A Naïve Bayes Learning Based Website Reconfiguration System

Jia Li[1], Huiqing Li[2*] , and Xiumei Jia[1]
[1]*Department of Computing Science, University of Alberta, Canada, {jial, xiumei}@cs.ualberta.ca*
[2]*Department of Computer Science, The University of Kent, U.K., H.Li@kent.ac.uk*

Abstract

The continuous and sharp growth of web sites in terms of size and complexity has made improving the website organization to facilitate users' navigation something of an emergency. To address this problem, in this paper we propose a website reconfiguration system using the machine learning approach. First, a Naïve Bayes Classifier is trained and then applied to identify each page in a web site as important *or* unimportant *in terms of fulfilling visitors' information needs. For those* important *pages, we check the reasonableness of their locations, which is measured by the average number of hops needed to reach them during visitor sessions. Those* important *but difficult reach pages are considered for reconfiguration, which is done by either automatically moving them to some level closer to the visitors' starting point, making it easier for users to access them, or presenting webmasters with a list of suggestions. We also propose a formula to evaluate the "global structure" of a web site, and use it to examine the effect of our system on improving website design.*

Keywords*: Naïve Bayes Classifier, Web Reconfiguration, Machine Learning, Data Mining*

1. Introduction

The World Wide Web, which is comprised of a wide spectrum of host machines and web sites, has become a key medium for information dissemination. On the web, one may find on-line shopping malls, electrical chat rooms, university application and course materials, and much more. Daily life without the Web is now unimaginable for millions of people.

However, designing and maintaining a web site so that visitors are able to find their expected information as quickly as possible can be tricky, since a web site often contains a myriad of facts, images, and hyperlinks. With the continuous growth of web sites in size and complexity, the situation has become even more challenging.

One might think that a well-designed hierarchy would solve this problem. As a matter of fact, many webmasters have attempted to build such a hierarchical organization, to reflect the typical usage patterns. Unfortunately, the designers' *a priori* expectations often diverge from the visitors' intuitions (e.g., a site may be designed for a particular kind of use, but may be utilized in many different ways in practice; therefore the designer's *a priori* expectations may be invalid). Worse, many sites outgrow their original design, accumulating links and pages in unlikely places.

This raises the issue of adjusting and improving the organization and presentation of web sites on-the-fly, according to visitors' actual access patterns (*Adaptive Web Sites* [12]). Generally, web sites may be adaptive in two basic ways: *customization* and *optimization*. *Customization* focuses on adjusting a web site's presentation for individual users; while *optimization* tries to improve the site as a whole. Although individuals vary in their web navigation patterns, most web sites have an overall preference for general use. *Customization* allows fine-grained improvement, since the changes are customized for each user. However, it relies heavily on explicit user input (e.g., users' personal information), which is either unavailable or considered intrusive. On the other hand, *optimization* allows even new users, about whom we know nothing, to benefit from the improvements [12].

In this paper, we choose to investigate website optimization by means of identifying and then relocating the *important* pages in the web site for the majority of visitors to access them easily. More specifically, in our system, a Naïve Bayes Classifier is trained to build a model for identifying *important* pages using a collected set of annotated access logs (i.e., logs that are augmented with the user's assessment of whether each page is an important page in fulfilling their information needs). The Naïve Bayes classifier has been shown to be a reasonable approach to many modeling problems, given its advantages of quick learning and low computational overhead [15]. In our model, the classifier learns from the annotated logs "browsing properties" of a web page that indicate whether the page is *important* or

∗ This work was partly done while the author was at University of Alberta

not to fulfill users' information needs. For instance,
"Any web page which is the last click of a visit session, and on which the visitor stays more than 120 seconds is an important *page."*
Please note that the rules are learned exempt from any information related to a specific page (e.g., the textual content of a web page). Therefore, our system can be applied to any web site without modification. After that, the model is applied to label each page in a given web site *important* or *unimportant*. Those frequently visited *important* pages in the web site that require a large number of "hops" to reach are moved to some level closer to the visitors' starting point, making it easier for users to access them.

A considerable number of website optimization systems have been proposed in the literature [17] [12] [14] [1]. Most of these take advantage of web server access logs, which record information regarding users' requests to a web server, in order to model users' navigational patterns. The navigational patterns are used to shed light on the gaps between web designers and web users. However, focusing solely on web server logs for website optimization has its problems [11], which could jeopardize the task of optimization:

- Incomplete Information Problem: One restriction with web server logs is that the information in them is very limited. A number of heuristic assumptions are typically made before applying any data mining algorithm; as a result, some patterns generated may not be proper or even correct.

- Incorrect Information Problem: When a web site visitor is lost, the clicks made by this visitor are recorded in the log, and may mislead future recommendations. This becomes more problematic when a web site is badly designed and more people end up visiting unsolicited pages, making them seem popular.

The contribution of this paper is as follows: First, we propose a novel system to improve website organization, which addresses the problems mentioned above by using the machine learning approach. Second, our generic approach to identifying the importance of individual pages is not restricted to any particular web site, or any particular task. Thus, our system can be applied to any web site. Third, we propose a new metric to evaluate the overall structure of a web site and the effect of a website optimization system.

This paper is organized as follows: Section 2 reviews some previous related work. Section 3 gives a brief introduction of the Naïve Bayes learning technique. In Section 4, we present our website reconfiguration system in detail, including the system architecture, learning sub-system, data preprocessing sub-system, and evaluation sub-system. Section 5 discusses the experimental evaluation of our system. Finally, Section 6 concludes this paper and suggests future work.

2. Related Work

Considerable work has been done on the web site optimization issue. Much of this work makes use of web server access logs to exact users' navigational patterns, and then uses them to improve website organization.

Nakayama et al. [17] attempt to discover the gap between website designers' expectations and visitor behavior. The former is assessed by measuring the inter-page conceptual relevance, and the latter by measuring the inter-page access co-occurrence from web server logs. The discovery of pages that are conceptually related but rarely co-occur in visits suggests areas for improvement in the web site design.

Perkowitz et al. [12] investigate the problem of index page synthesis, which is the automatic creation of pages which facilitate a visitor's navigation of a web site. By analyzing web server logs, their cluster mining algorithm finds collections of pages that tend to co-occur in visits, and groups them under one topic. They then generate index pages consisting of links to pages pertaining to a particular topic.

In [1], the PROTEUS architecture for adaptive web navigation is proposed. Adaptation in PROTEUS follows a two-step approach. First, PROTEUS mines web server logs to build models of users. Second, as users request pages at the site, PROTEUS considers all the ways in which to adapt the site (e.g., add a link between two pages, rearrange list items on a page, elide uninteresting content from a page). Among these methods, the author found adding shortcut links to be particularly useful.

[14] investigates discovering pages in a web site whose location is different from where visitors expect to find them. The authors' key insight is that visitors will backtrack if they do not find information where they expect it. The point from where they backtrack is the expected location for the page. Expected locations with a significant number of hits are presented to webmasters for site optimization. However, it can be difficult to differentiate between visitors who backtrack because they are browsing a set of target pages, and visitors who backtrack because they are searching for a single target page.

In terms of measuring the quality of a web site, methods have been proposed, but are not effective for our purposes. Berthon et al. [2] suggest a methodology for measuring the success of a web site in turning visitors into customers. However, the problem of improving a web site to become more successful in our context is not addressed. Sullivan [16] proposes a number of measures to assess the quality of individual pages: (1) quality of service, e.g. re-

sponse time; (2) quality of navigation: navigation modes supported; and (3) accessibility: whether a page's existence can be ascertained, and whether it can be found. However, it is not clear how to aggregate these measures to assess the quality of a whole site.

3. Naïve Bayes Classifiers

In this section, we give a brief background introduction of Naïve Bayes Classifiers. For further detailed information, please refer to [6] [7] [10] [3].

The Naïve Bayes classifier is an old and well-known type of classifiers. Classifiers, in turn, are programs which automatically classify a case or an object, i.e. assign it according to its features to one of several predefined classes. For example, if the cases are patients in a hospital, the attributes are properties of the patients (e.g. sex, age etc.) and their symptoms (e.g. fever, high blood pressure etc.), the classes may be diseases or drugs to administer [3].

Naïve Bayes classifiers use a probabilistic approach to assign the classes. That is, they try to compute the conditional probabilities of the different classes given the values of other attributes and then predict the class with the highest conditional probability. Since it is usually impossible to store or even to estimate these conditional probabilities, they exploit Bayes rule and a set of conditional independence statements to simplify the task. More precise, let C denote a class attribute with a finite domain of m classes, i.e., $dom(C) = c_1, ..., c_m$, and let $A_1, ..., A_n$ be a set of other attributes used to describe a case or an object of the universe of discourse. These other attributes may be symbolic, i.e., $dom(A_j) = a_1^{(j)}, ..., a_{m_j}^{(j)}$, or numeric, i.e., $dom(A_j) = IR$. For simplicity, the notation $a_{i_j}^j$ is used for a value of an attribute A_j, independent of whether is is a symbolic or a numeric one. With this notation, a case or an object can be described by an instantiation $\omega = a_{i_1}^{(1)}, ..., a_{i_n}^{(n)}$ of the attributes $A_1, ..., A_n$ and thus the universe of discourse is $\Omega = dom(A_1) \times ... \times dom(A_n)$.

For a given instance ω, a Naïve Bayes classifier tries to compute the conditional probability

$$P(C = c_i|\omega) = P(C = c_i|A_1 = a_{i_1}^{(1)}, ..., An = a_{i_n}^{(n)})$$

for all c_i and then labels the instance with the highest probability. Of course, it is usually impossible to store all of these conditional probabilities explicitly, so that a simple lookup would be all that is needed to find the most probable class. If there are numeric attributes, this is obvious (some parameterized function is needed then). But even if all attributes are symbolic, such an approach most often is infeasible: A class (or a class probability distribution) has to be stored for each point of the Cartesian product of the attribute domains, whose size grows exponentially with the number of attributes. To circumvent this problem, Naïve Bayes classifiers exploit – as their name already indicates - Bayes rule and a set of conditional independence assumptions. With Bayes rule the conditional probabilities are inverted, i.e., Naïve Bayes classifiers consider

$$P(C = c_i|A_1 = a_{i_1}^{(1)}, ..., An = a_{i_n}^{(n)}) = \frac{(f(A_1=a_{i_1}^{(1)},...,An=a_{i_n}^{(n)}|C=c_i) \cdot P(C=c_i)}{f(A_1=a_{i_1}^{(1)},...,An=a_{i_n}^{(n)})}$$

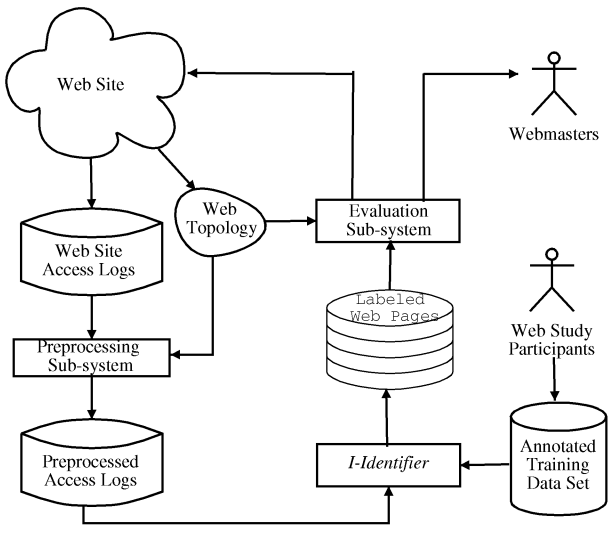

Figure 1. System Architecture

4. A Naïve Bayes Learning Based Website Reconfiguration System

The overall architecture of our Naïve Bayes Learning Based Website Reconfiguration System is depicted in Figure 1. In the Important Page Identification Sub-system (*I-Identifier*), a Naïve Bayes Classifier is developed based on an annotated log. Meanwhile, the Preprocessing Subsystem pre-processes web access logs to make the logs accessible for the classifier, since the raw web log data are ambiguous and incomplete. The web site topology is crawled and stored to aid in the data preprocessing work, which is also used later by the Evaluation Sub-system. Subsequently, the *I-Identifier* labels each entry in the preprossed web access logs *important* or *unimportant*. An Evaluation Sub-system is used to evaluate the labeled pages to decide whether they need to be reorganized. The judgements are either sent to webmasters, suggesting manual changes, or used to reconfigure web pages automatically by moving them to some level closer to the visitors' starting point. In the following sections, we present our system in detail.

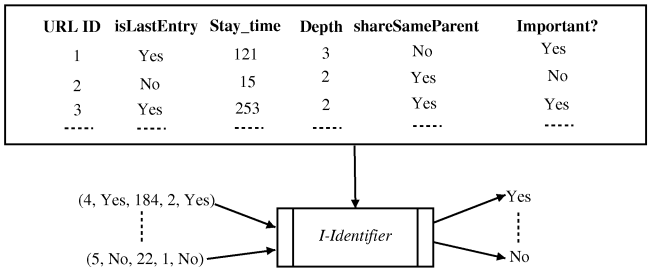

Figure 2. *I-Identifier*

4.1. *I-Identifier*

For all web pages visited by users, our goal is to identify the most important ones for the majority of visitors to fulfill their information needs. The important pages requiring a significant number of hops to reach are then considered for reconfiguration. Therefore, we use a Naïve Bayes Classifier to build a model, which can in turn identify whether any page is *important* or not. This is illustrated in Figure 2.

In order to accomplish this task, we obtained a set of annotated web logs[1] , which were collected during a web study [18]. In the study, participants were asked to access the Internet using an enhanced version of INTERNET EXPLORER, called AIE[2]. During individual visit sessions, the participants were required to indicate each page they were visiting as *important* or *unimportant*. This is done by clicking the *IMPORTANT* button on the top bar of their AIE. The URL of the current page, whether it is *important* or not, as well as other information, such as date and time of the request and referring URL, are collected by the AIE, and then recorded in the annotated web log. After some data cleaning process work (e.g., clean the unsuccessful access and assign each URL an ID), each entry of the annotated log can be represented as a matrix $m = (\vec{v_1}, \vec{v_2}, ..., \vec{v_n})$, where $\vec{v_i}$ is a vector of 4 tuples:
$\vec{v_i} = (URL, isImportant, requestTime, referringURL)$
where *isImportant* is 1 is the visitor indicated it to be important, and 0 otherwise.

Overall, 129 undergraduate business students participated in the study, which collectively requested 15,105 pages, involving 5,995 distinct URLs. The participants labelled a total of 1,887 pages as *important* pages during the study. A distinction of this dataset is its low support – 82.39% of the URLs were visited only once or twice [18]. We take this distinction into consideration when selecting learning attributes of our Naïve Bayes Classifier, as discussed below.

1 This data set is provided by Dr. Russell Greiner

2 www.cs.ualberta.ca/tszhu/software/AIE

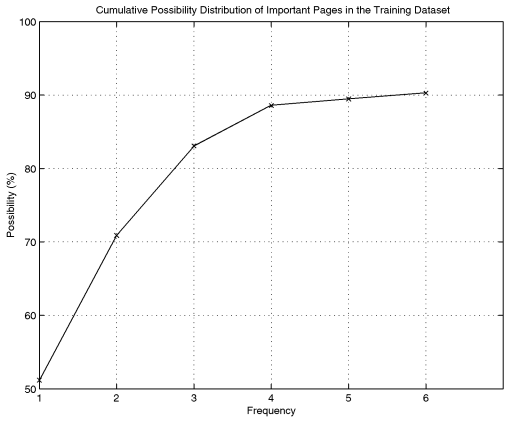

Figure 3. Cumulative Distributaion of Visit Frequency of the Annotated *Important* Pages

4.1.1. Attribute Selection As noted above, we use the annotated web logs to train the Naïve Bayes Classifier to build a model which is able to classify any page in a given web site *important* or *unimportant*. To do so, we must identify suitable attributes for training. The challenge is how to determine whether or not a certain property could help improve the performance of our classifier (or even make it worse).

The annotated logs provide 5 naïve attributes:

- Frequency: Number of times a page is visited;
- Stay_time: Time a user spends on the page during a visit;
- isFirstEntry: If the page is the initial page of a visit session or not;
- isLastEntry: If the page is the last page visited in a visit session or not;
- Depth: The number of the layer which the page belongs to in the web site hierarchy (The number of the root node is 0).

However, we need to determine whether all the attributes above are useful to build our classifier. What's more, is there more complex information we can get from the data to improve the quality of the classifier? Figure 3 shows the cumulative visit frequency distribution of *important* pages, annotated during the web study. From the figure, we can see that more than 90% of *important* pages have a visit frequency of less than 6. The low visit frequency in the annotated web logs makes the attribute *Frequency* an unsuitable attribute of our *I-Identifier*. The reason for this is that we prefer to apply our system to any generic web site, which usually includes more than thousands of clicks per week. As a result, a page with a visit frequency less than of 6 has very little chance to be considered as an *important* page. In addi-

tion, our experiment indicates that the attribute *isFirstEntry* is an irrelevant attribute for the performance of our classifier (see below for details). Thus, three attributes are kept after our preliminary analysis.

On the other hand, there are more attributes we may extract from the log data, albeit implicit. In our preliminary experiment, we attempt to add another attribute *isSameParent* to our attribute list. If a visited page shares the same parent with the previously visited one, in terms of web topology, its *isSameParent* is YES, and NO otherwise. Although this attribute is not as explicit as the previous ones to be extracted directly from the access log, we can get this attribute by combining web access logs with website topology and connectivity information. Our experiment (please refer to Table 1) show that adding this attribute significantly increases the *Predict Recall* of our classifier. Thus, we have included it in our attribute list. We will further investigate other potentially useful attributes.

4.1.2. Learning from Imbalanced Data Another challenge we face in implementing our *I-Identifier* is the imbalance of our training dataset [13], in which 13,218 out of 15,105 (87.5%) pages are annotated as negative (i.e., *unimportant*); only about 10% of the tuples are positive (i.e., *important*). In this scenario, simply labeling each instances *unimportant*, we can achieve an accuracy close to 90%. Of course, this will not serve our needs; we need to deal with the issue of learning from imbalanced data. Learning from imbalanced data can be accomplished through either sampling instances (e.g., up-sampling or under-sampling) [8] or model selection [13]. As model selection approaches usually require knowledge about the misclassification cost distribution context that is usually not known, we utilize sampling instances (more specifically, up-sampling) methods in this paper as follows (assume that we have total $I + U$ labeled web pages in our training dataset, in which I of them are labeled *important*, while U entries are labeled *unimportant*):

- Randomly choose P percent (we use 10% in our preliminary experiment) of the positive samples (PI) from the original training dataset, which are set aside as positive samples in the test dataset;
- Randomly choose the same number of negative samples (PI) from the original training data set, setting them aside as negative samples in test data set;
- Set the remaining positive samples $(1 - P)I$ in the original training data set as seed. Then, randomly choose $(1 - P)(U - I)$ samples from the seeds (with replacement), where $(1 - P)U$ is the number of remaining negative samples in the original training data set, and $(1 - P)I$ is the number of the seeds.

	Precision	*Recall*
AL_1	85.0%	53.2%
AL_2	85.0%	53.2%
AL_3	84.5%	77.0%

AL_1: (*Stay_time*, *isLastEntry*, *depth*)
AL_2: (*Stay_time*, *isFirstEntry*, *isLastEntry*, *depth*)
AL_3: (*Stay_time*, *isLastEntry*, *depth*, *isSameParent*)

Table 1. Empirical Results of *I-Identifier*

As a result, we have $(1 - P)U$ positive training data as well as $(1 - P)U$ negative training data, while have PI positive data and same number of negative data for testing.

4.1.3. Empirical Results To test the model learned in our *I-Identifier* and justify our attribute selection methodology, we performed 10-fold cross validation using the test dataset mentioned above. The results, averaged over all 10 cross validation folds, are depicted in Table 1. In the table, we measure both *Precision* and *Recall* on different combination of attributes, or attribute list (AL). *Precision* is defined as the ratio of correct labels among all pages that are labeled *important*. The higher the precision, the better the classifier. *Recall* measures the ability of our *I-Identifier* to identify *important* pages. The higher the recall, the better the classifier. With the classic confusion matrix [9] as shown below,

Actual/Predicted	Positive	Negative
Positive	*a*	*b*
Negative	*c*	*d*

Precision and *Recall* can be defined by the following formulas:

$$Precision = \frac{a}{a+c}$$

$$Recall = \frac{a}{a+b}$$

According to Table 1, we can see that attaching the attribute *isFirstEntry* (in AL_2) does not make a difference with regard to both *Precision* and *Recall*, compared with AL_1. Therefore *isFirstEntry* is an irrelevant attribute. Through adding the attribute *isSameParent* (in AL_3), however, we could achieve a higher *Recall*, but a lower *Precision*. Please note that our ultimate goal is to accurately identify all *important* pages in a web site. Thus we need a balance between *Precision* (to prevent generating misleading classification) and *Recall* (to be able to classify as many pages as possible). Therefore, we select AL_3 as the final attribute list upon which to build our model to predict *important* pages, because we can achieve a much higher *Recall* with a light loss of *Precision*.

4.2. Preprocessing Sub-system

The task of the data preprocessing sub-system is to preprocess the data recorded in a web server access log, based on which our *I-Identifier* can label each of them *important* or *unimportant*. Because different web servers may use different formats of web logs, this sub-system may need to be re-designed. We are here discussing a typical web server access log, each entry in which contains a client IP address, the requested URL, a timestamp to record the request time, referring URL, and other related information. We have identified individual users and visit sessions, as well as extracted required attributes before feeding them into our *I-Identifier*.

4.2.1. User Visit Session Identification In session-based access logs, the annotated log above serves as an example, users and visit sessions have been recorded, since users have to log-in and log-out for a visit session. However, a typical web server access log have no entries identified by users and sessions because of the anonymity of web visits. To identify individual users and visit sessions, we assume that each IP address represents a different user. The goal of visit session identification is to divide all page accessed by each user into individual sessions. Under the assumption that for logs that span long periods of time, it is very likely that users will visit the web site more than once, we use similar techniques as in [5] to identify individual sessions by idle time. That is, once the idle time between two requests exceeds a certain limit (we chose an idle time of 30 minutes, which is suggested by a former study [4]), it is assumed that the user is starting a new session.

4.2.2. Attribute Extraction In order to allow our Naïve Bayes Classifier to label each web page recorded in a web log as *important* or *unimportant*, there is further preprocessing work in our preprocessing sub-system: extracting attributes from the web access logs to match the attributes we identify in Section 4.1.1. That is, we extract *Frequency*, *Stay_time*, *isFirstEntry*, *isLastEntry*, and *isSameParent* for each entry of web server access logs.

In summary, after applying the above data preporcessing work to a web server access log, each entry of the log turns into a vector of 8 tuples:
($userId$, $sessionId$, $URLID$, $Frequency$, $stay\_time$, $isLastPage$, $depth$, $same\_parent$).

4.3. Evaluation Sub-system

After preprocessing, the preprocessed web server access logs are fed into the *I-Identifier*. *I-Identifier* labels each entry *important* or *unimportant* (See Figure 2). Please note that rather than label all pages in the web site, we label only pages recorded in access logs. Our goal is to identify the important pages which are frequently visited but difficult to access. However, only labeling web pages with the *I-Identifier* creates some difficulties in fulfilling this task. Therefore, an evaluation sub-system is implemented to perform the following processes, based on the labeled web pages – including the calculation of web page overall importance and average hops.

4.3.1. Total Occurrences and Overall Importance Our *I-Identifier* attempts to label each web page visited *important* or not. However, the same page may appear more than once in web logs, and it may sometimes be labeled as *important*, and sometimes not. Therefore, we calculate the probability of a page being labeled as an *important* page and use it as an overall evaluation of the page's importance. Given any page *P*, its overall importance *OI(P)* is computed as follows:

$$OI(P) = \frac{\text{\# of Positive Occurrences}}{\text{\# of total occurrences}}$$

where *# of positive occurrences* is the number of times page *P* were labeled as *important pages*, while *# of total occurrences* is the number of times pages *P* occur in the web access log data. We use the threshold of 80% in our experiment to give an overall evaluation of each page.

4.3.2. Average Visit Hops We then calculate the average number of hops required for visitors to reach the identified *important* pages. For example, consider that the page *P* has appeared in total *S* visit sessions, among which *P* is labeled as *important* in *S1* sessions, but as *unimportant* in *S2* sessions ($S1 + S2 = S$). The general formula for average hops to reach page *P*, *H(P)*, is defined as:

$$H(P) = \frac{\sum_{S1} Hops(P)}{S1}$$

Let us give an example to illustrate this point. Suppose page *p* appears in three visit sessions which is identified by the Preprocessing Sub-system:
Session 1: (*a*, *b*, *c*, *p*);
Session 2: (*d*, *p*);
Session 3: (*e*, *p*, *f*)
and our *I-Identifier* classes page *p* as an *important* page in Session 1 and Session 2 above. In Session 1, the user used 3 hops to reach *p*; in Session 2, only 1 hop is used to reach *p*. Therefore, the average hops to reach page *p* $H(p) = (3 + 1)/2 = 2$.

To summarize, for each entry in the preprocessed web log,
($userId$, $sessionId$, $URLID$, $Frequency$, $stay\_time$, $isLastPage$, $depth$, $same\_parent$),
the evaluation sub-system generates a corresponding vector of 4 tuples,
($URLID$, Total Occurrence, Possibility of Importance, Average Hops).

4.4. Web Page Reconfiguration

Based on the 4-tuple vector we have built for each web page, we recognize those pages which are *generally important* but difficult for users to access, as pages to be reconfigured. A page is *generally important* if its *Overall Importance* is larger than a threshold. Another threshold is then applied to these *generally important* pages, and a page is recognized as needing reconfiguration if the *Average Hops* to reach it is larger than this threshold. In addition, a third threshold is used in our system to prevent rarely visited pages being treated as *generally important* pages. In sum, the pages for which *Total Occurrence*, *Overall Importance*, and *Average Hops* are above corresponding thresholds, are identified as pages that need to be reconfigured.

Once the pages that need to be reconfigured are identified, our system provides two ways to deal with them: automatically moving them to a perfect location (or adding a shortcut link between the perfect location and current location as an alternative), or reporting them to the webmaster with a suggestion for relocation. The former is sometimes preferred because of its automation. However, deciding on the "perfect" location is a complex issue. A simple idea is that a page should be moved to some upper level in the website hierarchy. However, what if the page often occurs in the sessions which start some points below in the hierarchy? Rather than moving a recognized page to some absolute upper level, therefore,our system attempts to identify the most common "visit starting point" to reach each discovered *important* page, and then adds shortcut links between the "starting points" and corresponding page. Of course, how to accurately identify "starting points" is another open question worthy of further investigation. Our system also generates a report regarding our discovery, which is sent to the webmaster, to aid them in figuring out design problem and making the change manually. This report is also used to help us evaluate our system.

5. Experimental Evaluation

We used a generic web server access log from the Department of Computing Science, University of Alberta (www.cs.ualberta.ca) to test our system. Data was collected for 8 months (Sept. 2002 – Apr. 2003), and partitioned into months. On average, each monthly partition contains more than 40,000 pages, resulting in on average 150,000 links between them. The log of each month averaged more than 200,000 visit sessions. We ran our system on this dataset, and specified the three threshold values as follows: $Important\_threshold = 0.8$, $Length\_threshold = 5$, and $Occurrence\_threshold = 5$. In total, 125 pages are identified as needing reconfiguration.

The effect of our system in our preliminary experiment appears to be slight (125 out of about 40,000 pages), However, there is an explanation for this. First, this web site is a Computer Science department site and was well designed and maintained by computer professionals. Second, a majority of users of this web site are from the local department and they are quite familiar with the web site structure. Even so, our experiment do produce some significant discoveries. For example, the department has an internationally recognized Game research group, and many external visitors visiting the department web site were looking for the group home page. In its current design, it takes a visitor at least 4 hops to reach the Game group homepage from the department homepage (Home → Research Activities → Artificial Intelligence → Research Group Home Page → Game Group). The suggestion given by our system, adding a link to the Game group homepage in the departmental homepage, therefore provides a significant improvement in the organization of the web site.

5.1. Website Overall Structure Evaluation

In order to evaluate the effect of our reconfiguration system on the organization of a web site, we also propose a quantitative formula, which can be used to quantitatively measure its impact on the overall accessibility of the web site. We recognize that an evaluation of the global structure of a web site would be helpful. If the overall structure of the reconfigured web site based on our system is better than the initial design, we can be confident in the performance of our system. Therefore, in this section, we propose a method to evaluate the global structure of a web site, which is then used to evaluate our system.

From our point of view, a good website should help users fulfill their information needs (reach their targets) as quickly as possible. We therefore make use of the average number of hops to reach target pages over all users to evaluate a website structure. More specifically, an overall web structure of a website *W* can be measured by a score *S(W)*:

$$S(W) = \sum_k OI(k) * H_k/V$$

where k is a target page, $OI(k)$ represents the overall importance of the page k, as defined in Section 4.4, H_k is the average number of hops to reach page k, and V is the total number of web pages recorded in the web server access log.

Let us give an example to illustrate this point. Suppose that there are 4 web pages, *a*, *b*, *c*, and *d* in a web site, and the overall importance of each page is $OI(a) = 0.5$, $OI(b) = 0.7$, $OI(c) = 0.8$, and $OI(d) = 0.9$. Suppose the average hops to these pages are $H(a) = 1$, $H(b) = 2$, $H(c) = 3$, and $H(d) = 2$, respectively. In addi-

tion, all 4 pages have been visited, and therefore, have entries in the web log. Then, the score of this web site is: $(1 * 0.5 + 2 * 0.7 + 3 * 0.8 + 2 * 0.9)/4 = 1.7$.

Applying this formula to our experimental data, we get a score for the UofA CS department website of 2.114. After adding shortcut links between the recognized 125 pages needing reconfiguration and their corresponding starting points, the new score of the reconfigured web site is 1.752. This result shows that the department website is a well organized website, but our reconfiguration system can still help improve its quality.

6. Conclusion and Future Work

In this paper, we presented a framework for a Naïve Bayes Learning Based Website Reconfiguration System. The system makes use of an annotated web log to build a model that is then used to identify *important* pages in a given web site. The overall importance of web pages, along with other discovered information such as average visit hops, are used to determine the pages in the web site which are important to enable users to fulfill their information needs but difficult to access. These pages may then be reconfigured, resulting in an improvement in the overall structure of the web site. Our system can be applied to any web site, and our preliminary experiments show the applicability and potential of our system to improve the quality of web site design. A further task would be to explore identifying other implicit but useful attributes to improve our model. We would also embed with other learning models, such as decision tree, and test our system on more dataset to discover the best model for the task of important page identification as well as website reconfiguration.

7. Acknowledgments

The original work of this paper were done as a term project of the course *Machine Learning* under the instruction of Dr. Russell Greiner. The authors gratefully acknowledge Dr. Russell Greiner for his advice and support. His guidance is always invaluable, and his patience is highly appreciated. We also wish to express our gratitude to Dr. Tingshao Zhu and Dr. Tong Zheng for sharing of their knowledge and many insightful discussions. This work is in part funded by the **Alberta Ingenuity Funds**.

References

[1] C. R. Anderson, P. Domingos, and D. S. Weld. Adaptive web navigation for wireless devices. In *the Seventeenth International Joint Conference on Artificial Inelligence*, 2001.

[2] P. Berthon, L. F. Pitt, and P. T. Watson. The world wide web as an advertising medium. *Journal of Advertising Research*, 36(1):34–54, 1996.

[3] C. Borgelt. A naive bayes classifier plug-in for dataengine.

[4] L. Catledge and J. Pitkow. Characterizing browsing behaviors on the world wide web, 1995.

[5] R. Cooley, B. Mobasher, and J. Srivastava. Data preparation for mining world wide web browsing patterns. *Knowledge and Information Systems*, 1(1):5–32, 1999.

[6] R. O. Duda and P. E. Hart. Pattern classification and scene analysis, 1973.

[7] I. J. Good. An essay on modern bayesian methods, 1965.

[8] N. Japkowicz. The class imbalance problem: Significance and strategies. In *the 2000 International Conference on Artificial Intelligence (IC-AI'2000)*, pages 111–117, 2000.

[9] R. Kohavi and F. Provost. Special issue on applications of machine learning and the knowledge discovery process, 1998.

[10] P. Langley, W. Iba, and K. Thompson. An analysis of bayesian classifiers. In *The 10th National Conference on Artificial Intelligence*, 1992.

[11] J. Li and O. R. Zaiane. Combining usage, content, and structure data to improve web site recommendation. In *5th International Conference on Electronic Commerce and Web Technologies (EC-Web 2004)*, 2004.

[12] M. Perkowitz and O. Etzioni. Towards adaptive web sites: Conceptual framework and case study. *Artificial Intelligence*, 118(5):245–275, 2000.

[13] F. Provost. Learning with imbalanced data sets 101., 2000.

[14] R. Srikant and Y. Yang. Mining web logs to improve website organization. In *World Wide Web*, pages 430–437, 2001.

[15] M. K. Stern, J. E. Beck, and B. P. Woolf. Naive bayes classifiers for user modeling. In *The 7th International Conference on User Modeling*, 1999.

[16] T. Sullivan. Reading reader reaction: A proposal for inferential analysis of web server log files. In *The Web Conference '97*, 1997.

[17] H. K. Takehiro Nakayama and Y. Yamane. Discovering the gap between web site designers' expectations and users' behavior. In *9th international World Wide Web conference on Computer networks*, pages 811–822, 2000.

[18] T. Zhu, R. Greiner, and G. Haubl. Learning a model of a web user's interests. In *The 9th International Conference on User Modeling*, 2003.