Computer Networks 33 (2000) 377–386

COMPUTER NETWORKS

www.elsevier.com/locate/comnet

Link prediction and path analysis using Markov chains ☆

Ramesh R. Sarukkai [1]

Yahoo Inc., 3420 Central Expressway, Santa Clara, CA, USA

Abstract

The enormous growth in the number of documents in the World Wide Web increases the need for improved link navigation and path analysis models. Link prediction and path analysis are important problems with a wide range of applications ranging from personalization to Web server request prediction. The sheer size of the World Wide Web coupled with the variation in users' navigation patterns makes this a very difficult sequence modelling problem. In this paper, the notion of probabilistic link prediction and path analysis using Markov chains is proposed and evaluated. Markov chains allow the system to dynamically model the URL access patterns that are observed in navigation logs based on the previous state. Furthermore, the Markov chain model can also be used in a generative mode to automatically obtain tours. The Markov transition matrix can be analysed further using eigenvector decomposition to obtain 'personalized hubs/authorities'. The utility of the Markov chain approach is demonstrated in many domains: HTTP request prediction, system-driven adaptive Web navigation, tour generation, and detection of 'personalized hubs/authorities' from user navigation profiles. The generality and power of Markov chains is a first step towards the application of powerful probabilistic models to Web path analysis and link prediction.

Keywords: Link prediction; HTTP request; Adaptive navigation; Tour generation; Hubs/authorities; Markov chains

1. Introduction

1.1. Problem description

With the rapid growth of the World Wide Web (currently estimated to be about 800 million pages [10]), it is almost impractical for individual users to navigate effectively through many of the Web documents. The most obvious and prominent methods are search engines (e.g. Google) and directory services (e.g. Yahoo!) to access information from the World Wide Web. While search tools and directories are very useful in indexing Web documents relevant to a particular topic, they are seldom efficient for the user to 'navigate' through a set of related/connected pages.

There are alternate approaches that are currently adopted to address the navigation problem [12]. The first concept is agent-assisted navigation (e.g. [3, 14]). In such a system, the system suggests links that the user can follow during the process of browsing. The second approach is that of tour generation (e.g. [7]) wherein the system generates a tour which takes the user from one link to another. Another approach is analysis of the World Wide Web structure to identify important hubs and authorities [8] which are

☆ This work was done by the author prior to his employment at Yahoo Inc.

[1] E-mail: rsarukkai@yahoo.com

1389-1286/00/$ – see front matter
PII: S1389-1286(00)00044-X

important sites that the user might want to browse. The concept of Hubs/Authorities leads to the notion of a 'Web community' [6].

While the above techniques are the right direction towards solving the navigation problem, the key lies in 'personalization'. Personalization can be achieved in a variety of forms. A common example of personalization is matching of a user's profile with a set of users. The system then suggests to the user items that other users with similar interests have purchased (e.g. books on amazon.com). Another example of personalization is a configurable information filtering agent to deliver personalized news. Similarly, the notion of navigation can be personalized, and we believe that this approach will lead to a satisfactory solution to navigating the huge World Wide Web space.

While users' access to the information on the World Wide Web is an important problem, the ability of Web servers to provide this information in a rapid manner is also crucial. Link prediction may be used to prefetch documents while the user is perusing the current document. This allows the server to utilize free cycles to reduce the latencies of users' requests.

At the heart of all the above problems lies the analysis and modelling of Web link sequences. An efficient and accurate mechanism of modelling users' navigation link sequences will offer a general and extensible solution to all the above problems. Thus, this paper focuses on a probabilistic approach to the problem of Web link sequence modelling, analysis and prediction.

1.2. Our contributions

Given that the main problem is 'Web sequence modelling', the next step is the selection of an appropriate mathematical model. Probabilistic models have been applied successfully to numerous time-series prediction problems. In particular, Markov chains and hidden Markov models have been enormously successful in sequence matching/generation. In this paper, we demonstrate the utility of applying such techniques to World Wide Web link prediction and path analysis.

Markov chains have many attractive properties. They can be easily estimated statistically. Since the Markov chain model is also generative, navigation tours can be automatically derived. The Markov chain model can also be adapted on-the-fly with additional user navigation information. When used in conjunction with a Web server, the same model can be used to predict the probability of seeing a link in the future given a history of accessed links. The Markov state transition matrix can be viewed as a 'user traversal' representation of the Web space, and eigenvector decomposition techniques can be applied to generate hubs/authorities. In such a case, since the transition matrix is generated using client/user traversal data, the hubs/authorities can be viewed as 'personalized hubs/authorities'. Thus, the main contribution of our work is the notion of probabilistic link prediction and path analysis using Markov chains.

1.3. Organization of the paper

Section 2 presents the basics of Markov chains [4], and describes their utility in the context of link prediction. Section 3 describes the overall architecture of a system that utilizes the Markov chain prediction and analysis module. Section 4 summarizes four applications: HTTP server request prediction, link prediction, automatic tour generation and 'personalized' hub/authority detection. Experimental results are summarized in Section 5. Section 6 on 'related work' contrasts our work with other approaches to navigation, personalization, link prediction, and HTTP request analysis. This is followed by concluding remarks and references.

2. Markov chain models for link prediction

A discrete Markov chain model can be defined by the tuple $\langle S, \boldsymbol{A}, \lambda \rangle$. S corresponds to the state space, $\boldsymbol{A}$ is a matrix representing transition probabilities from one state to another, and λ is the initial probability distribution of the states in S. The fundamental property of the Markov model is the dependency on the previous state. If the vector $\boldsymbol{s}(t)$ denotes the probability vector for all the states at time t, then:

$$\hat{\boldsymbol{s}}(t) = \hat{\boldsymbol{s}}(t-1)\boldsymbol{A}. \tag{1}$$

If there are n states in our Markov chain, then the matrix of transition probabilities $\boldsymbol{A}$ is of size $n \times n$.

Markov chains can be applied to Web link sequence modelling. In this formulation, a Markov state can correspond to any of the following:

- URI/URL
- HTTP request
- Action (such as a database update, or sending e-mail)

The matrix $\boldsymbol{A}$ can be estimated using many methods. Without loss of generality, the maximum likelihood principle is applied in this paper to estimate $\boldsymbol{A}$ and λ. Each element of the matrix $\boldsymbol{A}[\boldsymbol{s}, \boldsymbol{s}']$ can be estimated as follows:

$$\boldsymbol{A}(\boldsymbol{s}, \boldsymbol{s}') = \frac{C(\boldsymbol{s}, \boldsymbol{s}')}{\sum_{\boldsymbol{s}''} C(\boldsymbol{s}, \boldsymbol{s}'')}, \tag{2}$$

$$\lambda(\boldsymbol{s}) = \frac{C(\boldsymbol{s})}{\sum_{\boldsymbol{s}'} C(\boldsymbol{s}')}. \tag{3}$$

$C(\boldsymbol{s}, \boldsymbol{s}')$ is the count of the number of times $\boldsymbol{s}'$ follows $\boldsymbol{s}$ in the training data. Although Markov chains have been traditionally used to characterize asymptotic properties of random variables, we utilize the transition matrix to estimate short-term link predictions. An element of the matrix $\boldsymbol{A}$, say $\boldsymbol{A}[\boldsymbol{s}, \boldsymbol{s}']$ can be interpreted as the probability of transitioning from state s to s' in one step. Similarly an element of $\boldsymbol{A} * \boldsymbol{A}$ will denote the probability of transitioning from one state to another in two steps, and so on.

Given the 'link history' of the user $L(t-k)$, $L(t-k+1) \ldots L(t-1)$, we can represent each link as a vector with a probability 1 at that state for that time (denoted by $\boldsymbol{i}(t-k)$, $\boldsymbol{i}(t-k+1) \ldots \boldsymbol{i}(t-1)$). The Markov chain models estimation of the probability of being in a state at time t is shown in Eq. (4).

$$\hat{\boldsymbol{s}}(t) = \hat{\boldsymbol{i}}(t-1)\boldsymbol{A}. \tag{4}$$

The Markovian assumption can be varied in a variety of ways. In our problem of link prediction, we have the user's history available; however, a probability distribution can be created about which of the previous links are 'good predictors' of the next link. Therefore we propose variants of the Markov process to accommodate weighting of more than one history state. In the following equations, we can see that each of the previous links are used to predict the future links and combined in a variety of ways. It is worth noting that rather than compute $\boldsymbol{A} * \boldsymbol{A}$ and higher powers of the transition matrix, these may be directly estimated using the training data. In practice, the state probability vector $\boldsymbol{s}(t)$ can be normalized and thresholded in order to select a list of 'probable links/states' that the user will choose.

$$\hat{\boldsymbol{s}}(t) = a_0\hat{\boldsymbol{i}}(t-1)\boldsymbol{A} + a_1\hat{\boldsymbol{i}}(t-2)\boldsymbol{A}^2 + a_2\hat{\boldsymbol{i}}(t-3)\boldsymbol{A}^3 \ldots, \tag{5}$$

$$\hat{\boldsymbol{s}}(t) = \mathrm{Max}(a_0\hat{\boldsymbol{i}}(t-1)\boldsymbol{A}, a_1\hat{\boldsymbol{i}}(t-2)\boldsymbol{A}^2, a_2\hat{\boldsymbol{i}}(t-3)\boldsymbol{A}^3 \ldots). \tag{6}$$

3. System overview

Fig. 1 shows the overview of a typical architecture utilizing the Markov chain link prediction system. The major components of the probabilistic predictor are the following.

(a) *The Markov chain model*. The Markov Chain model consists of a (sparse) matrix (compressed to an appropriate form) of state transition probabilities, and the initial state probability vector. These are stored in the form of both counts and probabilities.

(b) *Client path buffer*. All client requests are buffered into a client buffer, and flushed once a minimum sample threshold is exceeded, or the session timeouts. Each client is assigned a separate buffer, and the sequence of client requests stored in the buffer.

(c) *Adaptation module*. This module updates the Markov chain model with user path trace information available to the system. The update is typically achieved by smoothing the default/current count matrix with the counts derived from the additional path sequences available.

(d) *Tour generator*. Given a start URL, the tour generator outputs a sequence of states (URL/URIs) which corresponds to the tour generated by the model. A tour generation algorithm is described in the next section.

(e) *Path analysis and clustering*. The path analyser currently extends the hub/authority [8] weight estimation algorithm using the Markov transition matrix. The clustering module (still under development) is used to cluster the states into 'similar

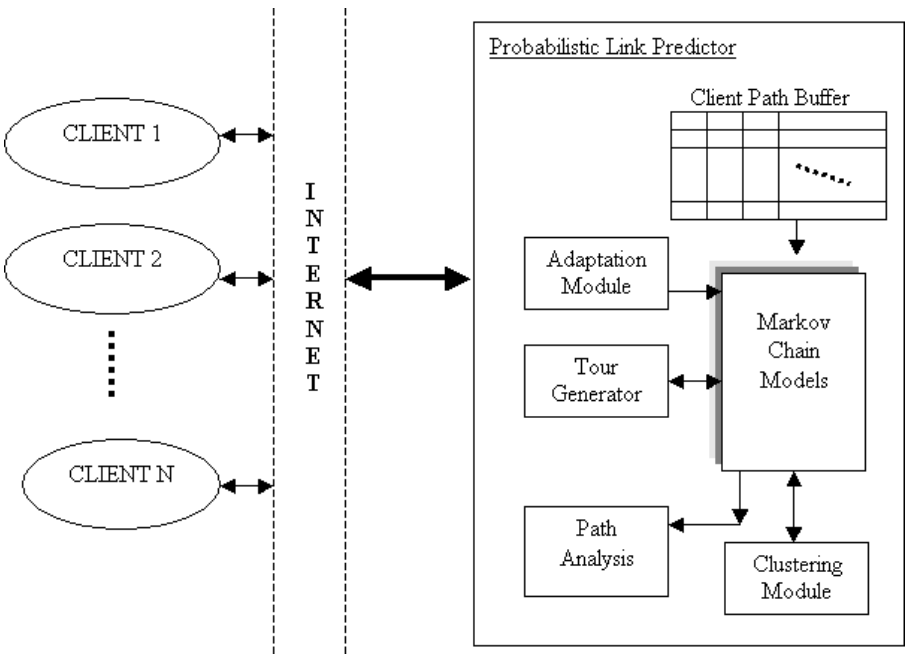

Fig. 1. Overview of the probabilistic link prediction system.

groups' in order to reduce the dimensionality of the transition matrix. A last component that is not shown in the figure is the URL table, which retains the URL (string) for each state index.

4. Applications

In this section, four applications of Markov chains to link prediction and path analysis are discussed.

4.1. Web server HTTP request prediction

The first application of the probabilistic link prediction discussed in this paper is HTTP request prediction. Extensive work (e.g. see [1]) has been done on the analysis of HTTP requests in order to enhance server performance. Most of the work involves statistical analysis of request file sizes, request patterns, and caching mechanisms. Recently, Schechter et al. [13] discuss methods of building a sequence prefix tree using path profiles and using the longest matched most-frequent sequence to predict the next request. To our knowledge, probabilistic sequence generation models such as Markov chains have not been applied to the problem of HTTP request prediction.

The incorporation of the Markov chain models and its extensions into a server or proxy is quite straightforward. The client sends a request to the Web server (or proxy) which uses the HTTP probabilistic link prediction module in order to predict the probabilities of the next requests from the same user based on the history of requests from that client. The server can also use the Markov chain model in an adaptive mode, updating the transition matrix using the sequence of requests that arrive at the Web server.

4.2. Application 2: adaptive Web navigation

The second application of the Markov chain probabilistic link predictor is system-aided Web navigation. Link prediction is used to build a navigation agent which suggests (to the user) which other sites/links would be of interest to the user based on the statistics of previous visits (either by this particular user or a collection of users). In the current framework, the predicted link does not strictly have to be a link present in the Web page currently being

viewed. This is because the predicted links are based on actual user traversal sequences which can include explicit user jumps between disjoint Web sites.

If the link modelling is user-specific then the link predictor module can be resident at the client side rather than the server side. In the architecture that we have implemented, we have the link predictor as a servlet in the server side. Whenever a client clicks on a link, this information is posted to the link predictor servlet, which processes this link and suggests a list of possible links that the user can go to next. These links are ordered according to the probability of prediction (and thresholded to discard low-probability links).

4.3. Application 3: tour generation

The tour generator module is given as input the starting URL (e.g. the current document the user is browsing). The tour module generates a sequence of states (or URLs) using the Markov chain process. This is returned and displayed to the client as a tour. A simple example of such a tour is presented in the experimental section.

In order to demonstrate tour generation with Markov chains we implemented the following *Tour generation Using Markov Models* (TUMMs) algorithm. The tour generation uses the Markov model to predict a sequence of states (URLs) to visit next. Since the chain can generate a cyclic sequence of links, we can mark each state as either 'visited' or 'unvisited'. Furthermore, in the case of a tie (i.e. multiple states have the same probability), a mechanism of choosing the next state should be formulated. Lastly, if the outgoing probability of all states from the current state is below some threshold, then the facility to 'restart' should be provided. Note that this can be extended in a variety of ways including better handling of 'restarts', and tie-breaking mechanisms. In the TUMMs algorithm, ties are broken by choosing the first link with the longest matching prefix URL as the parent.

Algorithm TUMMs

(1) Set start state to be $s0$;
(2) Mark start state $s0$ as already visited. $s' = s0$
(3) While the length of tour not reached or not(Exit Criteria) do thru' step (9)
(4) For all unvisited states s:
(5) Compute $P(s' \rightarrow s)$ using the Markov chain model
(6) Choose the $\text{Max}(P(s' \rightarrow s))$ and let the corresponding set of states be S
(7) If $|S| > 1$ then pick the state s'' from S, such that URL(s'') and URL(s') have the maximal URL path prefix match (further ties arbitrarily broken).
(8) If $|S| = 0$ then restart (for example: $s'' = s0$)
(9) $s' = s''$
(10) End of tour generation procedure

4.4. Application 4: personalized hub/authority

The notion of 'hubs/authorities' [8] is typically applied on the Web graph structure. Hubs refer to Web sites that are often good starting points to find information. Authority refers to Web sites that contain a lot of useful information on a particular topic. The term 'personalized' is used here (somewhat loosely) to pertain to a specific set of users, or a specific type of sites.

'Personalized hubs/authorities' extend the notion of hubs/authorities to focus on a specific group of users/sites *using the path traversal* patterns that are collected. The Markov state transition matrix is a representation of the users traversal patterns, and can be viewed as a 'traversal connectivity' matrix. Thus, the idea of iterative estimation of hub and authority weights using this Markovian transition matrix can be applied to extract the prominent 'personalized' hubs/authorities. The algorithm is similar to the one described in [8] with the important difference of initialization of the hub and authority weights using the transition probabilities specified by the Markov chain transition matrix.

5. Experimental results

In this section we report various experimental results. The first subsection describes a number of simulation experiments, followed by real-data experiments on HTTP request prediction, link prediction, and tour generation. Lastly we discuss results obtained by eigenvector decomposition of the Markov chain matrix to obtain 'personalized hubs/authorities'.

5.1. Simulations

Each simulation consists of generating a link connectivity for a specified number of states/links. The density of link connectivity was generated to be within a specified value. Two million training samples and twenty thousand test samples are generated from the same random process for each experiment. Each data point in the graphs shown corresponds to ten trials. Some trials are discarded due to zero probability link connectivity. For each test sample, the probability of being in any state at that time is computed using the various methods detailed earlier. 'Correct link' refers to the actual link chosen at the next step. The rank/depth of the correct link is measured by counting the number of links which have a probability greater than or equal to the correct link.

Fig. 2 shows the results of varying number of links on the average depth of the correct link. The link connectivity density of the set of simulations depicted in Fig. 2 is ~6.5%. It can be seen that the average depth of the correct link slightly increases with increasing dimensionality.

Next, we studied the effect of the link connectivity density on the average depth of the correct link (see Fig. 3). As expected, with increasing densities, the average depth of the correct link increases, although non-linearly.

In real-world situations, the user's navigation history is often noisy. The user may go through a few pages, then get distracted by an advertisement, and then return to his/her original navigation goal. In order to simulate such conditions, noisy state sequences are added to the test data (whose underlying probabilistic generator is the same as the training data). Experiments are conducted for different levels of noise in the test data (note that the training data is uncorrupted in these experiments). Fig. 4 shows the results of link prediction in the presence of noise. PH represents 'prediction history' as applied in Eq. (5) and the weighting coefficients *a* being uniform. Fig. 4 shows that for low noise data, using single history is sufficient, whereas when the data get noisier, 'voting' with probability estimates from more than one history is useful.

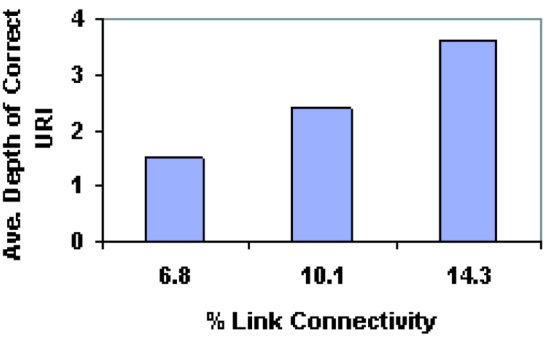

Fig. 3. Effect of linkage density on average depth of correct link.

5.2. HTTP server request prediction

Next we turn to real server log data experiments. The EPA-HTTP data server logs collected at Research Triangle Park, NC, are used in the server-log experiments (and downloadable at http://ita.ee.lbl.gov/html/traces.html). Since we are interested in applying the techniques to user-oriented patterns, the

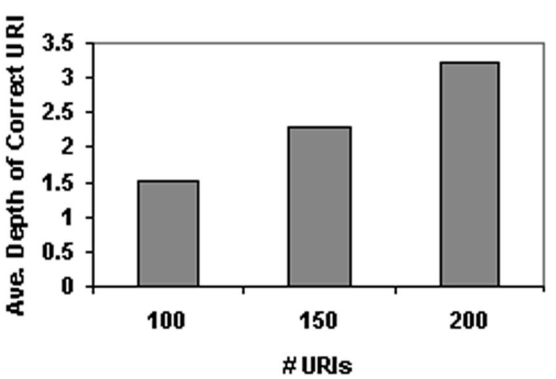

Fig. 2. Effect of dimensionality on average depth of correct link.

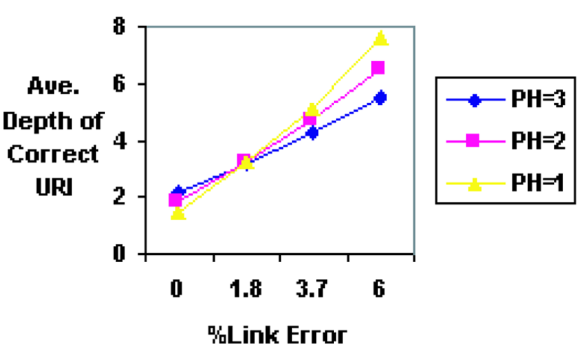

Fig. 4. Effect of noise on the probabilistic link prediction technique.

server logs are sorted according to the originating host (assuming that each session host corresponds to the same user). The total number of unique URIs (including html documents, directories, gifs, and cgi requests) is 6572. The total number of samples in the server log data is 47,751. 40,000 samples were used as training data, and the remaining as test data.

Fig. 5 shows the results of experiments on the EPA-HTTP server logs. It can be seen that using the Markov chain prediction technique, over 50% of the Web server requests can be predicted to be the state with the highest probability. Even with just the top twenty links, slightly less than 70% of the requests can be predicted with the proposed probabilistic predictor.

5.3. Link prediction

The next experiment performed with the EPA-HTTP log data is link prediction. For this experiment, the data consist of only html documents (i.e. discarded images and cgi posts). The results of such an experiment are shown in Fig. 6 ('static' columns). The test set consisted of 3576 samples, and a Markov chain ($H = 1$) was used. When compared to Fig. 5, we see that the performance of link prediction is better than generic HTTP server request prediction.

The Markov chain model can also be used in an *adaptive* manner. In this case, the (test) sequence of links that have already been evaluated are also added to the training data, and the Markov chain model dynamically adapted. This updated adapted model is used to probabilistically predict the next link. The

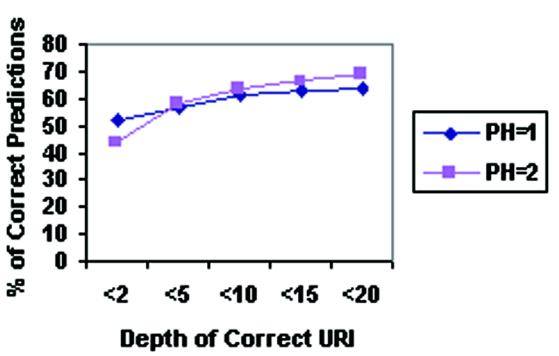

Fig. 5. Probabilistic link prediction on the EPA-HTTP server log data.

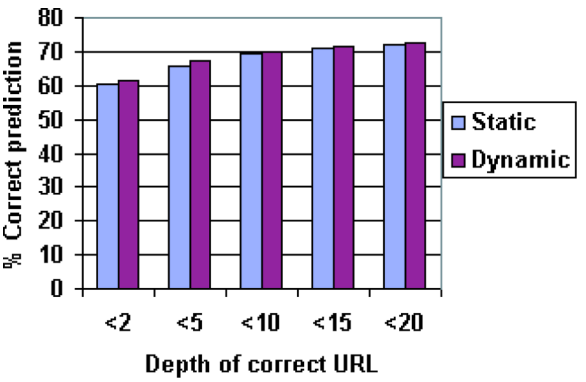

Fig. 6. Probabilistic link prediction on the EPA-HTTP server log data without gif's, cgi requests, and xbm's.

results of such an experiment are shown in Fig. 6. It should be noted that the adaptive model results in a relative error rate reduction of 3.5% for the '<2' case in comparison to the static model.

5.4. Tour generation

It is difficult to quantitatively evaluate the tours generated by the model. Some anecdotal evidence is provided to give the reader an idea of the types of tour generated. Shown below is a sample tour (limit of 20 URLs) generated using the TUMMs algorithm.

(1) /docs/ozone/index.html (Starting URL)
(2) /docs/ozone/science/science.html
(3) /docs/ozone/science/q_a.html
(4) /docs/ozone/science/process.html
(5) /docs/ozone/science/marcomp.html
(6) /docs/ozone/defns.html
(7) /docs/ozone/mbr/mbrqa.html
(8) /docs/ozone/events.html
(9) /docs/ozone/othlinks.html
(10) /docs/GCDOAR/OAR-APPD.html
(11) /docs/GCDOAR/gcd-goal.html
(12) /docs/GCDOAR/EnergyStar.html
(13) /docs/GCDOAR/esrp.html
(14) /docs/GCDOAR/homes.html
(15) /docs/GCDOAR/geothermal.html
(16) /docs/ozone/index.html
(17) /docs/ozone/olaw/olaw.html
(18) /docs/ozone/olaw/olawlet.html
(19) /docs/ozone/index.html
(20) /docs/ozone/olaw/consumer.html

The above tour can be summarized as follows.
(a) Start at the index pages on Ozone.
(b) Go to the science section, and the question and answer section.
(c) Later stop at the definitions, current events.
(d) Examine the GCDOAR subpages (which is reached from the Ozone 'other links' page).
(e) Finally browse through the law section.

It should be noted that the above tour has been generated just using the EPA-HTTP logging done for a short period. Thus, the tour reflects the browsing patterns for the duration of the EPA-HTTP logging. Furthermore, we did not include session markers (sessions were not explicitly marked in these data) between sessions of different users, and that could account for the topic drift. Another advantage of the Markov chain approach to tour generation is the ability to start a tour from any URL (which has been traversed before).

5.5. Personalized hubs/authorities

The hub/authority iterative algorithm described in [8] is applied using the Markov transition matrix obtained using the EPA-HTTP server logs (no images/cgi requests). The important difference is in the initialization of the hub and authority weights. Typically, the Web graph connectivity structure is used to determine the hub and authority weights. In our case, the initial authority weight of a state is initialized to be the sum of the transition probabilities of all other states into that state as determined by the Markov chain transition matrix. Similarly, the hub weight for a state is initialized to be the sum of the transition probabilities of all the states that are reachable from that state. Note that this is entirely based on the user traversal patterns, and not the static structure of the Web linkage.

Table 1 lists a few of the top hubs and authorities selected by the application of the modified hub/authority algorithm on the Markov chain transition matrix extracted using EPA-HTTP client requests.

Table 1
Identified hubs and authorities using path analysis

Hub	Authority
/	/
/Rules.html	/Rules.html
/docs/CSO.html	/docs/WhatsHot.html
/Access/	/Info.html
/docs/WhatsNew.html	/Software.html
/docs/titlesearch.html	/Offices.html
/docs/WhatsHot.html	/docs/Access
/Research.html	/docs/WhatsNew.html
/Information.html	/PIC.html
/Software.html	/Research.html
/Standards.html	/News.html
/docs/Welcome/EPA.html	/docs/Contacts

6. Limitations of the current approach

The Markov chain approach has the following limitations.

(a) *Amount of training data*. Since the approach is statistical, by definition, the goodness of the model is dependent on the amount of data available. This is usually not a problem when modelling a particular site with high visitation, but becomes more difficult when multi-site analysis with low visitations are required.

(b) *Dimensionality*. The second problem is one of dimensionality. The Markov chain matrix is typically very large ($N \times N$ for N URIs). This is clearly not scalable for very large numbers of sites. Such a dimensionality problem may be addressed by modelling site-specific transition models and within-site transition models. Another approach would be to cluster sites based on their access patterns before applying the Markov chain analysis. Finally, the Markov chain matrix is usually very, very sparse for high-dimensional matrices, and sparse matrix storage representations can reduce the memory requirements.

7. Related work

Perkowitz and Etzioni [11] discuss the notion of adaptive Web sites which semi-automatically improve their organization by learning from visitor access patterns. The PageGather algorithm uses page co-occurrence frequencies to find clusters of related but unlinked pages. Based on the PageGather algorithm, index pages are created for easier navigation.

In [18], data mining and data warehousing techniques are used to analyse Web log records. The Web log miner is implemented through the following phases: (a) data cleaning and transformation, (b) a multi-dimensional Web data cube is constructed, (c) OLAP (Online Analytical Processing) operations are performed on this data cube. [2] is also an earlier application of data mining approaches to Web path analysis.

Shahabi et al. [14] describe a remote Java agent that captures client's selected links and page orders, accurate page viewing time, and cache references. Link path sequences are enumerated and clustering in this path space is done using the cosine angle distance metric. While the link sequence information is used in this model, the link prediction is not probabilistically based on the frequency of user navigation profiles.

Intelligent agents that detect topics of interest to the users are described in [3]. The Web pages visited by users are analysed and topic spotting performed. This enables suggesting Web documents based on the topic of interest demonstrated by the user's navigation.

In [17], each user session is represented as an N-dimensional vector capturing the frequency of access to different documents within the site. These collections of vectors are clustered based on users' interests and the clusters used to determine which pages are most interesting to the particular set of users. Sequence information is ignored in this analysis.

Wexelblat and Maes [15] describe the footprint system which provides a metaphor of travellers creating footpaths which other travellers can use. The Web site map that is displayed is adapted through a modified Boltzmann algorithm "which works by computing attraction and repulsion forces among objects" [16].

A Web tour guide is described in [7] where the agent guides the user along an appropriate path of links based on the user's interests. WebWatcher also uses words from the document to detect the topics of interest to the user, and estimates link probabilities using TF-IDF heuristic using the extracted keywords. A second approach used by WebWatcher is based on reinforcement learning where each link is represented as a state in the reinforcement learning state space, and the rewards correspond to the TF-IDF measures.

Numerous approaches to Web client access modelling have been done in relation to server caching modelling. Notably many researchers have found that the Web access follows a Zipf distribution. Various statistical properties are extracted using server logs including file sizes of transferred data, relative document popularity etc. Researchers have studied performance of various caching strategies in light of these statistical properties of server requests (e.g. [1]).

Schechter et al. [13] present a mechanism of constructing path profile maximal prefix trees using client HTTP requests. However, there is no probabilistic weighting of the paths: the maximal matching most-frequent prefix is used to predict the next request. Kraiss and Weikum [9] apply continuous Markov chains to influence caching priorities between primary, secondary and tertiary storages, and report experimental results on synthetic workloads.

All the above techniques do not directly try to capture the sequence of link traversal in a probabilistic manner. The novelty of our approach stems from the application of probabilistic link sequence modelling using maximum likelihood estimation of Markov models and robust extensions. Furthermore, the estimation is purely data-driven and statistical in nature, techniques commonly used in various other sequence modelling applications such as speech recognition.

8. Conclusion

The rapid growth of the World Wide Web makes good models for link prediction and path analysis absolutely crucial. Markov chain models lend themselves as viable models for Web sequence modelling. Markov chain models can be estimated statistically, adaptively, and are generative. This facilitates the application of Markov chains to HTTP server request prediction, link prediction, tour generation and identification of 'personalized' hubs/authorities.

The experimental results obtained by applying Markov chains to the above problems are very promising. Web server requests can be predicted correctly, using the highest probability state, over

50% of the time on the EPA-HTTP data. Links can be predicted correctly with the highest state probability over 60% of the time, and over 70% of the correct links are in the top 20 scoring states. A novel algorithm for tour generation (TUMMs) using Markov chains is presented and anecdotal experimental results presented. Finally, the state transition matrix of the Markov chain model can be viewed as a 'weighted traversal' representation of the user's model of the World Wide Web, and further analysis can be done on this matrix. We demonstrated the application of the hub/authority weight estimation procedure in order to generate 'personalized' hubs/authorities from client navigation profiles. The results suggest that Markov chains are useful tools for Web link sequence modelling and path analysis.

Acknowledgements

We thank Prof. S.K. Rangarajan for useful discussions on Markov chains. We are also grateful to Prof. Mark Crovella and Dr. Barford of Boston University for pointing us to the server log data. We also thank Dr. Sekhar Sarukkai of HP Labs for many useful discussions on server performance issues. The EPA-HTTP logs were collected by Laura Bottomley of Duke University.

References

[1] P. Barford, A. Bestavros, A. Bradley and M. Crovella, Changes in web client access patterns: characteristics and caching implications, to appear in World Wide Web, special issue on Characterization and Performance Evaluation.
[2] M.S. Chen, J.S. Park and P.S. Yu, Data mining for path traversal in a web environment, in: Proc. 16th International Conference on Distributed Computing Systems, Hong Kong, May 1996.
[3] D.W. Chueng, B. Kao and J.W. Lee, Discovering user access patterns on the World-Wide Web, in: Proc. First Pacific–Asia Conference on Knowledge Discovery and Data Mining (PAKDD-97).
[4] W. Feller, Introduction to Probability Theory and Its Applications, Vols. 1 and 2, Wiley, New York, 1971.
[5] J. Garofalakis, P. Kappos and D. Mourloukos, Web site optimization using page popularity, IEEE Internet Computing, July–August 1999, pp. 22–29.
[6] D. Gibson, J. Kleinberg and P. Raghavan, Inferring web communities from link topology, in: Proc. 9th ACM conference on Hypertext and Hypermedia, 1998.
[7] T. Joachims, D. Freitag and T. Mitchell, WebWatcher: a tour guide for the World Wide Web, IJCAI'97.
[8] J. Kleinberg, Authoritative sources in a hyperlinked environment, in: Proc. 9th ACM-SIAM Symposium on Discrete Algorithms, 1998.
[9] A. Kraiss and G. Weikum, Integrated document caching and prefetching in storage hierarchies based on Markov-chain predictions, The VLDB Journal 7 (7) (1998) 141–162.
[10] S. Lawrence and C.L. Giles, Accessibility of information on the Web, Nature 400 (1999) 107–109.
[11] M. Perkowitz and O. Etzioni, Towards adaptive Web sites: conceptual framework and case study, WWW8, Toronto, 1999.
[12] R.R. Sarukkai, Fundamentals of Internet Technology, book proposal in progress.
[13] S. Schechter, M. Krishnan and M.D. Smith, Using path profiles to predict HTTP requests, WWW7, 1998.
[14] C. Shahabi, A.M. Zarkesh, J. Adibi and V. Shah, Kowledge discovery from users Web-page navigation, IEEE RIDE 1997.
[15] A. Wexelblat and P. Maes, Footprints: history-rich tools for information foraging, CHI'99.
[16] A. Wexelblat and P. Maes, Visualizing histories for web browsing, RIAO'97, Computer Assisted Information Retrieval on the Internet, Montreal, 1997.
[17] T.W. Yan, M. Jacobsen, H. Garcia-Molina and U. Dayal, From user access patterns to dynamic hypertext linking, 5th International World Wide Web Conference, May 1996.
[18] O.R. Zaiane, M. Xin and J. Han, Discovering Web access patterns and trends by applying OLAP and data mining technology on Web logs, in: Proc. Advances in Digital Libraries Conf. (ADL'98), Santa Barbara, CA, 1998, pp. 19–29.

Ramesh Sarukkai received his MS and PhD degrees in computer science from the University of Rochester, NY. Prior to his current position at Yahoo! Inc., Dr. Sarukkai spent two summers at IBM Watson Research Center and was a senior scientist at L& H. His research interests range from Internet applications, agents, Web information retrieval to speech technology and machine learning. He holds an Internet patent, and many publications in the above areas. He is also a member of the World Wide Web Consortium (W3C) Working group on Voice Browsers.