Mining Longest Repeating Subsequences to Predict World Wide Web Surfing

James Pitkow
Xerox PARC
Palo Alto, CA 94304 USA
pitkow@parc.xerox.com

Peter Pirolli
Xerox PARC
Palo Alto, CA 94304
pirolli@parc.xerox.com

Abstract

Modeling and predicting user surfing paths involves tradeoffs between model complexity and predictive accuracy. In this paper we explore predictive modeling techniques that attempt to reduce model complexity while retaining predictive accuracy. We show that compared to various Markov models, longest repeating subsequence models are able to significantly reduce model size while retaining the ability to make accurate predictions. In addition, sharp increases in the overall predictive capabilities of these models are achievable by modest increases to the number of predictions made.

1. Introduction

Users surf the World Wide Web (WWW) by navigating along the hyperlinks that connect islands of content. If we could predict where surfers were going (that is, what they were seeking) we might be able to improve surfers' interactions with the WWW. Indeed, several research and industrial thrusts attempt to generate and utilize such predictions. These technologies include those for searching through WWW content, recommending related WWW pages, and reducing the time that users have to wait for WWW content downloads, as well as systems for analyzing the designs of web sites. Our previous work [12, 17] has attempted to characterize basic empirical properties of user surfing paths at web sites. Studied properties include the distribution of the number of clicks made by surfers at a web site, the complexity of path structures, and the consistency (or change) of paths over time. In this paper we explore pattern extraction and pattern matching techniques that predict future surfing paths. We expect that future work may exploit these modeling techniques in applications such as WWW search, recommendations, latency reduction, and analysis of web site designs.

Modeling and predicting user surfing paths involves tradeoffs between model complexity and predictive accuracy. In this paper we explore predictive modeling techniques that attempt to reduce model complexity while retaining predictive accuracy. The techniques merge two methods: a web-mining method that extracts significant surfing patterns by the identification of *longest repeating subsequences* (LRS) and a pattern-matching method that embodies the principle of *weighted specificity*. The LRS technique serves to reduce the complexity of the model by focusing on significant surfing patterns. This technique has been explored in connection with other areas of user interaction [8]. The weighted specificity principal exploits the finding that longer patterns of past surfing paths are more predictive. These techniques are motivated by results from previous predictive models and our prior empirical characterization [17] of surfing data. We shall show that when compared against a base standard of two different Markov model representations, the LRS pattern extraction and weighted specificity pattern matching techniques are able to dramatically reduce model complexity while retaining a high degree of predictive accuracy.

1.1. Surfing Paths

Figure 1 models the diffusion of surfers through a web site [12, 17]: (a) users begin surfing a web site starting from different entry pages, (b) as they surf the web site, users arrive at specific web pages having traveled different surfing paths, (c) users choose to traverse possible paths leading from pages they are currently visiting and (d) after surfing through some number of pages, users stop or go to another web site. Research in a number of application areas assumes, either explicitly or implicitly, that information about surfing paths observed in the past can provide useful information about surfing paths that will occur in the future.

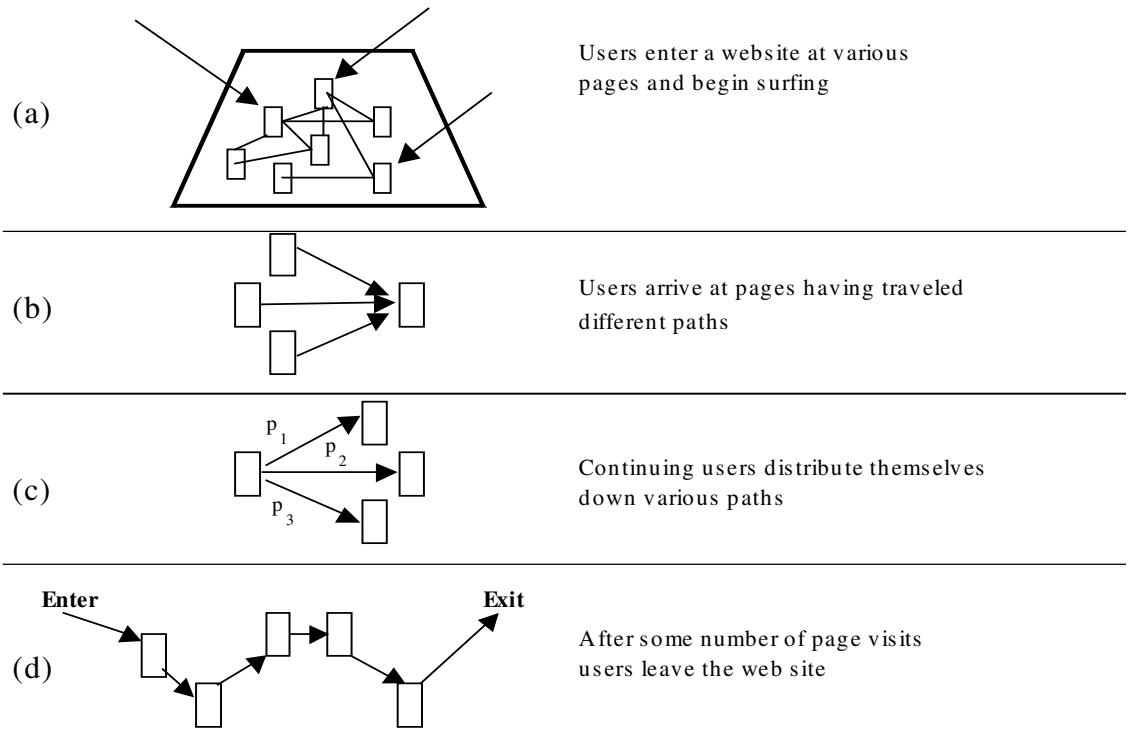

Figure 1. A conceptual model depicting the various stages of users traversing a web site.

2. Applications of Predictive Models

2.1 Search

The ability to accurately predict user surfing patterns could lead to a number of improvements in user-WWW interaction. For instance, the Google search engine [3] assumes that a model of surfing can lead to improvements in the precision of text-based search engine results. Conceptually, Google models surfers pursuing random walks over the entire WWW link structure. The distribution of visits over all WWW pages is obtained from this model. This distribution is then used to re-weight and re-rank the results of a text-based search engine. Under this model, surfer path information is viewed as an indicator of user interests, over and above the text keywords entered into a standard search engine. Following this line of reasoning one might also assume that surfing models with higher predictive accuracy would yield better search engines since the models provide a more realistic view of real world usage. The approach we propose here aims to be more informed than the random walk model implicit in Google. To the extent that surfing predictions improve text-based search results, we would expect that a more informed approach would yield better improvements than a random walk model.

2.2 Recommendation of Related Pages

Recently, tools have become available for suggesting related pages to surfers [10, 16, 18]. The "What's Related" tool button on the Netscape browser developed by Alexa, provides recommendations based on content, link structure, and usage patterns. Similar tools for specific repositories of WWW content are also provided by Autonomy. One can think of these tools as making the prediction that "surfers who come to this page (site) are most likely to be interested in the following pages (sites)." The predictive model of surfing proposed here could be used to enhance the recommendations made by these and other systems.

2.3 Web Site Models

Producers of WWW content are often interested in improvements in web site design. Recent research has developed visualizations to show the flow of users through a web site [5]. Businesses have emerged (e.g., Web Techniques, www.webtechniques.com) that send simulated users through existing web sites to provide data on web site design. Predictive models of surfer paths could help move the state of web site analysis from post-hoc modeling of past user interactions, or a current web site, to predictive models that can accurately simulate surfer paths through hypothetical

web site signs. Web site designers could explore different arrangements of links that promote desired flows of surfers through content.

2.4 Latency Reduction

Predictive models have significant potential to reduce user-perceived WWW latencies. Year after year, users report WWW delays as their number one problem in using the WWW [19]. One emerging strand of research that aims to improve WWW access times has grown out of research on improving file access time though prefetching and caching methods (e.g., [4]). Of particular interest to our research, Griffioen and Appleton's work [11] on file system prediction introduced the notion of automatic prefetching and evaluated the effectiveness of a one-hop Markov model.

A number of recent methods based on the use of Markov models as well as other methods have recently been proposed for prefetching and caching of WWW pages [1, 9, 13, 15, 20]. Roughly, the idea is that if a system could predict the content a surfer was going to visit next, and there was little cost involved, then the system could prefetch that content. While the user processes one page of content, other pages could be prefetched from high-latency remote sites into low-latency local storage.

Kroeger, Long, and Mogul [13] explored potential improvements in WWW interaction latencies that might be gained by predicting surfer paths. Current proxy servers typically mediate WWW access by accepting requests from user client applications. These requests are serviced by delivering content that has been cached, prefetched, and retrieved from other caches, or retrieved directly from origin Web servers. Proxies are typically accessed faster by clients than WWW servers are accessed by proxies. This usually occurs because the proxies are located on the same local area network as the client, whereas the proxies must access WWW servers over external network connections. Assuming this standard configuration, Kroeger et al. divided user-WWW latencies into (a) internal latencies caused by computers and networks utilized by the clients and proxies and (b) external latencies caused by computers and networks between the proxies and external WWW servers. Examining WWW user traces collected at the Digital Equipment Corporation WWW proxy, Kroeger et al. found that external latencies accounted for 88% of the total amount of latency seen by users geographically close to proxies. Other analyses by Kroeger et al. suggest that up to 60% of the observed external latencies could be reduced by improved caching and prefetching methods.

3. Predictive Surfing Models

Several predictive models of surfing have been developed in order to improve WWW latencies. It is instructive to review how their effectiveness varies. This review, plus a review of prior empirical characterizations of surfing patterns, motivate the pattern extraction and pattern matching techniques that we present in Section 5.1.

3.1 Path Profiles

Schechter, Krishnan, and Smith [20] utilized path and point profiles generated from the analysis of Web server logs to predict HTTP requests. They used these predictions to explore latency reductions through the pre-computation of dynamic Web pages. The profiles are constructed from user session. During a single session, a user interacting with the WWW traverses some sequence, *S*, of URLs. From that single sequence, the set of all possible subsequences is extracted as paths. Over some time period (say a day), the frequency of all observed paths is recorded. The resulting path profile consists of the set of all ordered pairs of paths and their observed frequencies.

Schechter et al. propose a method for predicting the next move of a surfer based on matching the surfer's current surfing sequence against the paths in the path profile. The ranking of matches is determined by a kind of specificity heuristic: the maximal prefixes of each path (the first *N*-1 elements of an *N*-length path) are compared element-wise against the same length suffixes of the user path (i.e., a size *N*-1 prefix is matched against the last *N*-1 elements of the user path), and the paths in the profile with the highest number of element-wise matches are returned. Partial matches are disallowed. In other words, if a surfer's path were <A, B, C>, indicating the user visited URL A, then URL B, then URL C, the path would be better matched by a path in the profile of <A, B, C, D> than <B, C, E>. For the paths in the profile that match, the one with the highest observed frequency is selected and used to make the prediction. Using our example, if <A, B, C, D> were the best match and most frequently observed path in the profile, then it would be used to predict that the user who just visited <A, B, C> would next visit URL D. Schechter et al. found that storing longer paths in the profile offered some improvements in prediction but they did not study this systematically.

Schechter et al. were also concerned with reducing model complexity. They pointed out that since a session *S* consists of an ordered sequence of URLs visited by a user, the worst case scenario of a naïve algorithm to store the decomposition of every path is order $|S|^3$. To reduce the model size, Schechter et al. used a maximal prefix trie with a minimal threshold requirement for repeating prefixes.

3.2 First-Order Markov Models

Based on the first-order Markov prediction method described in [11] for file prediction, Padmanabhan and Mogul [15] constructed a dependency graph containing nodes for all files ever accessed at a particular WWW server. Dependency arcs between nodes indicated that one file was accessed within some number of accesses *w* of another file. The arcs were weighted to reflect access rates. Padmanabhan and Mogul found that a predictive prefetching method based on this dependency representation reduced WWW latencies, and the reductions increased as *w* increased from *w*=2 to *w*=4.

Bestravos [1] used a method where one first estimates the conditional probabilities of transitioning directly from each page to every other page within a time T_w based upon server log file analysis. Like Padmanabhan and Mogul, this is a first-order Markov model approximation for predicting surfer paths, except the forward window is measured by time instead of number of pages. Using this approach, Bestravos developed a *speculative* service method that substantially reduced server loads and latencies. The architecture allows for the server and/or client to initiate the retrieval of resources predicted to be requested in the near future. Such systems now are generally referred to as *hint-based* systems, e.g., [7, 13, 15]. Bestravos did not, however, explore the effects of using longer surfer paths (higher-order Markov models) in the predictive model.

3.3 Summary

Regardless of where the transitions were recorded (proxy, server, etc.) all of these prefetching methods essentially record surfing transitions and use these data to predict future transitions. It is interesting to note that the methods of Schechter et al. and Padmanabhan and Mogul seemed to improve predictions when they stored longer path dependencies. In order to further motivate the rational behind the specificity principle for path matching, we next summarize the results of empirical analyses we [17] performed on server log files. In Section 5 we introduce the longest repeating subsequence (LRS) method as a technique that adheres to the principles of path specificity and model complexity reduction and evaluate it against surfing path data in Section 6.

4. Empirical Analysis Surfing Paths using K^{th}-Order Markov Models

In [17], we systematically evaluated the predictive capabilities of K^{th}-order Markov using ten days of log files collected at the xerox.com web site. Among other things, the results of this analysis suggest that storing longer path dependencies would lead to better prediction accuracy. This section reviews the methods and results of that study.

4.1 N-Gram Representation of Paths

Surfing paths can be represented as *n-grams*. N-grams can be formalized as tuples of the form $<X_1, X_2, \ldots X_n>$ to indicate sequences of page clicks by a population of users visiting a web site. Each of the components of the n-gram take on specific values $X_i = x_i$ for a specific surfing path taken by a specific user on a specific visit to a web site.

Users often surf over more than one page at a web site. One may record surfing n-grams, $<X_1, X_2, \ldots X_n>$ of any length observable in practice. Assume we define these n-grams as corresponding to individual surfing sessions by individual users. That is, each surfing session is comprised of a sequence of visits made by a surfer, with no significantly long pauses. Over the course of a data collection period—say a day—one finds that the lengths, *n*, of surfing paths will be distributed as an inverse Gaussian function[1], with the mode of the distribution being length one. This appears to be a universal law that is predicted from general assumptions about the foraging decisions made by individual surfers [12]. In practice one typically finds that the majority of users visit one page on a web site and then click to another web site.

4.2 K^{th}-Order Markov Approximations

The first-order Markov model used by Bestravos [1] and Padmanabhan and Mogul [15] were concerned with page-to-page transition probabilities. These can be

[1] The inverse Gaussian distribution is a heavily skewed distribution (much like the log normal distribution) that predicts that the bulk of recorded paths will be very short with a few very long paths.

estimated from n-grams of the form $<X_1, X_2>$ to yield the conditional probabilities

$$p(x_2 \mid x_1) = \Pr(X_2 = x_2 \mid X_1 = x_1).$$

If we want to capture longer surfing paths, we may wish to consider the conditional probability that a surfer transitions to an n^{th} page given their previous $k = n-1$ page visits:

$$p(x_n \mid x_{n-1},...x_{n-k}) = \Pr(X_n = x_n \mid X_{n-1},...,X_{n-k}).$$

Such conditional probabilities are known as K^{th}-order Markov approximations (or K^{th}-order Markov models). The zeroth order Markov model is the unconditional base rate probability:

$$p(x_n) = \Pr(X_n)$$

which is the probability of a page visit. This might be estimated as the proportion of visits to a page over a period of time.

4.3 Summary of Prior Empirical Analysis

Using data collected from xerox.com for the dates May 10, 1998 through May 19, 1998 we [17] systematically tested the predictive properties of K^{th}-order Markov models. The site received between 220,026 and 651,640 requests per day during this period. Over this period, there were 16,051 files on the xerox.com web site, of which 8,517 pages were HTML.

The reliable identification of user paths in a web site is often a complicated and site specific task. The xerox.com web site issued cookies to users only upon entry to the Xerox splash page and recorded the "User-Agent" and "Referer" field for each request when present. User paths were identified using cookies and a set of fallback heuristics when cookies did not exist[2]. The Xerox server permitted caching of resources. When present, Get-If-Modified headers were included in the construction of user paths. Still, under certain client and proxy configurations, the xerox.com caching policy resulted in missed client navigation, e.g., when a user clicked on the "Back" button. As a result, user paths constructed by our heuristics often contained transitions to other pages not linked to from the current page. Over the ten days used in this study, 176,712 user paths were observed.

[2] The exact methods, tradeoffs, and the effects of counting each IP as a user are described greater detail in [17].

The models were estimated from surfing transitions extracted from training sets of WWW server log file data and tested against test sets of data that occurred after the training set. The prediction scenario assumed a surfer was just observed making k page visits. In order to make a prediction of the next page visit the model must have (a) an estimate of $p(x_n|x_{n-1},\ldots x_{n-k})$ from the training data, which required that (b) a path of k visits $<x_{n-1},\ldots x_{n-k}>$ (a penultimate path) had been observed in the training data. Given a penultimate path match between paths in the training and test data, the model examined all the conditional probabilities $p(x_n|x_{n-1},\ldots x_{n-k})$ available for all pages x_n, and predicted that the page having the highest conditional probability of occurring next would in fact be requested next. Whether the observed surfer made the predicted visit (a hit) or not (a miss) was then tallied. The model did not make a prediction when a matching path in the model did not exist. It is important to examine the performance of the model in making correct predictions as well as incorrect predictions since incorrect predictions often result in undesirable costs, which can mitigate any benefits.

Table 1 presents a subset of the analyses presented in [17], where predictions based on a training set collected one day and were tested against data collected the next day. We define

- Pr(Match), the probability that a penultimate path, $<x_{n-1},\ldots x_{n-k}>$, observed in the test data was matched by the same penultimate path in the training data,
- Pr(Hit|Match), the conditional probability that page x_n is visited, given that $<x_{n-1},\ldots x_{n-k}>$, is the penultimate path and the highest probability conditional on that path is $p(x_n|x_{n-1},\ldots x_{n-k})$,
- Pr(Hit) = Pr(Hit|Match)•Pr(Match), the probability that the page visited in the test set is the one estimated from the training as the most likely to occur (in accordance with the method in our scenario),
- Pr(Miss|Match), the conditional probability that page x_n is *not* visited, given that $<x_{n-1},\ldots x_{n-k}>$, is the penultimate path and the highest probability conditional on that path is $p(x_n|x_{n-1},\ldots x_{n-k})$,
- Pr(Miss) = Pr(Miss|Match)•Pr(Match), the probability that the page visited in the test set is *not* the one estimated from the training as the most likely to occur (in accordance with the method in our scenario), and

Table 1. Probability of matching a path of the same length, Pr(Match), conditional probability of accurately predicting the next page visit of a surfer given a path match, Pr(Hit|Match), and the overall accuracy of predicting a surfer page visit, Pr(Hit). Conditional miss probabilities, Pr(Miss|Match), overall miss rate, Pr(Miss), and hit to miss ratios, Pr (Hit)/Pr(Miss), are also provided. Training data and test data were collected on successive days. Matching and predictions were conducted on paths of the same length. No predictions were made in the absence of a match.

Order of Markov Model	Pr (Match)	Pr (Hit|Match)	Pr (Hit)	Pr (Miss|Match)	Pr (Miss)	Pr(Hit)/ Pr(Miss)
1	.87	.23	.20	.77	.67	.30
2	.61	.30	.18	.70	.43	.42
3	.30	.34	.10	.66	.20	.51
4	.21	.41	.08	.59	.12	.65
5	.21	.29	.06	.71	.15	.40
6	.20	.31	.06	.69	.14	.43
7	.20	.27	.05	.73	.15	.34

- Pr(Hit)/Pr(Miss), the probability of correctly predicting page *x* divided by the probability of making an incorrect prediction for all transitions.

The last metric provides a coarse measure of the benefit-cost ratio. That is,

$$\text{Benefit:Cost} = B * \Pr(\text{Hit}) / C * \Pr(\text{miss})$$

where B and C vary between 0 and 1 and represent the relative weights associated with the benefits and costs for each application. Naturally, different applications have different inherent tradeoffs associated with the benefits of making a correct prediction versus the costs of making an incorrect prediction and will often require a more complex metric to encode the true tradeoffs. From Table 1 it can be seen that,

- Lower-order models have higher Pr(Match). This indicates that the chances of seeing short surfing paths across days are much higher than for longer surfing paths.
- Higher-order models have higher Pr(Hit|Match). If one can find a match of longer surfing paths then they are likely to be better predictors than shorter surfing paths.
- Lower-order models have higher Pr(Hit) overall. This indicates that the overall hit rate is dominated by the probability of finding a penultimate path match, Pr(Match).
- For the xerox.com data, if one assumes that the benefit of making a correct hit equals the cost of making an incorrect prediction ($B = C = 1$), using a 4^{th} order model is optimal.

Included in [17] is an analysis of prediction stability over time using entropy analysis and the improvements due to increasing the size of the training data set.

The results of the above analyses lead us to explore methods that would improve both pattern extraction and pattern matching. In this next section, we introduce the notion of longest repeating subsequences (LRS) to identify information rich patterns and present two modifications of LRS to improve pattern matching. These models are then compared to different Markov models in Section 6.

5. Model and Prediction Methods

Schechter et al. [20] discuss the explosive growth of storage requirements for path profiles (we discuss their analysis in detail below). Producing an accurate predictive model using the least amount of space has many computational benefits as well as practical benefits. One might even imagine a model being compact enough to reside in memory for each thread handling requests on a busy WWW server.

One solution to reduce model space is to use compact data structures like tries as was used in Schechter et al. While an issue of great interest and complexity, we will not treat more efficient data structure solutions in this paper. Another solution is to attack the problem at the core and remove low information elements from the model. The LRS technique treats the problem as a data-mining task, where some of the paths are considered noise. This hinges off the insight that many paths occur infrequently, often as a result of erroneous navigation. Identifying repeating subsequences enables common sub-paths to be extracted. This has the benefit of preserving the sequential nature of the paths while being robust to noise. Using LRS, the storage requirement is reduced by saving only information rich paths.

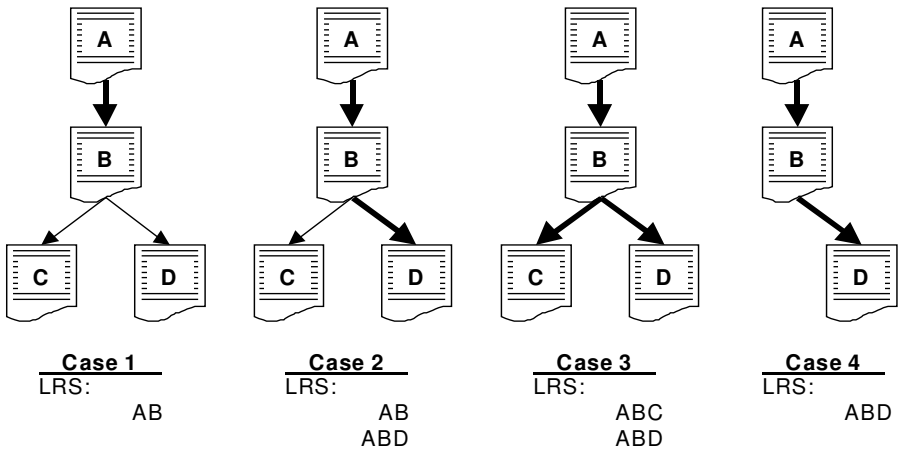

Figure 2. Examples illustrating the formation of longest repeating subsequences (LRS). Thick-lined arrows indicate more than one traversal whereas thin-lined arrows indicate only one traversal. For each case, the resulting LRS are listed.

The second method we explore is to use a specificity heuristic in pattern matching. As shown by the results of Schechter et al., Padmanabhan and Mogul, and Table 1, higher-order Markov models result in higher prediction rates when there is a penultimate path match. The principle of specificity encourages the use of higher-order path matches whenever possible to maximize hit rates. The drawback of this approach is that the likelihood of a higher-order path match is quite small, resulting in lower overall hit rates. The decreased likelihood of a long path match is in part a function of the inverse Gaussian distribution of path lengths, where over 80% of the distribution is accounted for by paths of length four or less. It also is a function of the exponential growth in the combinations of possible paths, where the length of the path is the base and the exponent is the average number of links per page within the web site. As an example, given a site with an average of three links per page, a path of length eight will have 6,561 possible combinations assuming only forward traversals. Due to limitations in the ability of server logs to completely capture user navigation, recorded paths are not limited to only forward links, which makes the set of possible combinations even larger.

5.1 Longest Repeating Sequences

A longest repeating subsequence [8] is a sequence of items where

1) subsequence means a set of consecutive items,
2) repeated means the item occurs more than some threshold T, where T typically equals one, and
3) longest means that although a subsequence may be part of another repeated subsequence, there is at least once occurrence of this subsequence where this is the longest repeating.

To help illustrate, suppose we have the case where a web site contains the pages A, B, C, and D, where A contains a hyperlink to B and B contains hyperlinks to both C and D. As shown in Figure 2, if users repeatedly navigate from A to B, but only one user clicks through to C and only one user clicks through to D (as in Case 1), the longest repeating subsequence is AB. If however more than one user clicks through from B to D (as in Case 2), then both AB and ABD are longest repeating subsequences. In this event, AB is a LRS since on at least one other occasion, AB was not followed by ABD. In Case 3, both ABC and ABD are LRS since both occur more than once and are the longest subsequences. Note that AB is not a LRS since it is never the longest repeating subsequence as in Case 4 for the LRS ABD.

LRS have several interesting properties. First, the complexity of the resulting n-grams is reduced as the low probability transitions are automatically excluded from further analysis. This reduction happens for all transitions that occur only T times, which in some cases, will result in a prediction not being made. For example, with a threshold of T=1 the penultimate match for the LRS AB is A in Case 1. In this case, a prediction will not be made after the pages A and B have been requested. The reduction in complexity typically results in a slight loss of pattern matching, as will be made evident in the below experiments.

To contrast, if all 2^{nd}-order Markov approximations were being used to make predictions and path AB has been observed, the system would have to randomly

select either C or D since each are equally probable. If a hint-based prefetching system were used, a set of predictions could also be made by the server and passed along to the client with the contents of the currently requested page. In this case, a list could be returned that contains both C and D, each with equal probability. We will study the effect of varying result list size in section 6.3. Note also that in Case 2, after seeing AB the LRS predictive model will predict D, just as would all the 2nd-order Markov approximations. In this manner, the LRS model can be viewed as a proper subset of the Kth-order Markov approximations.

Another interesting property of the LRS model is its bias towards specificity. Any single page-to-page transition that is always repeated as part of longer sequences is not included into the LRS model. For web sites, a transition from the popular entry points usually results in all forward links being followed more than some threshold *T*. For predictive purposes, this reduces the number of penultimate matches that the LRS model can make for shorter paths and as direct result, lowers the overall hit rate for the model.

5.2 Hybrid LRS-Markov Models

We now propose two straightforward hybrid models that use LRS subsequences extracted from training data. In the first hybrid LRS model we extract LRS patterns from the training data and use these to estimate a first-order Markov model. That is, we decompose each LRS pattern into a series of corresponding one-hop n-grams, e.g., the LRS ABCD would result in AB, BC, and CD 1-grams. We call this model a *one-hop LRS* model. In Section 6.1, we compare the one-hop LRS model against a first-order Markov model estimated from all the paths in a training data set.

The second hybrid LRS model decomposes the extracted LRS subsequences into all possible n-grams. The resulting model is a reduced set of n-grams of various lengths. We call this the *All-Kth-Order LRS* model, since all orders of *k* are able to make predictions. The main advantage of this model is that it incorporates the specificity principle of pattern matching by utilizing the increased predictive power contained in longer paths. Below, we test the All-Kth-Order LRS model against an *All-Kth-Order* Markov model. As with the All-Kth-Order LRS model the All-Kth-Order Markov model is constructed by decomposing all possible subsequences into varying length n-grams.

5.3 Model complexity reduction

Schechter et al. [20] note that their profiling method requires that a path of length *S* be decomposed into all subpaths. They point out that to store the profile for a single path of length *S* (i.e., all the necessary subpaths) requires storage that grows as $O(S^3)$. Note, however, that *n* distinct paths of length *S* will not necessarily each require $O(S^3)$ storage. The distinct paths of length *S* will share many redundant subpaths of length less than *S*. The problem with the Schechter et al. analysis is that it fails to incorporate the combinatorics of surfing paths. Moreover, it is only in the worst case that one needs to store all distinct paths and all distinct subpaths. One may have the goal of storing those that are likely to be needed, and this is the aim of the LRS pattern extraction method.

The amount of space required for all models—LRS and Markov—depends on the combinatorics (e.g., the redundancy) of user paths. Basically, we need to know how the *number of patterns* of paths and subpaths grows as a process of surfing. A model of the path combinatorics could be formulated if we understood the underlying path-generating process [6]. Unfortunately, the process that generates user paths within web sites has not been characterized in detail (although see [12] for beginnings) and may well very from site to site as well as from time to time.

In the absence of a more informed surfing process model, we explore the following simple model. Assume that surfing paths have an average branching factor *b*. That is, surfers may start in *b* places, and from each page they move to one of *b* pages on average. Assume that the surfers move at random, thereby generating random paths. Surfing paths of length *S* can be divided into *S*/*k* subpath partitions of length $0 < k \leq S$. Each subpath partition of length *k* will have b^k patterns (assuming randomly chosen paths along the *b* branches). So the complexity cost, $C(k)$, in terms of number of patterns as a function of subpath length *k* will be

$$C(S) = \sum_{i=1}^{S} (S/i) b^i$$

As noted by Newell and Rosenbloom [14], this does not have a closed-form solution but one can show that the derivative is such that

$$C'(S) = e^{\log(b) S} .$$

It should also be noted, that under this random process model longer path patterns are less likely to recur than shorter path patterns.

For the one-hop models, the worst case complexity of a one-hop Markov model is $O(V^2)$, where V is the number of pages in the web site and every page is connected to every other page. In practice, the connectivity of web pages is sparser as evidenced by the low number of hyperlinks per page [2]. Such sparse connectivity graphs are well suited for adjacency-list representations, which require $O(V+E)$ where E is the number of edges between pages V. As mentioned previously, the worst-case storage requirements for All-K^{th}-order Markov approximations are $O(S^3)$.

The result of applying LRS to a set of path data is a possible pruning of the resulting Markov models. The extent of the pruning depends on the amount of redundancy and the value selected for the repeating threshold T. In the worst case, the amount of storage required for the LRS models is equal to the worst-case for the equivalent Markov models. As we shall see, in practice, the reduction can be quite significant, though we note that the amount of reduction will likely vary from web site to web site and even within sites depending on traffic patterns.

6. Evaluation

In order to test whether the hybrid LRS models help achieve the goal of reducing complexity while maintaining predictive power, we conducted a set of experiments on the same data used in our previous study [17] summarized above. For the below analyses, three consecutive weekdays were chosen starting on Monday May 11, 1998 for the training data and Thursday May 14, 1998 for the test day. There were 63,329 user paths for the training days and 19,069 paths for the test day. As with our previous evaluation, predictions were not made when the model did not contain a matching pattern.

6.1 One-hop Markov and LRS Comparison

One-hop Markov models and their derivatives are important to study empirically as they are conceptually simple and easy to implement. Developing a rich understanding of these models helps frame the results and tradeoffs of more complex models. For this experiment, the one-hop Markov and the one-hop LRS models were built using three training days. For each path in the test day, the path was decomposed into a set of sequential transitions. Each model was then tested to see if there was a matching prefix (Match) for each path, and if so, if the correct transition was predicted (Hit). From this, the probability of a match Pr(Match), the probability of a hit given a match Pr(Hit|Match), the hit rate across all transitions Pr(Hit), and the benefit-cost ratio Pr(Hit)/Pr(Miss) were computed along with the corresponding miss probabilities.

Table 2 displays the results for the one-hop Markov model and the one-hop LRS model. Overall, there were 13,189 unique hops in the training data with the one-hop LRS model reducing this set by nearly a third to 4,953 unique hops. The hit probabilities in Table 2 are slightly higher than in Table 1. These discrepancies result from weekday and weekend traffic differing significantly at xerox.com and that the data used in Table 2 contained only weekday data for the training and testing sets.

Table 2. Results of comparing a one-hop Markov model with the one-hop LRS model.

	One-hop Markov	One-hop LRS
Number of one-hops	13,189	4,953
Model Size (bytes)	372,218	136,177
Total transitions in test data	25,485	25,485
Matches	25,263	24,363
Hits	6,402	6,056
Pr(Match)	.99	.96
Pr(Hit|Match)	.25	.25
Pr(Hit)	.25	.24
Pr(Miss|Match)	.75	.75
Pr(Miss)	.74	.72
Pr(Hit)/Pr(Miss)	.34	.33

The one-hop LRS model produces a reduction in the total size required to store the model thus satisfying the complexity reduction principle. One might expect that the sharp reduction in the model's complexity would result in an equally sharp reduction in predictive ability. However, this is not the case as the one-hop LRS model performs nearly as well as the one-hop Markov model in terms of the total number of predictions made (25,263 one-hop Markov versus 24,363 one-hop LRS), total number of hits (6,402 one-hop Markov versus 6,056 one-hop LRS), and the overall probability of correctly predicting the next page Pr(Hit) (25% one-hop Markov versus 24% one-hop LRS). Both models produced similar miss probabilities, with the Markov model resulting in a 75% incorrect prediction rate and the LRS model 72%. The benefit-to-cost ratio for each model is nearly identical.

The one-hop LRS model was able to significantly reduce model complexity while preserving predictive ability. However both these models are very simple and they do not leverage the greater predictive abilities of longer paths.

6.2 All-Kth-order Markov approximation and All-Kth-order LRS Comparison

In order to adhere to the principle of specificity, longer paths should be used wherever possible given their greater predictive power. With this in mind, we set out to explore the differences between the All-Kth-order Markov model and the All-Kth -order LRS model. As with the one-hop analysis above, since the All-Kth-LRS is a subset of all All-Kth-order Markov approximations, we did not expect to see better predictive capabilities, but rather wanted to examine the tradeoffs between complexity reduction and the model's predictive power.

Table 3. Results from testing the All-Kth-order Markov and the All-Kth-order LRS models.

	All-Kth-order Markov	All-Kth-order LRS
Number of one-hops	217,064	18,549
Model Size (bytes)	8,847,169	616,790
Total transitions in test data	25,485	25,485
Matches	25,263	24,363
Hits	7,704	6,991
Pr(Match)	.99	.99
Pr(Hit|Match)	.31	.27
Pr(Hit)	.30	.27
Pr(Miss|Match)	.69	.73
Pr(Miss)	.69	.72
Pr(Hit)/Pr(Miss)	.44	.38

For this experiment, the same three training days and test day were used. For each training day, each path was decomposed into all corresponding n-grams for the All-Kth-order Markov model. With the All-Kth-order LRS model, the LRS were first computed and then each LRS was decomposed into the set of all n-gram subsequences. For the evaluation, each transition in a path of the test data was broken down into its corresponding subsequence suffixes. The resulting suffixes were then checked to see if there was a matching n-gram for each model. The prediction with the greatest specificity (number of element-wise matches between training and test paths) weighted by conditional probability was then selected as the prediction. This weighted specificity measure differs slightly from that used in our previous study and that used by Schechter et al. A prediction was not made if a suffix match was not found.

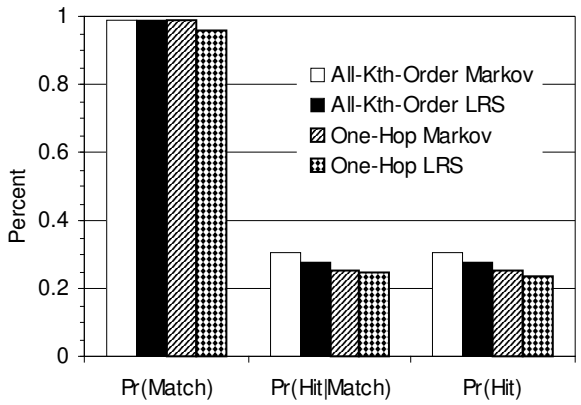

Figure 3. Summary of the likelihood of being able to make a prediction Pr(Match), correct prediction given a match pr(Hit|Match) and overall hit rate Pr(Hit) of each model.

Table 3 shows the results of the experiment. As with the previous experiments, the test data consisted of 25,485 transitions. As one would expect, the complexity of storing the All-Kth-order Markov is significant as there are close to one quarter million n-gram sequences which in a naive representation require 8,800 Kbytes. The All-Kth-order LRS model reduces the model space by an order of magnitude, decreasing the total number of n-grams to 18,549, which consume 616 Kbytes of space, which is nearly double that for the one-hop Markov model and almost six times that for the one-hop LRS model. The All-Kth-order Markov model was able to match nearly all transitions and resulted in a correct prediction 31% of the time. The performance of this model is better than the one-hop Markov model tested earlier, highlighting the specificity principle. The All-Kth-order LRS model performed nearly as well, correctly predicting the next page request 27% of the time, with an overall miss rate of 73% compared to 69% for the Markov model. The benefit-to-cost ratio for the Markov model exceeds that for the LRS model (.44 versus .38).

Figure 3 summarizes the results of the two experiments with respect to hit rates. In certain cases, the difference between the predictive power of the All-Kth may not outweigh the considerable space savings of the one-hop models. While the All-Kth-order Markov model provides the highest hit rate, the one-hop Markov model provides 83% of the predictive power while consuming only 4.2% of the space. The same is true for the one-hop LRS, where 80% of the predictive power is accomplished using only 1.5% of the space.

6.3 Parameterization of Prediction Set

Restricting the prediction to only one guess imposes a rather stringent constraint. One could also predict a larger set of pages that could be surfed to on the next click by a user. For certain applications that aim to reduce user latency like hint-based perfecting architectures, it is important to understand the cost-benefit tradeoffs in a systematic manner. This section explores how the hit rate varies when considering larger sets of predictions.

We evaluated each model's performance when returning sets of between one and ten predictions. Each set was constructed by ranking the predictions about the next page visit in decreasing order of likelihood, and selecting the topmost *n* predictions. For the one-hop models, the predictions were ranked by probability of predicted transition. For the All-K^{th}–order models, the predictions were ranked by the weighted specificity principle.

Figure 4 shows the probability of each model correctly predicting the next page visit across different prediction set sizes. As with the previous experiments, the All-K^{th}-order models performed better than the one-hop models due to the increased information contained in the longer paths and the LRS models performed slightly worse than the Markov models at a fraction of the model space. Increasing the prediction set has a dramatic impact on predictive power, with the predictive power of each method nearly doubling by increasing the set size to four elements.

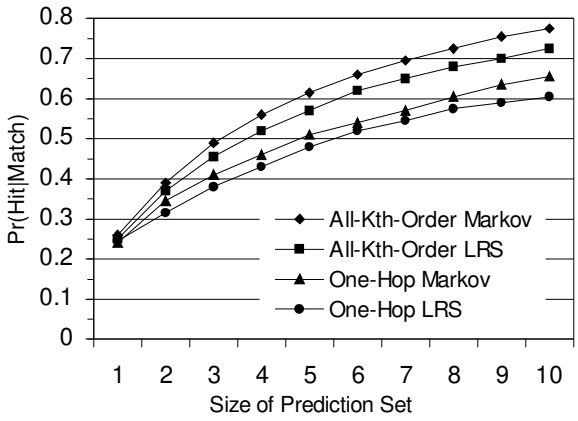

Figure 4. The effect on making a correct prediction as a function of the number of predictions made by each model.

7. Future Work

Motivated by the principle of weighted path specificity and complexity reduction, we have shown that small compact models of user surfing behavior exist that retain predictive power. Clearly, other methods exist to predict future access of resources. While this paper has focused on enhancements to various Markov models, we believe that the concept of LRS can be successfully applied to Markov models in other domains as well as to other suffix-based methods.

In the above experiments, subsequences that occurred more than once were considered repeating. In theory, repeating can be defined to be any occurrence threshold *T*. We hypothesize that model complexity can be reduced even further while maintaining acceptable predictive power by raising the threshold *T*. Determining the ideal threshold will depend upon the specific data and the intended application.

Another variable that was not parameterized in the above experiments was the confidence level for each prediction. Inspection of the data revealed that in several cases, each model was making predictions that were not very likely. A modified pattern-matching algorithm could be restricted to only make predictions when a given probability of making a successful prediction was achieved. While lowering the probability of a match Pr(Match), the reduced overall hit Pr(Hit) rate could be offset by the increased likelihood of being correct Pr(Hit|Match). This is especially appropriate for applications where the cost of being wrong outweighs the benefits of being right.

The application of the LRS models to prefetching and latency reduction is also of interest. Given the compact size of the LRS models, one can imagine HTTP server threads issuing hint lists to clients while maintaining the model in memory. Our preference is a system in which the server provides the client with a list of suggested prefetchable items. The client then decides what items to prefetch when. This would not require, but might benefit greatly from modifying the current pre-computed static LRS model into an adaptive, real-time model, especially since the optimal hint set size will most likely vary from server to server as well as page to page within a server. The overall effectiveness of this application and modifications needs to be evaluated. In a similar manner, LRS provides a compact information-dense method to store user paths for later data analysis.

From a more psychological perspective, we note that LRS may represent the common navigational sub-units or "chunks" across all users and documents on a web site. That is, the repeating subsequences may be an appropriate logical unit to encode the paths most traveled. We postulate that these chunks are well suited

for document and user clustering since they preserve the sequential nature of surfing, are robust against noise, and reduce overall computational complexity.

Finally, the exact space reduction achievable by LRS for Web surfing requires the generating function underlying web surfing to be identified and other traces to be examined.

8. Conclusion

Clearly there exists a tradeoff between model complexity and predictive power. Our initial exploration into the predictive capabilities of user paths led us to postulate the principles of complexity reduction and specificity. From this, we employed the notion of longest repeating subsequences to produce a subset of all paths. We showed that in the simplest case of modeling paths as a one-hop Markov model, the reduced one-hop LRS model was able to match the performance accuracy of the one-hop Markov model while reducing the complexity by nearly a third. We then showed that overall hit rates could be raised by including the principle of specificity, with the All-K^{th}-Order LRS model almost equaling the performance of the All-K^{th}-order Markov model while reducing the complexity by over an order of magnitude. We further showed that varying the size of prediction set results large gains in predictive.

9. Acknowledgements

We would like to the USENIX reviewers and our shepherd Jeff Mogul for their helpful comments and suggestions.

10. References

1. Bestravos, A. (1995). Using speculation to reduce server load and service time on the WWW. *Proceedings of the 4th ACM International Conference on Information and Knowledge Management, (CIKM '95)* Baltimore, MD.

2. Bray, T. (1996). Measuring the Web. *Proceedings of the Fifth International WWW Conference* Paris, France.

3. Brin, S. and Page, L. (1998). The anatomy of a large-scale hypertextual web search engine. *Proceedings of the Seventh International WWW Conference* Brisbane, Australia.

4. Cao, P., Felten, E.W., Karlin, A.R., and Li, K. (1996). Implementation and performance of integrated application-controlled file caching, prefetching, and disk scheduling. *ACM Transactions on Computer Systems, 14*, 311-343.

5. Chi, E., Pitkow, J., Mackinlay, J., Pirolli, P., Gossweiler, R., and Card, S.K. (1998). Visualizing the evolution of web ecologies. *Proceedings of the Conference on Human Factors in Computing Systems, (CHI '98)* Los Angeles, CA.

6. Clement, J., Flajolet, P., and Valle, B. (1998). The analysis of hybrid trie structures. *Proceedings of the 9th Annual ACM-SIAM Symposium on Discrete Algorithms.*

7. Cohen, E., Krishnamurthy, B., and Rexford, J. (1998). Improved end-to-end performance of the web using server volumes and proxy filters. *Proceedings of the ACM SIGCOM.*

8. Crow, D. and Smith, B. (1992). DB_Habits: Comparing minimal knowledge and knowledge-based approaches to pattern recognition in the domain of user-computer interactions. In R. Beale and J. Finlay (Eds.), *Neural networks and pattern recognition in human-computer interaction* (pp. 39-63). New York: Ellis Horwood.

9. Cunha, C.R. (1997). *Trace analysis and its applications to performance enhancements of distributed information systems*. Unpublished thesis, Boston University, Boston.

10. Dean, J. and Henzinger, M.R. (1999). Finding related pages in the World Wide Web. *Proceedings of the Eighth International World Wide Web Conference* Toronto, Canada.

11. Griffioen, J. and Appleton, R. (1994). Reducing file system latency using a predictive approach. *Proceedings of the 1994 Summer USENIX Technical Conference* Cambridge, MA.

12. Huberman, B.A., Pirolli, P., Pitkow, J., and Lukose, R.J. (1998). Strong regularities in World Wide Web surfing. *Science, 280*, 95-97.

13. Kroeger, T.M., Long, D.D.E., and Mogul, J.C. (1997). Exploring the bounds of web latency reduction from caching and prefetching. *Proceedings of the USENIX Symposium on Internet Technologies and Systems, (USITS '97)* Monterey, CA.

14. Newell, A. and Rosenbloom, P.S. (1981). Mechanisms of skill acquisition and the law of practice. In J.R. Anderson (Ed.) *Cognitive skills and their acquisition* (pp. 1-55). Hillsdale, NJ: Lawrence Erlbaum.

15. Padmanabhan, V.N. and Mogul, J.C. (1996). Using predictive prefetching to improve World Wide Web latency. *Computer Communication Review, 26*, 22-36.

16. Pirolli, P., Pitkow, J., and Rao, R. (1996). Silk from a sow's ear: Extracting usable structures from the web. *Proceedings of the Conference on Human Factors in Computing Systems, (CHI '96)* Vancouver, Canada.

17. Pirolli, P. and Pitkow, J.E. (1999). Distributions of surfers' paths through the World Wide Web: Empirical characterization. *World Wide Web, 2(1-2)*, 29-45.

18. Pitkow, J. and Pirolli, P. (1997). Life, death, and lawfulness on the electronic frontier. *Proceedings of the Conference on Human Factors in Computing Systems, (CHI '97)* Atlanta, GA.

19. Pitkow, J.E. and Kehoe, C.M. (1999). GVU's Tenth WWW User Survey. *Online Publication: http://www.gvu.gatech.edu/user_surveys.*

20. Schechter, S., Krishnan, M., and Smith, M.D. (1998). Using path profiles to predict HTTP requests. *Proceedings of the Seventh International World Wide Web Conference* Brisbane, Australia.