Constructing Web User Profiles: A Non-invasive Learning Approach

Philip K. Chan

Computer Science, Florida Institute of Technology, Melbourne, FL 32901
pkc@cs.fit.edu
WWW home page: http://www.cs.fit.edu/~pkc/

Abstract. Our web user profiles consist of Page Interest Estimators (PIE's) and Web Access Graphs (WAG's). We discuss a non-invasive approach to estimating the user's interest of a web page without directly asking the user. A time and space efficient method is proposed for locating multi-word phrases to enrich the common bag-of-words representation for text documents. PIE's are then learned to predict the user's interest on any web page. A WAG summarizes the web page access patterns of a user. We describe how a user profile can be utilized to analyze search results and recommend new and interesting pages. Our empirical results on PIE's are encouraging.

1 Introduction

Recently researchers have started to make web browsers more *adaptive* and *personalized*. A *personalized* web browser caters to the user's interests and an *adaptive* one learns from the users' (potentially changing) access behavior. The goal is to help the user navigate the web. Lieberman's Letizia [23] monitors the user's browsing behavior, develops a user profile, and searches for potentially interesting pages for recommendations. The user profile is developed without intervention from the user (but the details of how that is performed is not clear in [23]). While the user is reading a page, Letizia searches, in a breadth-first manner, from that location, pages that could be of interest to the user. Pazzani et al.'s Syskill & Webert [28, 29] asks the user to rank pages in a specific topic. Based on the content and ratings of pages, the system learns a user profile that predicts if pages are of interest to the user. They investigated a number of topics and a different user profile is learned for each topic. Pages are recommended from preselected web sites. Similar to Syskill & Webert, Balabanovic's Fab [2] requires the user to rank pages and learns a user profile based on the rankings and content of pages. However, Fab considers recommendations based on profiles of all the other users on the system (though Fab does not identify users of similar interests). Fab's collection agent performs an off-line best-first search of the web for interesting pages.

Our approach toward an adaptive personalized web browser does not require the user to explicitly provide information to the browser. Instead, it monitors the user's access behavior and captures his/her interests in a user profile. We developed a metric for estimating interestingness of each visited page. A user profile consists of two components: a *Page Interest Estimator* (PIE) and a *Web Access Graph* (WAG). Based on the content of web pages, a page interest estimator, learned from the user's access behavior, characterizes the interests of a user. A web access graph summarizes the web page access patterns of a user. To provide personalized on-line search, our search engine consults multiple existing search engines ([38, 39]), collates the returned records, and ranks them according to the user profile. Using similar search techniques, recommendation and prefetching of interesting pages are performed off-line at night.

In this article we focus on PIE's (Section 2), and investigate how a user's interest of a page can be approximated without asking the user (Section 2.1) and how phrases can be identified to enrich the common bag-of-words representation for documents (Section 2.2). We then describe WAG's in Section 3. In Section 4 we discuss on how user profiles can be utilized in analyzing search results and recommending novel pages. We evaluate PIE learning on data from our web site in Section 5. Finally, we conclude in Section 6 with some remarks on the challenging issues that PIE learning exhibits.

2 Page Interest Estimator (PIE)

We can identify patterns in pages that constitute the user's interest. For example, certain words or phases, are of interest to the user. Given a set of labeled (interesting or not interesting) pages, we can apply learning algorithms (for example, C4.5 [32]) to induce classifiers that predict if a page is of interest to the user. These classifiers are called *Page Interest Estimators* (PIE's). More concretely,

$$Interest \leftarrow PIE_{user}(Page)$$

and PIE can be learned:

$$PIE_{user} \leftarrow MachineLearningAlgorithm(Page, Interest).$$

In addition, PIE's are learned, stored, and used at the individual users' sites, hence privacy is maintained.

Related work in this area usually takes an *invasive* approach that requires user involvement in providing ranking (for example, [2, 29]). However, as we will discuss our *non-invasive* approach in the next section (Section 2.1), the user interest of each visited page can be approximated without any user involvement. Also, how can a page be represented so that machine learning algorithms can be applied? We discuss page representation in Section 2.2.

2.1 User Interest of a Page

How do we find out if a user is interested in a page? One way is to ask the user directly (e.g., [29]). This is the common approach, but it is *invasive* and, in most cases, requires the user to provide some ad hoc ranking score (e.g., 0 to 10 or bad to good). Furthermore, the user may provide inconsistent rankings across pages and the process is time consuming. Another way is to monitor the user's behavior and evaluate the user's interest. This approach is *non-invasive* and the user is not subject to ad hoc ranking.

Throughout the course of web browsing, a user leaves behind a trail of information that can be used to model his or her interests. Four general sources of information are available: *history*, *bookmarks*, content of pages, and *access logs*.

1. A web browser usually maintains a *history* of the user's requests in the current session and in the past. The history of the current session allows the user to go back and forth between the pages he/she has visited. In addition, a global history maintains the timestamp of the last time each page is visited (this allows links to expire after a specified amount of time and be displayed as if they have not been visited). In Netscape's Navigator [27], the title, URL, first visit timestamp, last visit timestamp, expiration timestamp, and visit count of each visited URL is stored in the global history. We conjecture that a higher frequency and more recent visits of an URL indicate stronger user interest of that URL.
2. *Bookmarks* serve as a quick access point for interesting URL's chosen by the user. With a few mouse clicks, a user can easily jump to URL's in his/her bookmarks. It seems natural to assume that pages that are bookmarked are of strong interest to the user.
3. Each page usually contains links to other pages. If the page is interesting to the user, he/she is likely to visit the links referenced by the page (Lieberman [23] made a similar observation). Hence, a higher percentage of links visited from a page indicates a stronger user interest in that page. This is particularly important for index pages, which contain a lot of related links and on which the users spend less time than pages with real content.
4. Each entry in an *access log* corresponds to an HTTP request, which typically contains the client IP address, timestamp, access method, URL, protocol, status, and file size. A sample entry is:

   maelstrom.cs.fit.edu - - [19/Jun/1998:19:02:15 -0400] "GET /toc.html HTTP/1.0" 200 2540

   From these entries, time spent on each page can be calculated. The longer a user spent on a page, the likelier the user is interested in the page. If a page is not interesting, a user usually jumps to another page quickly. Experimental studies in [25, 19] confirm this observation. However, a quick jump might be caused by the short length of the page, hence the user's interest might be more appropriately approximated by the time spent on a page normalized by the page's length. We note that activities other than surfing the web (e.g., answering a phone call) can inadvertently be included in the time spent on a page. We cannot avoid this problem without a more complicated way of obtaining time statistics (e.g., tracking the gaze of user's eyes [42, 43]). However, to reduce the problem, we impose an upper limit on time (e.g., 15 minutes) spent on a page and time intervals of more than the upper limit are considered as separate sessions. Unfortunately, browsers usually have a *history*, but not an *access log* (which is essentially a more detailed history) since a history is sufficient for traversing pages in a session and maintaining timestamps for link expiration. Access logs are usually found in HTTP servers. In order to maintain an access log for the browser client, one can modify the source code of the browser (e.g., [26]). However, this requires changes to a complex piece of software and is browser-dependent. A simpler approach is to use a web proxy server that logs HTTP requests (e.g., [41]). A proxy serves as a relay between the browser client and the Web—it is usually used for security (clients behind firewalls), performance (system-wide caching of external pages), and/or filtering (blocking out undesirable sites).

User Interest Approximation Given the above four sources of information, we can devise a measure for approximating the interest of a page to a user. One simple measure is:

$$Interest(Page) = Frequency(Page), \tag{1}$$

where $Frequency(Page)$ is the frequency of $Page$ visited by the user. We consider the number of visits a primary indicator of interest. A more sophisticated measure uses $Frequency(Page)$ as a base and incorporates all the factors discussed above:

$$\begin{aligned} Interest(Page) = {} & Frequency(Page) \times (1 + IsBookmark(Page) + \\ & Duration(Page) + Recency(Page) + LinkVisitPercent(Page)), \end{aligned} \tag{2}$$

where

$$IsBookmark(Page) = \begin{cases} 1 \text{ if } page \text{ is a bookmark} \\ 0 \text{ otherwise} \end{cases},$$

$$Duration(Page) = \frac{TotalDuration(Page)/Size(Page)}{\max_{Page \in VisitedPages}(TotalDuration(Page)/Size(Page))},$$

$$Recency(Page) = \frac{Time(LastVisit) - Time(StartLog)}{Time(Now) - Time(StartLog)}, and$$

$$LinkVisitPercent(Page) = \frac{NumberOfLinksVisited(Page)}{NumberOfLinks(Page)}.$$

The maximum value of $Interest(Page)$ is $Frequency(Page) \times 5$.

All the visited pages can be considered interesting to various degrees since the user accessed them. However, how do we find pages that are not interesting to the user? It is easier if the user is required to rank pages, but the user is not actively involved in our case. Furthermore, we cannot assume any page not visited on the web is of no interest to the user because he or she might not know of its existence (not to mention the staggering number of pages on the web). Since pages usually contain links to other pages and, in most cases, not all of them are followed by the user, one approach to identifying pages not interesting to the user is to consider links in visited pages that are not followed by the user. Related work is in text categorization, where documents are mapped into categories using learned models [1, 22, 47]. Usually, the documents are grouped into many categories and all the documents are known in advance. Our task, however, is to group the pages into two categories (interesting or not interesting) and cannot assume that all the pages on the web are known in advance.

Evaluation of user interest approximation Based on the frequency of visits, in Equation 2, we value each factor equally. Some weighted scheme will likely be more appropriate after we perform some experiments to validate this model. Such experiments will involve surveying the user's actual degree of interest and measuring the square errors. Regression techniques can be used to tune the weights in our model.

2.2 Page representation

Various representations of a web page have been widely studied. Most researchers use the vector-space model pioneered by Salton [36]. In this model each document is represented by a vector of weights, each of which corresponds to a feature, a word in most cases. The word ordering information in the document is usually not retained and hence the name "bag-of-words."

However, much research focuses on single words (unigrams) as features. This is partly due to the large combination of possible multi-word phrases. Another reason is that earlier results from "syntactic" and "statistical" phrases were mixed [11]. We suspect that the ad hoc way of constructing statistical phrases might have been a problem [13]. Much of the statistical work in building multi-word features focuses on co-occurrence (e.g., [9]), that is, if two words appear frequently together, they probably form a phrase. One co-occurrence metric is *mutual information*. Consider a and b are two words within a window of some size, the mutual information:

$$MI(a,b) = \log \frac{P(a,b)}{P(a)P(b)} \tag{3}$$

measures the reduction of uncertainty in knowing b's presence in the window if a's presence is known (or vice versa, the metric is symmetric). However, this metric does not consider the effect of the absence of either or both words in the window. Note that if two words always appear together or not at all, they are more likely to be a phrase than other situations. *Expected (or average) mutual information* [34, 35] (or *information gain* [32]) captures the effects of word absences.

$$EMI(A,B) = \sum_{a,\overline{a}\in A} \sum_{b,\overline{b}\in B} P(A,B) \log \frac{P(A,B)}{P(A)P(B)} \tag{4}$$

measures the expected mutual information of the four combinations of the presence and absence of a and b. Although larger $MI(a,b)$ and $MI(\overline{a},\overline{b})$ provide

PALO ALTO, Calif. – The ink on the first draft of the history of the Internet stock boom isn't yet dry, and the revisionists are already having their say. Even as Internet shares were tumbling from their highs, three of the companies backed by Accel Partners, a venture capital firm, went public last month. Though their net losses total $56 million, the three now have a combined market capitalization of $10 billion. "Astounding," James W. Breyer, the firm's managing partner, calls that performance, echoing the giddiness common in Silicon Valley when people talk about the Internet.

Fig. 1. First Paragraph of "Feeding a Frenzy: Why Internet Investors Are Still Ravenous[18]"

Table 1. Top twelve bigrams selected by MI, EMI, and $AEMI$

Metric	Top twelve bigrams (in ranked order)
MI	schwab dlj, ruth porat, research report, fiercest competitor, dlj direct, charl schwab, wit group, san francisco, p o, net loss, menlo park, equal stand
EMI	ventur capitalist, silicon vallei, new york, wall street, m whitman, go public, morgan stanlei, invest banker, doesn t, san francisco, p o, menlo park
$AEMI$	ventur capitalist, silicon vallei, invest banker, new york, go public, m whitman, morgan stanlei, wall street, invest bank, internet stock, doesn t, york time

more evidence for "ab" to be a phrase, larger $MI(a,\overline{b})$ and $MI(\overline{a},b)$ supply more counter evidence. Therefore, we introduce *augmented expected mutual information* (AEMI) which appropriately incorporates the counter-evidence:

$$AEMI(A,B) = \sum_{(A=a,B=b),(A=\overline{a},B=\overline{b})} P(A,B)\log\frac{P(A,B)}{P(A)P(B)} - \sum_{(A=a,B=\overline{b}),(A=\overline{a},B=b)} P(A,B)\log\frac{P(A,B)}{P(A)P(B)} \qquad (5)$$

In essence supporting evidence is summed, while damaging evidence is subtracted. Furthermore, we define A as the event of the first word, and B as the event in the words (of some window size W) following the first word. That is, a higher value of $AEMI$ indicates a is likely followed by b and one is less likely to be present when the other is absent. Moreover, window size W allows flexibility in the number of gaps between words in a phrase.

Evaluating the variants of mutual information We tested MI, EMI, and $AEMI$ on a New York Times article on the recent frenzy of internet companies going public in the stock market [18]. Table 1 lists the top twelve bigrams (two-word phrases) selected by the three metrics (words were stemmed by Porter's algorithm [31, 14] beforehand). We observed that the bigrams selected by MI are not as relevant to the article as EMI and $AEMI$, which are quite similar for the top twelve bigrams. However, while "internet stock," a key phrase, was ranked 10th by $AEMI$, it was ranked 37th by EMI. Another relevant bigram, "internet compani" was ranked 33rd by $AEMI$, but lower than 65th by EMI. This provides some empirical evidence for $AEMI$'s ability to find more relevant phrases than the other two metrics.

Efficient identification of n-grams Using $AEMI$ with a threshold, we can find highly probable two-word phrases (bigrams) from a training corpus. n-word phrases (n-grams) can be found using the same method—event A is the first $n-1$ words and event B is a word in the following W (window size) words. However, the memory requirement of storing the necessary statistics to find n-grams is $O(s^n)$, where s is the number of unique words in the corpus; this could be prohibitive even for locating trigrams. Consider that s is 1,000 (a relatively small number) and each counter takes 1 byte, one gigabytes are needed to find trigrams! However, to reduce the combinatorial explosion, one can safely consider only a small number of bigrams with high $AEMI$ values as event A. Though this scheme is memory efficient, it requires a second pass of the corpus, which incurs disk I/O time, since the $AEMI$ values can only be calculated at the end of one pass. This scheme requires $n-1$ passes for identifying n-grams.

We propose an approximate approach to finding n-grams without requiring a lot of storage and multiple passes on the disk-resident corpus. For each bigram with $AEMI$ above some threshold T, a directed edge is inserted into a graph, whose vertices are the words. A trigram, "*abc*," is identified if the edges $a \to b$, $b \to c$, and $a \to c$ exist. Similarly, a quadgram, "*abcd*," is located if, in addition to the three edges for trigram "*abc*," the edges $a \to d$, $b \to d$, and $c \to d$ also exist. Our scheme needs $O(s^2)$ storage ($O(s^2)$ for the counters plus $O(s^2)$ for the graph, which is sparse and requires much less than s^2 storage) and a single pass on the corpus. Formally, $ngram$ is defined as:

$$ngram(w_1, w_2, ..., w_n) = \begin{cases} edge(w_1, w_2) & \text{if } n = 2 \\ ngram(w_1, w_2, ..., w_{n-1}) \wedge \bigwedge_{i=1}^{n-1} edge(w_i, w_n) & \text{if } n > 2, \end{cases} \tag{6}$$

where

$$edge(w_i, w_j) = \begin{cases} true & \text{if } AEMI(w_i, w_j) > T \\ false & \text{otherwise} \end{cases}$$

Window size W plays a role in our method of building n-grams. Given a W, our scheme allows larger gaps in shorter phrases and smaller gaps in longer phrases. This notion stems from the observation that shorter phrases might have additional intervening words but longer phrases usually do not. For instance, when W is set to 2 for building n-grams upto trigrams, we allow an additional word between the words in bigrams but none in trigrams.

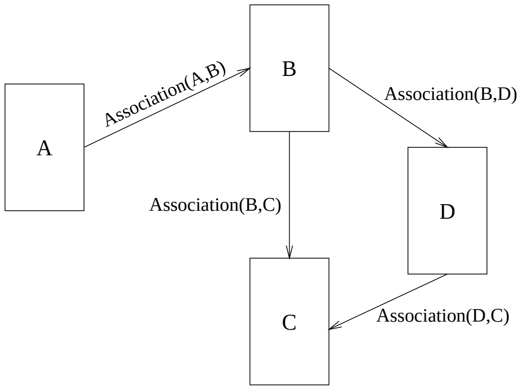

Fig. 2. A sample Web Access Graph (WAG)

3 Web Access Graph (WAG)

In addition to the Page Interest Estimator (PIE), our user profile contains a *Web Access Graph* (WAG). A WAG is a weighted directed graph that represents a user's access behavior (a similar but simpler graph is described in [45]). Each vertex in the graph represents a web page and stores the access frequency of that page. If the user visited $Page_A$ followed by $Page_B$, a directed edge exists from $Page_A$ to $Page_B$. For example, Figure 2 shows that the user traversed two sequences of web pages: $A \rightarrow B \rightarrow C$ and $A \rightarrow B \rightarrow D \rightarrow C$. The weight on each edge represents the degree of association between the two pages or the likelihood of accessing $Page_B$ from $Page_A$. One measure for association is the frequency of visits from one page to the other. That is,

$$Association(Page_A \rightarrow Page_B) = Frequency(Page_A \rightarrow Page_B), \qquad (7)$$

where $Frequency(Page_A \rightarrow Page_B)$ is the number of visits from $Page_A$ to $Page_B$. A user's web access graph can be displayed graphically to help the user locate related interesting pages that the user has visited [45]. It enhances the bookmark facility, which merely records what the user wants to store. Graphically displaying the web access graph provides a more user friendly and informative interface.

For instance, the intensity of a vertex indicates the interest level of the corresponding page and the thickness of an edge depicts the degree of association. Since a user can visit many pages, a web access graph can be quite complex to display. One approach to solve this problem is to partition the graph by clustering vertices/pages and display each subgraph separately.

3.1 Clustering

Clustering techniques can be used to find pages that are closely associated with each other and are likely to be accessed by the user consecutively. Before clustering algorithms can be applied, we need an undirected graph and a distance metric. The weighted directed web access graph can be converted into a weighted undirected graph. For each pair of vertices, the two directed edges (if one or both exist) are replaced by an undirected edge. The association of the undirected edge is the sum of the association of the directed edges:

$$\begin{aligned} Association(Page_A, Page_B) = {} & Association(Page_A \rightarrow Page_B) + \\ & Association(Page_B \rightarrow Page_A). \end{aligned} \tag{8}$$

The distance metric is the reciprocal of association.

Given a certain k, k-means clustering [16] yields k clusters. Since a user cannot effectively visualize a cluster that is too large, cluster size should be limited to some reasonable number (e.g., 30). Consequently, we have a way to determine k. Moreover, we can use the k most interesting pages, instead of random pages, to seed the centroids of the initial clusters. Another clustering method can be utilized if the user's bookmarks are organized into folders. Each folder corresponds to a cluster and the bookmarked pages are not moved from the clusters. Pages that are not bookmarked are assigned to the closest cluster measured by the average distance to each bookmarked page in the cluster. A maximum cluster size is maintained and only the closest pages are kept in the cluster. Pages that are not in any clusters go through the k-means clustering process. Besides k-means clustering, we plan to investigate hierarchical agglomeration clustering algorithms and the connected-component method used in [30].

The generated clusters can be ranked according to the average interest, average pair-wise distance, or a combination of the two in each cluster. This ranking provides a way of ordering the clusters to be indexed and displayed. Besides rendering a cluster graphically, we can also display it textually in the form of a hierarchical list. Starting from the page with the highest interest as the root, we can apply a minimal spanning tree algorithm to generate the hierarchical list.

Evaluating Clusters How do we know the generated clusters are indeed useful to help the user navigate? One evaluation metric is to measure how likely it is that when a page in a cluster is referenced, then another page (or other pages) in the same cluster is also accessed in the same session [30]. An alternative measure is the number of pages visited by the user in a cluster during a session. Furthermore, in order to evaluate the predictiveness of the clusters, we evaluate on sessions that are disjoint from sessions used in clustering. This evaluation metric can also be used to rank clusters for display purposes.

3.2 Alternatives for Association

As discussed above, $Association(Page_A \rightarrow Page_B)$ can be simply approximated by the frequency of access from $Page_A$ to $Page_B$. However, we observe that a user would have a stronger desire to visit $Page_B$ if it is more interesting. Hence, a more sophisticated association metric considers the interest (Section 2.1) of the destination page and is defined as:

$$Association(Page_A \rightarrow Page_B) = P(Page_A | Page_B) \times Interest(Page_B), \tag{9}$$

where $P(Page_A|Page_B)$ is the probability of $Page_A$ being the source page given $Page_B$ is the destination page and is approximated from the user's access frequencies. In other words, the interest at a destination vertex (page) is spread backward to the incoming edges. This way, an edge leading to a more interesting vertex has a larger weight and hence a stronger directed association between the two pages. (Note that if $Interest(Page)$ is simply $Frequency(Page)$, $Association(Page_A \rightarrow Page_B)$ degenerates back to $Frequency(Page_A \rightarrow Page_B)$.) We next discuss how a user profile can be utilized.

4 Utilization of a User Profile

A user profile contains a Web Access Graph (WAG) and a Page Interest Estimator (PIE). The profile provides a personalized context for analyzing results from search engines (Section 4.1) and recommending interesting pages (Section 4.2).

4.1 Analysis of search results

In order to provide a high success rate of returning relevant pages from search engines, we propose to use the user profile as a personalized filtering and ranking mechanism. Given a search query from the user, our search engine forwards the query to multiple search engines (similar to MetaCrawler [39]) and collates the returned records by removing duplicates. The PIE then determines if the pages are pertinent to the user preferences; i.e., $Ranking(Page) = PIE(Page)$. If the PIE is a classifier and provides confidence values of its classification, the pages are ranked accordingly. If the PIE is a regressor, the pages are sorted according to the estimated interest values.

Since the user does not want to wait too long for a search engine to return results, we need to consider the speed performance of various classifiers and regressors. Typically, 30 seconds should be the maximum amount of time to wait. We plan to employ multiple processors to collect, collate, and analyze the returned records in parallel. Each processor collects results from a different search engine. The results are collated using a "tournament-like" algorithm in a binary-tree manner—results from different search engines are at the leaves and pairs of results are merged recursively until the root is reached. In the process, we marked the pages that are purged on each processor. We then use the PIE to rank the collated pages in parallel. Using a "merge-sort-like' algorithm, ranked pages on each processor are recursively merged into the final sorted list of ranked pages.

Using different learning algorithms and combining the learned models has been demonstrated to be effective in improving performance [3, 4, 8, 37, 46]. We plan to investigate using multiple estimators to analyze and combine the results in parallel. Meta-learning [6] is one combining approach and we intend to examine other voting-based methods [7].

Properly evaluating how the user responds to the results returned by our search engine is important. In information retrieval, the common metrics are *precision* and *recall*. From all pages returned by the search engine, *precision* measures the percentage of pages that are relevant. From all possible relevant pages, *recall* calculates the percentage of relevant pages returned by the search engine. There is usually a trade-off between precision and recall. By varying the parameters of the search engine, different pairs of precision and recall can be obtained. Graphs can then be plotted to compare the different methods. Alternatively, some researchers use the *break-even* point [1] as a single value for comparison. The break-even point occurs when precision is the same as recall. The F-measure [21] allows weighting precision and recall differently. In our task, since we do not know all the relevant web pages that are of interest to the user, measuring recall is quite impossible. Furthermore, our search engine provides a ranking of pages, hence it considers some pages are more relevant than the others to the user. Another issue is which pages are considered relevant and of interest to the user. To measure the performance of our search engine, we propose to use $Interest(Page)$ (Section 2.1) to measure the relevance and calculate the sum of the interest values weighted by the ranking of pages. We then normalize the sum by the summation of rankings. (Note that if a page is not visited by the user, $Interest(Page)$ is zero.) Formally, the performance of our search engine is defined as:

$$\frac{\sum_{Page \in ReturnedPages} (Interest(Page) \times Ranking(Page))}{\sum_{Page \in ReturnedPages} Ranking(Page)}, \tag{10}$$

where $Ranking(Page)$ is the rank assigned by our search engine to $Page$ and is larger if it is more relevant to the user. Hence, the metric imposes larger reward/penalty to higher ranked pages and higher reward to pages that are more interesting to the user.

The user's behavior on the results returned by the search engine provides a feedback for the system to improve the user profile. Accordingly, the visited pages update the web access graph and become positive examples. On the other hand, the ignored (not visited) pages become negative examples. An updated PIE is then learned.

4.2 Recommendation of new and interesting pages

Recommending new and potentially interesting pages is a challenging problem mainly due to the large number of available pages. One approach is to perform a systematic search starting from terms that are of interest to the user or from pages visited by the user recently and recursively visit the referenced links [2, 23]. Potential interest of each page is estimated by PIE's. This approach necessitates large amounts of communication and computation in retrieving pages and analyzing them using PIE's. Alternatively, we propose to use a *collaborative* [15, 19] (or *social* [40]) approach. This approach assumes that users with comparable interests behave similarly. Hence, recommendations are based on unseen items seen by different but similar users. Related work compares users of how they rank music [40] and movies [17]. However, our non-invasive goal precludes asking the user to rank pages. We have already identified, in Section 2.1, how to estimate an user's interest of a page. Moreover, related work mainly investigates techniques in a relative small collection of items. Here, we try to examines techniques for arbitrary web sites. For efficiency reasons, we propose to analyze access behavior of web sites (domain names) instead of web pages. Here, we address how to find users with similar interests efficiently and which pages to recommend.

From a user's WAG, we can obtain the access frequency of visited sites. To identify users with similar behavior efficiently, we divide the process into two phases: the first provides a coarse filter and the second examines the degree of similarity. Phase one ignores the frequency information and calculates the degree of overlap between two users' visited sites and the top overlapping users are passed to the second phase. Phase two uses the frequency information to measure the similarity between two users. One metric is the Pearson r correlation coefficient [44]:

$$r = \frac{\sum xy - \sum x \sum y/n}{\sqrt{[\sum x^2 - (\sum x^2/n)][\sum y^2 - (\sum y^2/n)]}}, \tag{11}$$

where x and y are the paired values and n is the number of pairs. This calculates the degree of correlation between the site access frequency of two users. Moreover, instead of using all the users [33, 40], we select users that are correlated with high confidence according to:

$$t = \frac{r}{\sqrt{(1-r^2)/(n-2)}}, \tag{12}$$

which is t-distributed with $n-2$ degrees of freedom [44]. Among the selected similar users (S), we weight each page's interest by the correlation coefficient:

$$Recommendation(Page) = \sum_{s \in S} r(CurrentUser, s) \times Interest_s(Page). \tag{13}$$

Furthermore, we propose to convert the frequencies into probabilities and use Kullback divergence [20]:

$$\sum_x P(x) \log \frac{P(x)}{P_a(x)}, \tag{14}$$

to measure the degree of similarity. P is the original probability distribution (current user) and P_a is the approximate one (one of the other users). Pearson correlation has an underlying assumption of a linear relationship between the paired values if they are correlated; however, Kullback divergence does not have this assumption and hence might provide a closer estimate of similarity in our task.

5 Preliminary Experiments on PIE's

To evaluate the performance of Page Interest Estimators (Section 2), we conducted experiments on data obtained from our departmental web server. We

Table 2. Accuracy performance of PIE's learned from four algorithms and fifteen users

	Train	Test	Words and Phrases				Words Only			
User	size	size	C4.5	CART	BAYES	RIPPER	C4.5	CART	BAYES	RIPPER
1	102	114	67.5	73.7	58.8	73.7	69.3	73.7	57.0	73.7
2	148	162	77.2	74.7	67.3	74.7	80.2	71.6	72.8	74.1
3	106	76	80.3	77.6	54.0	78.9	82.9	84.2	57.9	78.9
4	68	96	67.7	58.3	57.3	64.6	70.8	61.5	58.3	66.7
5	52	64	60.9	64.1	59.4	67.2	60.9	59.4	54.7	65.6
6	80	62	58.1	58.1	59.7	58.1	74.2	67.7	59.7	58.1
7	86	150	54.7	65.3	54.0	65.3	60.0	65.3	53.3	65.3
8	42	70	58.6	48.6	54.3	64.3	55.7	48.6	54.3	64.3
9	44	46	65.2	69.6	65.2	69.6	71.7	69.6	54.3	69.6
10	128	80	82.5	76.2	60.0	77.5	76.2	77.5	66.2	70.0
11	38	36	75.0	80.6	72.2	69.4	83.3	75.0	58.3	69.4
12	64	116	69.0	67.2	53.5	70.7	65.5	56.0	51.7	52.6
13	46	196	83.7	84.2	54.1	84.2	82.7	84.2	55.1	84.2
14	44	112	59.8	66.1	65.2	67.0	61.6	60.7	58.0	67.0
15	76	76	80.3	80.3	64.5	80.3	80.3	80.3	63.2	80.3
Avg.	74.9	97.1	69.4	69.6	60.0	71.0	71.7	69.0	58.3	69.3

analyze how accurate PIE's are in predicting if a web page is interesting to a user. By analyzing the server access log from January to April 1999, we identified hosts that accessed our site at least 50 times in the first two months and also in the second two months. We use data from the first two months for training and the last two months for testing. We filtered out proxy, crawler, and our computer lab hosts, and identified "single-user" hosts, which are at dormitory rooms and a local company.

For each text web document (.html or .txt), we first extracted words, then applied a stop list to remove common articles, prepositions, and verbs, and finally stemmed the words according to Porter's stemming algorithm [31, 14]. Bigrams and trigrams were identified using our scheme described in Section 2.2. The threshold T was .0025 and the window size W was 2 (i.e., the words in a bigram might not be next to each other, but those in a trigram must be adjacent to each other in the text). Two hundred and fifty Boolean features of the presence and absence of words/phrases were selected based on *expected mutual information* (Equation 4) [48] and ties were broken with preference to features with higher document frequency in interesting documents and features from longer phrases. Moreover, accessed pages were considered interesting to the user and pages not accessed were not interesting. For simplicity, we randomly picked unaccessed pages to be included in training and testing. The number of unaccessed pages is the same as accessed pages, hence the class ratio is 1:1 in the training and test sets.

We ran C4.5 [32], CART [5], naive BAYES [12], and RIPPER [10] on the data set. Table 2 has two groups of columns on the right: one for results from features with words and phrases and the other for features with words only. The two groups are for comparing the utility of adding phrases into the feature sets (in both groups, only 250 selected features were used in training and testing). For all four algorithms, based on the paired t-test with 95% confidence, the difference in accuracy is significant—adding phrases led to higher accuracy in CART, BAYES, and RIPPER, but lower accuracy in C4.5. For User 12, phrases improved the performance of CART and RIPPER by more than 10%. This provides empirical evidence that phrases can improve the accuracy performance of learned PIE's. Among the algorithms, RIPPER achieved significantly higher accuracy in the first group and C4.5 in the second group (paired t-test with 95% confidence), while BAYES was consistently less accurate than the other algorithms. The overall top performer was C4.5 with word-only features. On average, except for BAYES, 70% (significantly higher than the default 50%) accuracy can be obtained. Finally, we would like to point out that the success of our approach largely depends on the user's behavior pattern. For instance, although User 13 has one of the smallest training sets and the largest test set, many of the algorithms achieved the highest accuracy on this user. Inspection of the generated trees and rules reveals that the user has a consistent appetite in pages that contain the word "java."

6 Concluding Remarks

We discussed a non-invasive learning approach to constructing web user profiles. Each user profile has a Page Interest Estimator (PIE) and a Web Access Graph (WAG). The construction of PIE's and WAG's are based on machine learning techniques. Various techniques were proposed to address the different research issues. We also describe how analyzing search results, recommending novel pages, and presenting information can benefit from user profiles.

We demonstrated that our proposed technique for locating multi-word phrases is time and space efficient and the reported preliminary empirical results indicate that our phrase-finding metric compares favorably to existing ones. Our results also show that PIE's can predict, with an average accuracy of 70% (significantly higher than the default 50%), whether a page will be visited by a user. Furthermore, including phrases as features to enrich the common bag-of-words representation of text documents can improve predictive accuracy.

More investigation will be conducted to evaluate and validate our proposed techniques; additional research issues will also be studied. Most machine learning tasks (including text categorization) assume all the features are known beforehand and are applicable to both the training and test sets. However, in PIE learning, words in the documents change over time, and so do the user's preferences. A monolithic classifier/regressor with a fixed set of features will not be able to adapt to these changes. We plan to investigate a multi-classifier approach with a different feature set for each classifier that is learned in a particular time period. The ensemble can then be combined via meta-learning [7].

Besides identifying phrases, other feature construction techniques can be used to enrich the bag-of-words representation. For instance, lexical knowledge from WordNet [24] can be used to group words semantically into synonym sets. Also, we can pay more attention to words in html tags like $< title >, < b >, < i >$...

In our experiments a page that is visited multiple times has only one record in the training or test sets. Removing duplicate instances from the training set is common in machine learning, mainly for efficiency reasons. However, in PIE learning, weighting pages proportional to the number of visits during training, in an effort to emphasize the relevant features in repeatedly visited pages, could enhance performance. Similarly, during testing, pages that are visited multiple times can be weighted accordingly to closely approximate the performance of PIE's.

Due to the high dimensionality of PIE learning (and related text retrieval tasks), feature selection is necessary for many learning algorithms. Besides using expected mutual information for feature selection, domain-specific heuristics like document frequency of features corresponding to interesting pages is beneficial since features identifying interesting pages are more valuable than those associated with uninteresting pages—the first kind is much rarer than the second and we want to find interesting pages, not the uninteresting ones. Also, multi-word phrases are intuitively more valuable than individual words.

Furthermore, our user profiles can be used on the server side as well. They can guide the customization of web pages displayed to the user (e.g., putting frequently accessed links at the top) and recommendation of pages that are of interest to similar users (collaborative filtering).

References

1. C. Apte, F. Damerau, and S. Weiss. Towards language independent automated learning of text categorization models. In *Proc. ACM SIGIR-94*, pages 23–30, 1994.
2. M. Balabanovic. An adaptive web page recommendation service. In *Proc. 1st Intl. Conf. Autonomous Agents*, pages 378–385, 1997.
3. L. Breiman. Bagging predictors. *Machine Learning*, 24:123–140, 1996.
4. L. Breiman. Stacked regressions. *Machine Learning*, 24:41–48, 1996.
5. L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. *Classification and Regression Trees*. Wadsworth, Belmont, CA, 1984.
6. P. Chan and S. Stolfo. Meta-learning for multistrategy and parallel learning. In *Proc. Second Intl. Work. Multistrategy Learning*, pages 150–165, 1993.
7. P. Chan and S. Stolfo. A comparative evaluation of voting and meta-learning on partitioned data. In *Proc. Twelfth Intl. Conf. Machine Learning*, pages 90–98, 1995.
8. P. Chan, S. Stolfo, and D. Wolpert, editors. *Working Notes for the AAAI-96 Workshop on Integrating Multiple Learned Models for Improving and Scaling Machine Learning Algorithms*, Portland, OR, 1996. AAAI.
9. L. Chen and K. Sycara. Webmate: A personal agent for browsing and searching. In *Proc. 2nd Intl. Conf. Autonomous Agents*, pages 132–139, 1998.

10. W. Cohen. Fast effective rule induction. In *Proc. 12th Intl. Conf. Machine Learning*, pages 115–123, 1995.
11. B. Croft, H. Turtle, and D. Lewis. The use of phrases and structure queries in information retrieval. In *Proc. SIGIR-91*, pages 32–45, 1991.
12. R. Duda and P. Hart. *Pattern classification and scene analysis*. Wiley, New York, NY, 1973.
13. J. Fagan. *Experiments in Automatic Phrase Indexing for Document Retrieval*. PhD thesis, Linguistics, Cornell Univ., Ithaca, NY, 1987.
14. W. Frakes and R. Baeza-Yates, editors. *Information retrieval: data structures and algorithms*. Prentice Hall, Englewood Cliffs, NJ, 1992.
15. D. Goldberg, D. Nichols, B. Oki, and D. Terry. Using collaborative filtering to weave an information tapestry. *Comm. ACM*, 35(12):61–70, 1992.
16. J. Hartigan. *Clustering algorithms*. Wiley, New York, NY, 1975.
17. W. Hill, L. Stead, M. Rosenstein, and G. Furnas. Recommending and evaluating choices in a virtual community of use. In *Proc. ACM CHI-95*, pages 194–201, 1995.
18. L. Holson. Feeding a frenzy: Why internet investors are still ravenous. New York Times, June 6 1999.
19. J. Konstan, B. Miller, D. Maltz, J. Herlocker, L. Gordon, and J. Riedl. GroupLens: Applying collaborative filtering to usenet news. *Comm. ACM*, 40(3):77–87, 1997.
20. S. Kullback. *Information Theory and Statistics*. Dover, New York, NY, 1968.
21. D. Lewis and W. Gale. A sequential algorithm for training text classifiers. In *Proc. ACM SIGIR-94*, pages 3–12, 1994.
22. D. Lewis, R. Schapire, J. Callan, and R. Papka. Training algorithms for linear text classifiers. In *Proc. ACM SIGIR-96*, pages 298–306, 1996.
23. H. Lieberman. Letizia: An agent that assits web browsing. In *Proc. IJCAI-95*, 1995.
24. G. Miller. WordNet: A lexical database for English. *Comm. ACM*, 38(11):39–41, 1995.
25. M. Morita and Y. Shinoda. Information filtering based on user behavior analysis and best match text retrieval. In *Proc. SIGIR-94*, pages 272–281, 1994.
26. Netscape. Netscape Mozilla. http://www.mozilla.org/.
27. Netscape. Netscape Navigator. http://www.netscape.org/.
28. M. Pazzani and D. Billsus. Learning and revising user profiles: The identification of interesting web sites. *Machine Learning*, 27:313–331, 1997.
29. M. Pazzani, J. Muramatsu, and D. Billsus. Syskill & Webert: Identifying interesting web sites. In *Proc. AAAI-96*, 1996.
30. M. Perkowitz and O. Etzioni. Adaptive web sites: Automatically synthesizing web pages. In *Proc. AAAI-98*, 1998.
31. M. Porter. An algorithm for suffix stripping. *Program*, 14(3):130–137, 1980.
32. J. R. Quinlan. *C4.5: programs for machine learning*. Morgan Kaufmann, San Mateo, CA, 1993.
33. P. Resnick, N. Iacovou, M. Sushak, P. Bergstrom, and J. Riedl. GroupLens: An open architecture for collaborative filtering of netnews. In *Proc. CSCW-94*, 1994.
34. C. Van Rijsbergen. *Information Retrieval*. Butterworths, London, 1979.
35. R. Rosenfeld. A maximum entropy approach to adaptive statistical language modeling. *Computer, Speech, and Language*, 10, 1996.
36. G. Salton. *Automatic Text Processing*. Addison-Wesley, Reading, MA, 1988.
37. R. Schapire. The strength of weak learnability. *Machine Learning*, 5:197–226, 1990.
38. E. Selberg and O. Etzioni. Multi-service search and comparison using the metacrawler. In *Proc. WWW4*, 1995.

39. E. Selberg and O. Etzioni. The metacrawler architecture for resource aggregration on the web. *IEEE Expert*, 12(1):8–14, 1997.
40. U. Shardanand and P. Maes. Social information filtering: Algorithms for automating "word of mouth". In *Proc. ACM CHI-95*, pages 210–217, 1995.
41. Squid. Squid internet object cache. http://squid.nlanr.net/Squid/.
42. R. Stiefelhagen, M. Finke, J. Yang, and A. Waibel. From gaze to focus of attention. In *Proc. Work. Perceptual User Interfaces*, pages 25–30, 1998.
43. R. Stiefelhagen, J. Yang, and A. Waibel. Tracking eyes and monitoring eye gaze. In *Proc. Work. Perceptual User Interfaces*, pages 98–100, 1997.
44. F. Wall. *Statistical Data Analysis Handbook*. McGraw-Hill, New York, NY, 1986.
45. A. Wexelblat and P. Maes. Footprints: Visualizing histories for web browsing. http://wex.www.media.mit.edu/people/wex/Footprints/footprints1.html, 1997.
46. D. Wolpert. Stacked generalization. *Neural Networks*, 5:241–259, 1992.
47. Y. Yang. An evaluation of statistical approaches to text categorization. Technical Report CMU-CS-97-127, CMU, Pittsburgh, PA, 1997.
48. Y. Yang and J. Pedersen. A comparative study on feature selection in text categorization. In *Proc. Intl. Conf. Machine Learning*, 1997.