages and its related tree model.

5 Website Classification

In previous sections, we described a model for each website class. We assume SCM_i is the model of website class C_i. Also we consider that we want to classify website ws. To find the category of ws, we calculate $P(SCM_i|ws)$ for $1 \leq i \leq n$. By Considering Bayes rule, we can determine the class of ws as

$$C_{map} = argmax P(SCM_i|ws) = argmax P(SCM_i) P(ws|SCM_i)$$

$P(ws)$ is constant for all classes and we neglect $P(SCM_i)$. It can be considered later or even it can be equal for all classes if we use a fair distribution of websites over different classes. Therefore $C_{map} = argmax P(ws|SCM_i)$.

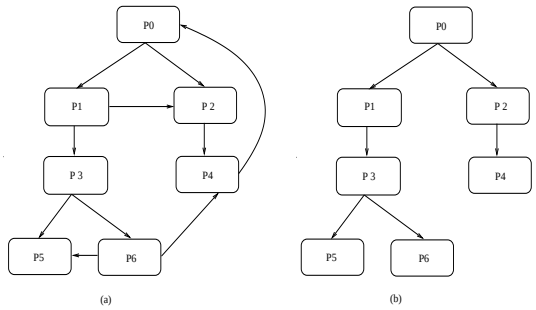

Fig. 2. Construction of website tree model from website graph of example 3. (a):A typical website with 7 pages and complete link structure between them which is modeled by a graph. (b): The corresponding tree model of website which is given in (a)

To find the probability of $P(ws|SCM_i)$ and also to solve the webpage classification problem, we should extend the Viterbi algorithm for our models.

5.1 Extended Viterbi Algorithm

To determine the most probable category which can be assigned to a certain given website, the general idea is to feed all the possible arrangements of a model's states into the website tree's pages, and find an arrangement which represents the website as closely as possible to the model.

For example suppose we want to compute the probability of generating the website in figure 2(b) from the model which is illustrated in figure 1. You can see the possible arrangements in figure 3. We should calculate all the possible

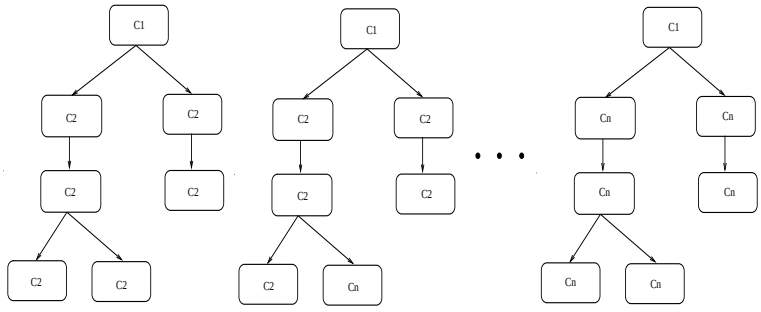

Fig. 3. Possible label Assignments for pages of the website which is presented in figure 2(b) from the website class model of figure 1

arrangements of a model's states into the website's tree model. We choose the arrangement that gives us the most probability of generating the website from

the model. Then between all models we choose the one with the highest probability. The arrangement of states that yields this probability is the most probable webpage labels. If the website tree model has n pages and we have C states for a model, we should compute probability of C^{n-1} various arrangements. Note that the root of the tree only takes c_1 label, but for the other nodes we have C different options.

We are seeking a more efficient way to compute the probability of a website. All of the structures and the ways of computing the probabilities lead us to use dynamic programming technique. So, using viterbi algorithm ideas seems reasonable, but as we mentioned before, the inputs of our models are trees of pages while viterbi algorithm is provided to calculate the maximum probability of generating sequences by HMMs. Therefore, we should modify the traditional viterbi algorithm.

Before introducing the new approach, it is necessary to present theorem 1 which is discussed in [10].

Theorem 1. *The probability of each node is only dependent to its parent and children, i.e., the probability of a node is independent of every node which is neither the parent nor the child. More formally, for every node n with p as its parent and $q_1, ...q_n$ as its children, if PL represents the webpage label set then*

$$P(PL_n|\{PL_k\}) = P(PL_n|PL_p, PL_{q_1}, ..., PL_{q_n}) \tag{1}$$

where, k is another node of tree.

For computing the probability of generating a tree model from a website class model, One might think that we can consider each of the paths from the root of the tree to a leaf as a sequence. So, it is possible to determine the probability of each sequence by Viterbi Algorithm, and we can multiply the calculated probabilities to conclude the probability of generating the whole tree from the specified website class model. However, this approach does not work correctly, due to the fact that the maximum probability of a node n_1 can be obtained by assigning webpage label pl_i to its parent p while the maximum probability of n_1's sibling, n_2, is acquired by assigning pl_j to p. Hence, by multiplying the probabilities of n_1 and n_2 we reach to an state which is meaningless as this state is constructed from an contrary label assignment to the node p. We illustrated this fact in example 4

Example 4. Assume we want to determine the probability of generating the simple website model T_{ws} which is illustrated in figure 4 from the model SCM_1 with 2 webpage class models pcm_1 and pcm_2. The State-Transition Matrix A and emission function E are as follows:

$$A = \begin{bmatrix} 0.2 & 0.8 \\ 0.6 & 0.4 \end{bmatrix}$$

$$e_{pcm_1}(p_1) = 0.25 \; e_{pcm_1}(p_2) = 0.5 \; e_{pcm_1}(p_3) = 0.7$$
$$e_{pcm_2}(p_1) = 0.4 \; e_{pcm_2}(p_2) = 0.7 \; e_{pcm_2}(p_3) = 0.3$$

We follow the proposed method now, to show the inaccuracy of that.

$$P(p_1 = pl_1) = 0.25$$

$$P(p_1 = pl_2) = 0.4$$

$$P(p_2 = pl_1) = 0.5 * \max(0.25 * 0.2, 0.4 * 0.6) = 0.12,$$
$$p_1 \leftarrow pl_2$$

$$P(p_2 = pl_2) = 0.7 * \max(0.25 * 0.8, 0.4 * 0.4) = 0.14,$$
$$p_1 \leftarrow pl_1$$

$$P(p_3 = pl_1) = 0.7 * \max(0.25 * 0.2, 0.4 * 0.6) = 0.168,$$
$$p_1 \leftarrow pl_2$$

$$P(p_3 = pl_2) = 0.3 * \max(0.25 * 0.8, 0.4 * 0.4) = 0.06,$$
$$p_1 \leftarrow pl_1$$

Therefore, the probability of generating T_{ws} from SCM_1 is as follows

$$\begin{aligned} P(T_{ws}|SCM_1) &= \max\{P(p_2 = pl_1), P(p_2 = pl_2)\} \\ &* \max\{P(p_3 = pl_1), P(p_3 = pl_2)\} \\ &= 0.14 * 0.168 = 0.02352 \end{aligned}$$

In the above calculation you can see that we have assigned pl_2 to p_2 and pl_1 to p_3. But what is the page label of p_1? It is obvious that in this case, we have to assign 2 contrary labels to p_1!

For solving this inaccuracy we propose a novel method in which the probability of a node and its siblings is computed simultaneously. Thus, in the nth level of the tree, we calculate the probability of the children of an $n-1$th level's node by equation 2.

Algorithm 1 *(Extended Viterbi Algorithm) For Classifying an indicated website ws which is modeled by a tree structure T_{ws} against n different classes $C_1, \ldots, C_n$ which are modeled by $SCM_1, \ldots, SCM_n$, if p is a node in level $n-1$ of T_{ws} and it has children $q_1, \ldots, q_n$ then*

$$P[(q_1, \ldots, q_n) \leftarrow (pl_{s_1}, \ldots, pls_n)] = \prod_{i=1}^{n} e_{pl_{s_i}}(q_i) * \max_{j=1}^{m}(P(p = pl_j) * \prod_{i=1}^{n} A_{jpl_{s_i}}) \tag{2}$$

where $pl_{s_i} \in PL$.

In equation 2, $P[(q_1, \ldots, q_n) \leftarrow (pl_{s_1}, \ldots, pls_n)]$ means the maximum probability of generating the nodes of upper levels by the model as well as assigning pl_{s_1} to page q_1,pl_{s_2} to page q_2,$\cdots$, and pl_{s_n} to page q_n.

To calculate equation 2 for lower levels, we should have the probability of each page individually. Therefore we calculate these probabilities by equation 3.

$$P(q_i = pl_j) = \sum P[(q_1, \ldots, q_n) \leftarrow (pl_{s_1}, \ldots, pl_{s_n})] \tag{3}$$

where $pl_{s_i} = pl_j$ *and* $\forall k \neq i : pl_{s_k} \in PL$.

Now we can estimate the probability of generating the tree model of the website ws from the model SCM_i

$$P(T_{ws}|SCM_i) = \prod_q max_{j=1}^{|PL|} P(q = pl_j) \tag{4}$$

in which q is a leaf of the tree. The pseudo code of Algorithm 1 is presented here.

Algorithm 1

//Initialization
$ClassProb = Array[N]$ $N = |SL|$
$ProbMatrix = Array[L][V][N][P]$
$for(n = 1\ to\ N)\{$
$M = |PL(n)|$
$for(m = 1\ to\ M)\{ProbMatrix[1][1][n][m] = P(root = pl_m) = e_m(root)\}\}$
$l = 1$
//Iteration
$while(l < L)\{$
$for(n = 1\ to\ N)\{$
$v = 0$
$for(p =$ a page in level $l)\{$
$C = |children(p)|$
$A = array[M^C]$

$for(d = 1\ to\ M^C)\{$
$A[d] = \prod_{c=1}^{C} e_{d_c}(child_c) * \max_{m=1}^{M}(P(p = pl_m) * \prod_{c=1}^{C} A_{md_c})$ where $d = (\overline{d_C \dots d_1})_M\}$
$for(child_c \in p's\ children)\{$
$for(m = 1\ to\ M)\{$
$ProbMatrix[l+1][v+c][n][m] = P(child_c = pl_m) = \sum_{d=1}^{M^C} A[d] * Eq(d_c + 1, m)$ where $d = (\overline{d_C \dots d_c \dots d_1})_M\}\}$
$v+ = C$
}
}
//Check Pruning conditions to download the next level pages
$level++$
}
//Comparison
$for(n = 1\ to\ N)\{$
$ClassProb[n] = 1$
$for(leaf\ q\ which\ is\ the\ vth\ vertex\ of\ level\ l)\{$
$ClassProb[n] * = \max_{m=1}^{M}\{ProbMatrix[l][v][n][m]\}\}\}$
$Class_{website} = arg\max_{n=1}^{N} ClassProb[n]$

In above pseudo code, N is the cardinality of the set of website labels ,SL, and M is the cardinality of its corresponding webpage label set $PL(n)$. Every element

i of array ClassProb is responsible for keeping the probability of generating the given website from the class model SCM_i. ProbMatrix is a 4-dimensional matrix in which L is the specific number of tree levels, V is the maximum number of vertexes in different levels, N is the cardinality of SL and $P = \max_n\{|PL(n)|\}$. Also Eq is a function that returns 1 if its two arguments are equal, otherwise it returns 0. We can suppose that always the root's label is"homepage". In this case, fifth line of pseudo code (for loop) is unnecessary and m is only 1 there. For clarifying the method, we explain it using example 5.

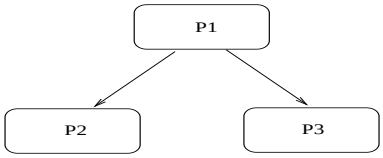

Fig. 4. The tree model of a simple website which is used in example 5

Example 5. There are two website classes C_1 and C_2, and we want to classify the website ws which is modeled in figure 4. The model SCM_1 for class C_1 has 2 webpage class labels pl_1 and pl_2. Also assume that SCM_2 for website class C_2 consists of $PL_2 = \{pl_3, pl_4\}$. The State-Transition Matrix A for SCM_1 and emission functions E are written below. In addition, model SCM_2 is illustrated in fig. 5.

$$a_{11} = 0.2, \quad a_{12} = 0.8, \quad a_{21} = 0.6, \quad a_{22} = 0.4$$
$$e_{pl_1}(p_1) = 0.25,\ e_{pl_2}(p_1) = 0.40,\ e_{pl_3}(p_1) = 0.70,\ e_{pl_4}(p_1) = 0.95$$
$$e_{pl_1}(p_2) = 0.50,\ e_{pl_2}(p_2) = 0.70,\ e_{pl_3}(p_2) = 0.40,\ e_{pl_4}(p_2) = 0.01$$
$$e_{pl_1}(p_3) = 0.70,\ e_{pl_2}(p_3) = 0.30,\ e_{pl_3}(p_3) = 0.60,\ e_{pl_4}(p_3) = 0.50$$

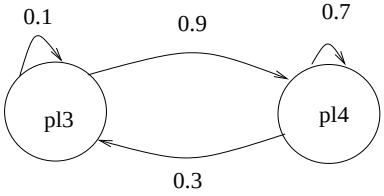

Fig. 5. The website class model SCM_2 which is used in example 5

The probability of generating ws from the model SCM_1 is obtained as follows:

$P(p_1 = pl_1) = 0.1$
$P(p_1 = pl_2) = 0.4$
$P(p_2 = pl_1, p_3 = pl_1) = 0.5 * 0.7 * \max(0.1 * 0.8 * 0.8, 0.4 * 0.4 * 0.4) = 0.0224$
$P(p_2 = pl_1, p_3 = pl_2) = 0.5 * 0.3 * \max(0.1 * 0.8 * 0.2, 0.4 * 0.4 * 0.6) = 0.0144$
$P(p_2 = pl_2, p_3 = pl_1) = 0.5 * 0.7 * \max(0.1 * 0.2 * 0.8, 0.4 * 0.6 * 0.4) = 0.0336$
$P(p_2 = pl_2, p_3 = pl_2) = 0.5 * 0.3 * \max(0.1 * 0.2 * 0.2, 0.4 * 0.6 * 0.6) = 0.0216$

now we calculate the individual probabilities

$P(p_2 = pl_1) = P(p_2 = pl_1, p_3 = pl_1) + P(p_2 = pl_1, p_3 = pl_2) = 0.0244 + 0.0144 = 0.0368$
$P(p_2 = pl_2) = 0.0336 + 0.0216 = 0.0552$
$P(p_3 = pl_1) = 0.0224 + 0.0366 = 0.0560$
$P(p_3 = pl_2) = 0.0144 + 0.0216 = 0.0360$

Therefore the probability of generating ws from SCM_1 is

$$P\ (T_{ws}|SCM_1) = \max(P(p_2 = pl_1), P(p_2 = pl_2)) * \max(P(p_3 = pl_1), P(p_3 = pl_2))$$
$$= 0.0522 * 0.0560 = 0.1112$$

By similar computation, we can calculate the probability of generating ws by using SCM_2 as class model. Thus

$$P(T_{ws}|SCM_2) = P(p_2 = pl_3) * P(p_3 = pl_4) = 0.002582$$

By comparing the probabilities which are obtained from these two classes, the website ws can be classified as a member of class C_1.

$$P(T_{ws}|SCM_1) > P(T_{ws}|SCM_2) \Rightarrow ws \in C_1$$

It is important to note that by calculating the probability of all siblings together, each set of label assignments to children of a specific node corresponds to a particular label for the parent node. Therefore, these label assignments are accurate. In the calculation of individual probabilities for every node, we compute the probability of a consistent set by adding the probabilities of some consistent label assignments. Thus, all of these probabilities are achieved from consistent states.

Here we want to compute the complexity of our algorithm. We should compute $p(q_i = pl_j)$ for every node in each level. Suppose we have branching factor of maximum b in this tree, so for each set of siblings the computation of $P[(q_1, ..., q_b) \leftarrow (pl_{s_1}, ...pl_{s_b})]$ for every possible order of $< s_1, \ldots, s_b >$ has $O(n^{b+1})$ time complexity, in which n is the number of page labels. For each node in this set of siblings we can calculate $p(q_i = pl_j)$ from $P[(q_1, ..., q_b) \leftarrow (pl_{s_1}, ...pl_{s_b})]$ probabilities in $O(n^{b-1})$ time. So we compute $p(q_i = pl_j)$ for each node in $O(n^{b+1})$. Whereas we should compute this probability for every node in the tree, the total complexity will be $O(n^{b+1}b^{L-2})$, where L is the number of tree's levels.

6 Learning Phase

As mentioned above, first, we determine website class and webpage class labels. Then a set of sample websites for learning phase are assigned by an expert or from sites like DMOZ as seeds for constructing class models. One of the fundamental steps in this phase is assigning a label to each web page. There are two types of pages: One that has predetermined labels (i.e. form a constant part in their URL) and the other which has no specified label. We have to assign labels to the second type. We assign labels to about %2 of pages in the Training set manually. The remaining %98 of the pages will be labeled by Naive Bayes based upon this 2 percent.

To construct website class models, we have to compute the state-transition matrix A and emission probability e. A can be easily computed as follows.

$$a_{ij} = \frac{N(c_i, c_j) + 1}{\sum_{k=1}^{n} N(c_i, c_k) + n}$$

in which $N(c_i, c_j)$ is the number of times that a page of type c_i links to a page of type c_j in the training set.

As we use naïve-bayes, we can also easily compute $e_{c_j}(p)$. Here we use feature selection and we select some keywords from page words. V represents the set of selected keywords. Therefore, for each page model c_j and word w_l

$$P(w_l|c_j) = \frac{N(w_l, c_j) + 1}{\sum_{i=1}^{|V|} N(w_i, c_j) + |V|}$$

$N(w_l, c_j)$ is the number of occurrence of word w_l in webpages of class c_j. Therefore, if p is formed from $w_1 \dots w_n$ then $e_{c_j}(p) = P(w_1|c_j) \dots P(w_n|c_j)$.

7 Website Sampling

There are two general reasons for our motivation to use page pruning algorithm. First, downloading web pages in contrast with operations that take place in memory is very expensive and time consuming. In a typical website there are too many pages that cannot convey useful information for website classification, if we can prune these pages, website classification performance improves significantly. The second reason that leads us to use pruning algorithm is that in a typical website, there are pages that affect classification in an undesirable direction, so pruning these unrelated or unspecific pages can improve our accuracy in addition to performance.

Here the basic approach is reading pages of n first levels, and prune all other pages. But this approach does not work properly because in the web pages we have many different structures that may not be appropriate for classification like flash intro, animations and etc; besides the same information can be published in different pages based on views of different authors. So a fixed value for n cannot be a good idea.

We use the pruning measures used in [6] for its efficiency and we modify the pruning algorithm for our method. To compute measures for a partial tree which we have downloaded up to now, we should be able to compute membership of this partial tree website for each website class. Our model is suitable for this computation because we compute the probability of each partial tree incrementally and when we add new level to previous partial tree we just compute the $P[(q_1, ..., q_b) \leftarrow (pl_{s_1}, ...pl_{s_b})]$ probabilities for every possible set of $\{s_1, \ldots, s_b\}$ for all the new siblings by using previous probabilities and according to our algorithm. Then by using these probabilities we calculate $p(q_i = pl_j)$ for each node in the new level. For each node q, we define $P(q|sl_i) = \max_{pl_j \in PL} p(q = pl_j)$ where $sl_i \in SL$.

We modify the measures described in [6], so we define weight for each node q of partial website tree t as follows: $weight(q) = \sigma^2_{sl_i \in SL}(P(q|sl_i)^{\frac{1}{depth(q)}})$. By adding a new node q to a partial tree t we obtain a new partial tree t_2. We stop descending the tree at q if and only if $weight(q) < weight(parent(q)) * \frac{depth(q)}{\omega}$. By means of this pruning procedure and saving probabilities, we perform classification in the same time we use pruning algorithm, and then we can determine the most similar class of the given website. We examine different ω to find appropriate one for our data set. Choosing proper ω can help us to achieve even higher accuracy than complete website download.

8 Experimental Results

In this section we demonstrate the results of some experimental evaluation on our approach and compare it with other existing algorithms, mainly extracted from [6]. We use these methods to classify scientific websites into ten following classes: Agriculture, Astronomy, Biology, Chemistry, Computer Science, Earth Sciences, Environment, Math, Physics, Other. We use DMOZ[3] directory to obtain websites for the first 9 classes. We downloaded 25 website for each class and a total of 86842 web pages. At this point we downloaded a website almost completely (Limited number of pages at most 400). For the "other" class we randomly chose 50 websites from Yahoo! Directory classes that were not in the first nine classes which had 18658 web pages. We downloaded all websites to local computer and saved them locally. To use our algorithm first we should prepare our initial data seed and then we build a model for each class and compute its parameter as stated in the learning phase. To classify the pages of a website class, we labeled about %2 of them in each web site class manually then The remaining %98 of the pages were labeled by Naive Bayes based upon this labeled pages. At the end of this process we have a naïve-bayes model for each page class of each website category. By means of these naïve-base models we classified web pages for other methods. For testing our methods we randomly downloaded 15 new website almost complete (limiting to 400 pages)for each class.

We compare our algorithm to 4 other methods: 0-order Markov tree, C4.5, Naïve-bayes, classification of superpage. In superpage, classifying a web site is to extend the methods used for page classification to our definition of web sites.

We just generate a single feature vector counting the frequency of terms over all HTML-pages of the whole site, i.e. we represent a web site as a single "super-page". For C4.5 and naïve-bayes, first we build Feature vector of topic frequencies and then apply naïve-bayes and C4.5 algorithms on them. For the 0-order Markov tree we used the method described in [6]. You can find the accuracy of tested methods on testing dataset in table 1.

Table 1. Comparison of accuracy between different classification methods

Classifier	Accuracy
Super Page	57.7
Naïve-Bayes	71.8
C4.5	78.5
0-Order Markov Tree	81.4
Our algorithm	85.9

As it can be seen the accuracy of our method is better than other methods. It is more accurate compared to 0-order Markov tree because page classes are hidden here and we calculate probability of the whole website that is generated from a model. It is trivial we have higher time complexity here in comparison to 0-order Markov tree.

At last we examine the impact of different ω values on the sampling algorithm in our training set. With an appropriate ω, the accuracy increases in comparison to complete website. To find appropriate ω, we increased ω gradually and when the overall accuracy stopped to increase, we choose ω. In our data set the appropriate ω was 6 , but this can change in respect to data set.

9 Conclusions and Future Works

In the growing world of web, taking advantage of different methods to classify websites seems to be very necessary. Website classification algorithms for discovery of interesting information leads many users to retrieve their desirable data more accurately and more quickly. This paper proposes a novel method for solving this problem. With extending Hidden Markov Model, we described models for website classes and looked for the most similar class for any website. Experimental Results show the efficiency of this new method for classification.

In the ongoing work, we are seeking for new methods to improve the efficiency and accuracy of our website classification method. Demonstrating websites with stronger models like website graphs can bring us more accuracy.

References

1. G. Attardi, A. Gullí, and F. Sebastiani: Automatic Web page categorization by link and context analysis. In: *Proceedings of THAI-99, European Symposium on Telematics, Hypermedia and Artificial Intelligence*, 105–119, Varese, IT (1999).

2. S. Chakrabarti, B. E. Dom, and P. Indyk: Enhanced hypertext categorization using hyperlinks. In: *Proc. ACM SIGMOD*, 307–318, Seattle, US (1998)
3. DMOZ. open directory project.
4. P. Frasconi, G. Soda, and A. Vullo: Text categorization for multi-page documents: A hybrid naïve bayes hmm approach. In: *1st ACM-IEEE Joint Conference on Digital Libraries* (2001)
5. J. Han and M. Kamber: *Data Mining: Concepts and Techniques.* Morgan Kaufmann Publisher, San Francisco, California (2006)
6. M.Ester, H. Kriegel, and M.Schubert: Web site mining: A new way to spot competitors, customers and suppliers in the world wide web. In: *Proceedings of SIGKDD'02, 249–258,* Edmonton, Alberta, Canada (2002)
7. HP. Kriegel, M. Schubert: Classification of Websites as Sets of Feature Vectors. In: Proc. IASTED DBA (2004)
8. J. M. Pierre: On the automated classification of web sites. In: *Linkö ping Electronic Article in Computer and Information Science*, Sweden 6(001)(2001).
9. D. Shen, Z. Chen, H.-J. Zeng, B. Zhang, Q. Yang, W.-Y. Ma, and Y. Lu: Web-page classification through summarization. In: *The 27th Annual International ACM SIGIR Conference* (2004)
10. Y.-H. Tian, T.-J. Huang, and W. Gao: Two-phase web site classification based on hidden markov tree models. *Web Intelli. and Agent Sys.*, 2(4):249–264 (2004)
11. Yahoo! Directory service.