Are Web Users Really Markovian?

Flavio Chierichetti[*]
Dept. of Computer Science
Cornell University
Ithaca, NY 14853
flavio@cs.cornell.edu

Ravi Kumar
Yahoo! Research
701 First Avenue
Sunnyvale, CA 94089
ravikumar@yahoo-inc.com

Prabhakar Raghavan
Yahoo! Labs
701 First Avenue
Sunnyvale, CA 94089
pragh@yahoo-inc.com

Tamás Sarlós
Yahoo! Research
701 First Avenue
Sunnyvale, CA 94089
stamas@yahoo-inc.com

ABSTRACT

User modeling on the Web has rested on the fundamental assumption of Markovian behavior — a user's next action depends only on her current state, and not the history leading up to the current state. This forms the underpinning of PageRank web ranking, as well as a number of techniques for targeting advertising to users. In this work we examine the validity of this assumption, using data from a number of Web settings. Our main result invokes statistical order estimation tests for Markov chains to establish that Web users are not, in fact, Markovian. We study the extent to which the Markovian assumption is invalid, and derive a number of avenues for further research.

Categories and Subject Descriptors. F.2 [**Theory of Computation**]: Analysis of Algorithms and Problem Complexity; G.3 [**Mathematics of Computing**]: Probability and Statistics

General Terms. Measurement, Theory

Keywords. Markov chains, Browsing behavior, User models

1. INTRODUCTION

The *Markovian* model for Web user behavior posits that when a user is browsing a Web page P, the next page she visits depends only on P and not on how the user arrived at P. This assumption is central to some of the most widely used Web algorithms and systems including Google's PageRank [15] and other forms of link analysis [13]. Markovian user models have also been proposed for advertising [1] and in fact, many systems used for behavioral targeting of advertisements use an even simpler *zeroth order* model in which the next page visited is drawn from a probability distribution that is independent of the user's current position. The central question we examine in this paper is: how valid are these simple (Markovian, as well zeroth order) models? Using data from a variety of different sources, we establish that the Markovian model (and by corollary the zeroth order model) is too simplistic to capture the behavior of Web users. Our data includes browser trails on large networks including Yahoo! as well data from user behavior *within* a page, using mouse- and eye-tracking. In other words, user behavior on the Web is rather more intricate than the simple models underlying the most commonly used algorithms on the Web.

Given a set of hyperlinked Web pages, we may view each page as a state in a Markov chain [12]. Each hyperlink is a potential transition of the Markov chain modeling a user following that hyperlink. The transition probabilities of the Markov chain represent the probabilities of the user following each hyperlink, if she is at the page containing that link. In some of the settings we study, we will naturally model a slot within a page as a state, when considering user actions within the page. In these settings we study the drift of the user's mouse or eyes over the page as transitions between states. Here too (as we will detail) one could naturally model the user's behavior using a Markov model.

Given this view of pages visited as states in a Markov chain, we next extend the notion of a Markov chain (in a manner routine in probability theory) to a richer class of user models that includes as special cases the Markovian and zeroth order models. Consider the probability that a user at stage (page) i goes next to page j. If for all j, the transition probability is independent of i, then we say the user model is *zeroth order*. If instead it is uniquely determined by i then we say the user model is *Markovian*; we will sometimes refer to this as the first order model. Thus in the Markovian model, the probability of going from i to j varies with i, but depends only on i and not on how the user arrived at state i. More generally for $k > 1$ we say the user model is of order k if it is the smallest integer such that the probability of going to page j is determined by the sequence $s_k, s_{k-1}, \ldots, s_2, i$ of the last k states (pages) visited by the user. Thus in a second order user model, the transition probabilities depend on the current page i as well as the previous page the user visited prior to arriving at i.

Intuitively the larger k is, the greater the influence of the user's historical trail on her behavior. The zeroth order model even ignores which pages are linked to which others; it simply views the user's next page as drawn from a fixed probability distribution independent of her current position. The (first order) Markovian model does take into account the user's current state (page) i and thus can take into account the links out of i; it ignores states visited prior to i and in this sense may be considered *memoryless*. Classic Web algorithms such as PageRank use this model. Some prior work [20, 14] offers weak evidence in support of users' behavior being Markovian. We know of no prior work that has examined whether

[*] Flavio Chierichetti was supported in part by the NSF grant CCF-091094.

WWW 2012, April 16–20, 2012, Lyon, France.
ACM 978-1-4503-1229-5/12/04.

Web users' behavior is in fact truly Markovian, thereby justifying the assumption implicit in PageRank and other algorithms. Aside from these applications, we believe the question "how memoryless are users' browsing habits" is of interest in its own right, as an important step in understanding user behavior.

We bring together two ingredients in addressing this question. First, we study a number of data sets each consisting of a large number of user trails; in some of these data sets the user trails visit Web pages. In others, each trail follows the user through other discrete steps abstracted as states. Second we appeal to a powerful result from statistics that considers the error in a model's power to explain given data as a function of increasing k. Informally the result [16] asserts that the magnitude of the error undergoes a sharp drop from order $k-1$ to the true order k. This enables us to ask the question: what is the smallest k at which a given set of user trails is adequately explained by the model?

The rest of the paper is organized as follows. In Section 3 we introduce our notation and the basics of Markov chains. We prove that several natural questions are computationally or information-theoretically hard to answer in Sections 4.1 and 5.2. The description of our novel algorithms for optimally estimating higher order Markov chains is given in Section 4. Finally we present our extensive experimental study using a diverse collection of large scale Web data sets in Sections 7 and 8.

2. RELATED WORK

Estimating the order of a Markov chain has been extensively studied by the statistics community [7, 16]. Multiple order estimators that are asymptotically consistent as the data size tends to infinity are known [7]; however, to the best of our knowledge, their finite sample convergence has not been investigated carefully. In fact, in Section 4.1 we show that the number of samples required for distinguishing between order 1 and order k Markov chains grows extremely rapidly in the worst case.

Variable order Markov chains (VOMC) were introduced in [4] though similar ideas were considered earlier in context-dependent data compression by Rissanen [18]. Ron et al. [19] gave a polynomial time algorithm that learns a VOMC such that the probability distribution of the emitted state sequences has small Kullback–Leibler divergence from those generated by the true source. Dalevi et al. [8] extended the recent order estimation algorithm of Peres and Shields [16] to VOMCs and conducted experiments with DNA sequences comparing the accuracy of several algorithms.

There has been some work on empirically modeling user browsing patterns with first [20, 14], second [22, 21], and higher order [17] Markov chains. Borges [3] fit VOMC to session logs and Deshpande and Karypis [10] studied the compression and pruning of higher order Markov models. However experimental evaluation has generally been limited to web access logs of a few relatively small web sites, e.g., a computer science department's or a merchant's web site, raising issues with the homogeneity, representativeness, and insufficient scale of the data. For a thorough overview of sequence prediction algorithms applied to learning web request patterns we refer the reader to the excellent survey of Davison [9].

First order Markov chains [2, 6] and variable order hidden Markov models [5] have often been applied to context-aware search, document re-ranking, and query suggestion as well.

3. PRELIMINARIES

In this section we describe the background material necessary for understanding higher order Markov chains. First we define a general kth order Markov chain. Let $v_1, \ldots, v_n$ be the elements of the state space S.

DEFINITION 1 (ORDER k MARKOV CHAIN). *A Markov chain of order k is a process $(X_i)_{i=1}^{\infty}$ such that for each $t \geq k$ and $v_{k+1}, \ldots, v_1$, it holds that*

$$\begin{aligned}\Pr\left[X_{k+1} = v_{k+1} \mid X_k = v_k, \ldots, X_1 = v_1\right] &= \\ \Pr\left[X_{t+1} = v_{k+1} \mid X_t = v_k, \ldots, X_{t-k+1} = v_1\right].\end{aligned}$$

In other words, the next state in a kth order Markov chain depends on the identity of the k states leading up to the current state. Note that the traditional Markov chain is order 1 according to this definition since the next state depends only on the current state. Along similar lines, one can also define a zeroth order Markov chain where the next state distribution is independent of the current state. Next, we define the order of an element.

DEFINITION 2. *Let u be an element of a Markov chain of any order. Then u has order k if for each $t \geq k+1$ and $v_{k+1}, \ldots, v_1$, it holds that*

$$\begin{aligned}\Pr\left[X_{t+1} = v_{t+1} \mid X_t = u, X_{t-1} = v_{t-1}, \ldots, X_1 = v_1\right] &= \\ \Pr\left[X_{t+1} = v_{t+1} \mid X_t = u, X_{t-1} = v_{t-1},\right. \\ \left. \ldots, X_{t-k+1} = v_{t-k+1}\right].\end{aligned}$$

Observe that if a higher order Markov chain has elements $v_1, \ldots, v_n$ respectively of orders $k_1, \ldots, k_n$, then the Markov chain can be represented by $\sum_{i=1}^{n} \sum_{j=1}^{k_i} n^{j-1}$ probability vectors: for each element v_i, and for each possible sequence of length at most k_i leading to v_i, store the probability vector representing the next transition to be taken in the chain.

Now, given a set of paths where each path (sometimes called a *trail*) is a sequence of states, a natural computational question is: what is the order of the underlying stochastic process that generates these paths, or more specifically, what is the order of a generic element v_i? This is the Markov chain *order estimation problem.*

Note that a kth order Markov chain on a state space S can be seen as a first Markov chain on the larger state space $S' = \bigcup_{i=1}^{k} S^i$. Also, a sequence of traces generated by M can be interpreted as a sequence of traces generated by M': a trace $(a_1, \ldots, a_i)$ on M, can be seen as a trace

$$((a_1), \ldots, (a_1, \ldots, a_k), (a_2, \ldots, a_{k+1}), \ldots, (a_{i-k+1}, \ldots, a_i))$$

on M'.

4. MAXIMUM LIKELIHOOD ESTIMATION

Given an integer $k \geq 1$ and a set $\mathcal{T} = \{T_1, \ldots, T_t\}$ of traces, we wish to compute the kth order Markov chain that maximizes the probability of observing $\mathcal{T}$; such a Markov chain is called a *Maximum Likelihood Estimate (MLE)* for $\mathcal{T}$. Without loss of generality we assume that all trails start and end with a special reset state R that represents the unobservable components of the users' trails.

We will continue the discussion assuming $k = 1$, and at the end we will show how the $k \geq 2$ case reduces to the $k = 1$ case.

An easy algorithm to compute the Maximum Likelihood Markov chain (of order 1) is the following. For each sequence of the form $T_i = \left(x_{i,1}, x_{i,2}, \ldots, x_{i,|T_i|}\right)$, increase each of the counters

$$C_{x_{i,1} \to x_{i,2}}, C_{x_{i,2} \to x_{i,3}}, \ldots, C_{x_{i,|T_i|-1} \to x_{i,|T_i|}}.$$

Each of the counters starts at 0. Observe that if there are n states $x_1, \ldots, x_n$ plus the special reset state R, then the number of counters will be $(n+1)^2$.

The transition probabilities will be chosen as

$$M_{a\to b} = \frac{C_{a\to b}}{\sum_{\text{states } c} C_{a\to c}}. \qquad (1)$$

Observe that the ratio is well-defined iff at least one trace passes through state a. We will consider M to be a matrix whose rows are indexed by the source states and the columns are indexed by the destination states. The reset state R will index the first row and the first column.

Equation (1) can be easily extended to the higher order case using our observation from Section 3.

LEMMA 3. *The Markov chain M given by (1) is a Maximum Likelihood Estimate.*

PROOF. We prove for first order Markov chains; the proof easily extends to higher order Markov chains using the observation from Section 3.

Let N be the MLE Markov chain for traces $T_1, \ldots, T_t$, with $T_i = \left(x_{i,1}, x_{i,2}, \ldots, x_{i,|T_i|}\right)$. Let $C_{a\to b}$ be the number of times that the states a and b were consecutive in a trace.

The probability of observing the traces $T_1, \ldots, T_n$ with Markov chain N is given by

$$P = \prod_{i=1}^{t} \prod_{j=1}^{|T_i|-1} \left(N_{x_{i,j}\to x_{i,j+1}}\right) \quad = \prod_{x_i}\prod_{x_j} \left(N_{x_i\to x_j}\right)^{C_{x_i\to x_j}}.$$

Take any state a and consider its product $P_a = \prod_{x_j} \left(N_{a\to x_j}\right)^{C_{a\to x_j}}$. We have $P = \prod_a P_a$. Observe that P_a is the likelihood of a multinomial distribution. We conclude the proof by stating the following fact:

FACT 4. *Let $n_1, \ldots, n_k$ be positive integers, and consider the function*

$$f(p_1, \ldots, p_n) = p_1^{n_1} \cdot p_2^{n_2} \cdot \cdots \cdot p_k^{n_k}.$$

The function f, given the constraints $p_i \geq 0$, for $i = 1 \ldots, k$, and $\sum_{i=1}^{k} p_i = 1$, is uniquely maximized at $p_i = \frac{n_i}{\sum_{j=1}^{k} n_j}$, for $i = 1, \ldots, k$.

By Fact 4, the maximum likelihood is attained by setting $N_{a\to b} = \frac{C_{a\to b}}{\sum_c C_{a\to c}}$. Therefore, $M = N$. □

4.1 Learning the Markov chain

While the Maximum Likelihood Estimate is easy to compute, we now show that it is impossible to reconstruct the unknown Markov chain to any good approximation, unless we are given a very large number of samples.

We will start by showing that learning a kth order Markov chain is not feasible even just (i) in an *approximate* fashion, (ii) at *stationarity*, and (iii) if states are *chosen uniformly at random* out of an arbitrary support. That is, (i) if we allow some slack in learning the transition probabilities, and (ii) if the slack is not worst-case but, rather, it is averaged over states in a way that mimics how the user travels around the Markov chain, and finally (iii) even if the transition probabilities are uniform (as opposed to chosen so to make life harder for an algorithm) — the number of samples needed to learn the Markov chain grows exponentially in k.

The next construction also shows that making any guess on the order of the Markov chain (even allowing the three relaxations above) is impossible unless we are given a very large number of samples.

LEMMA 5. *There exists a Markov chain M of order $k = O(1)$, satisfying point (iii) above, for which the average expected ℓ_1 distance of the best guess of the next step distribution is $\Omega(1)$, unless $\Omega\left(n^k\right)$ steps of the Markov chain have been observed.*

Furthermore, guessing whether the order of the Markov chain is k or $k-1$ with probability $\Omega(1)$ is impossible unless $\Omega\left(n^{\frac{k-1}{2}}\right)$ steps are observed.

PROOF. Let $n-3$ be a multiple of k, and create a Markov chain with k layers with $(n-3)/k$ states each, plus three extra states. Let L_i be the set of states of the ith layer, $i = 1, \ldots, \frac{n-3}{k}$, and let the three extra states be R (the reset state), A, and B.

The transition probabilities are defined as follows:

- if we are at node R, the next node to be visited will be chosen uniformly at random in L_1;
- from each node $v \in L_i$, $i \in \{1, 2, \ldots, k-2, k-1\}$, the next node to be visited will be chose uniformly at random from L_{i+1};
- if we are at a node $v_k \in L_k$, and the history up to that point is $(v_1, v_2, \ldots, v_k)$, the next node will be $f(v_1, \ldots, v_k)$ with probability 1, where f is a function chosen uniformly at random (when constructing the Markov chain) between those with domain $L_1 \times L_2 \times \cdots \times L_k$ and codomain $\{A, B\}$;
- finally, if we are at either A or B, the next state will be the reset state R with probability 1.

Observe that, given that f is chosen uniformly at random, if we happen to be at a node $v_k \in L_k$, with a history $(v_1, \ldots, v_k)$, for the first time, it will be impossible for us to guess whether the distribution of the next node is degenerate in favor of A or B. Therefore, regardless of which distribution we guess for the next step, it will have average ℓ_1-distance to the actual one of at least 1.

Let $\mathcal{H} = \{(v_1, \ldots, v_k) \mid v_i \in L_i, i = 1, \ldots, k\}$.

Now, if we only observe $o\left(k \cdot \left(\frac{n}{k}\right)^k\right)$ steps in the Markov chain, there will be a fraction of $1 - o(1)$ histories in $\mathcal{H}$ that we will not have seen. Since each such history is equally likely, if we only observe $o\left(k \cdot \left(\frac{n}{k}\right)^k\right)$ steps, our best guess to the distribution of the next step, if we are at a state $v_k \in L_k$, will have an average ℓ_1-distance to the actual one of at least $1 - o(1)$. The main claim then follows by observing that any walk will be in a state of L_k a fraction $\Theta(k^{-1}) = \Theta(1)$ of the time.

For the second claim, suppose that an adversary chooses f either (a) uniformly at random from the set of functions $L_1 \times L_2 \times \cdots \times L_k \to \{A, B\}$, or (b) uniformly at random from the functions of that set that satisfy $f(x, v_2, v_3, \ldots, v_k) = f(y, v_2, v_3, \ldots, v_k)$, for each $x, y \in L_1$, and for each $v_i \in L_i$, $i = 2, \ldots, k$.

Observe that, with high probability (over the random choice of f), choice (a) produces a Markov chain of order k, and choice (b) produces a Markov chain of order at most $k-1$.

Now, unless a sub-walk $v_2, \ldots, v_k$, $v_i \in L_i, i = 2, \ldots k$, is repeated twice, it will be impossible for the algorithm to distinguish between choices (a) and (b). The probability that one such subwalk is repeated at least twice is at most $o(1)$, if the number of observed steps is $o\left(\sqrt{n^{k-1}}\right)$. □

The above result shows that learning Markov chains (and their order) is quite costly in terms of how many steps are needed, even under assumption (iii), i.e., the transition probabilities of the Markov chain are not very small.

We show that, if we drop assumption (iii), then there is no function of n and k that upper bounds the number of steps needed to distinguish between n-states Markov chains of order k and a Markov chain of order 1. That is, the order estimation problem becomes so hard that it cannot be solved with a number of steps upper bounded by any finite function of n and k.

We sketch an argument of why this is the case. Consider a Markov chain on n states such that the probability of transitioning from any state $x_i \neq x_1$, to any state x_j, is exactly $\frac{1}{n}$.

The transition from x_1 to any other state x_i, $i \geq 2$, is uniform, regardless of the history. The adversary makes a single choice: either

(a) the probability of transitioning from state x_1 to itself is $p_{\min} \ll \frac{1}{n}$, regardless of the history; or

(b) the probability of transitioning from state x_1 to itself is $p_{\min} \ll \frac{1}{n}$ if the history is *not* a run of k continuous x_1's, and $\frac{1}{2} \cdot p_{\min}$ otherwise.

Now observe that if the adversary makes choice (a), then the Markov chain has order 1. If, on the other hand, the adversary makes choice (b), the order of the Markov chain is k.

To guess whether the choice is (a) or (b) we have to traverse a $(k+1)$-long sequence of x_1's. Since the probability of following such a trail in the next $k+1$ steps is at most $\frac{1}{n} \cdot p_{\min}^{k+1}$, it follows that if we have less than $o\left(n \cdot p_{\min}^{-k-1}\right)$ steps to learn from we are not able to distinguish between choices (a) and (b), i.e., we cannot distinguish between Markov chains of order k and order 1 with fewer steps. Crucially, the lower bound does not depend on just n and k, but rather on the minimum non-zero probability in the Markov chain.

5. STATE COMPRESSION

So far we have only hinted at one issue of higher order Markov chains: their state space can be quite large. In this section, we deal with this general issue in two different ways.

First, we consider the variable order Markov chain estimation problem. We change the MLE problem definition to allow each state to have a different order, but we insist on finding the MLE of the Markov chain under the constraint that the sum of the orders of the states is bounded by some value.

A solution to this problem can be used to obtain a more parsimonious assignment of "memory" to states. As a byproduct, such a solution can be used to classify states in those that, roughly speaking, benefit from a "deeper" memory, and those that can be reasonably represented with a shorter one. In fact, we present such a classification in Section 8.3.

Then, we consider a different problem that tackles more directly the issue of the bit-size of a higher order Markov chain. We start from the observation that the highest cost in the memory representation of a Markov chain is not given by the identifiers of the states[1], but rather by the probability distributions that each state has on its out-neighbors. We therefore define the "compressed MLE" problem: if we are allowed to keep in memory at most t probability distributions, what is the maximum likelihood estimate for the higher order Markov chain?

We show that the variable order Markov chain MLE problem is solvable efficiently in polynomial time; on the other hand, we show that the compressed MLE problem is NP-hard.

5.1 The variable order MLE problem

In this section we propose a dynamic programming algorithm (Algorithm 1) for solving the variable order Markov chain MLE problem.

In the algorithm description, $P_k(v_i)$ is the product of the maximum likelihood probabilities of the trace steps going out of v_i, if we fix at k the order of node v_i. Such kth order maximum likelihood probabilities at v_i can be computed exactly as in the uniform-order MLE algorithm.

Algorithm 1 for solving the variable order Markov chain problem.

```
1: Let $K$ be the target total order, and let
   $A[0,\ldots,K], B[0,\ldots,K]$ be two vectors of size $K+1$.
2: Initialize every element of $A$ to 0.
3: $A[0] \leftarrow 1$
4: for all states $v_i$ do
5:     Initialize every element of $B$ to 0.
6:     for $k = 0,\ldots,K$ do
7:         for $j = 0,\ldots,K-k$ do
8:             $B[j+k] \leftarrow \max\left(A[j] \cdot P_k(v_i), B[j+k]\right)$.
9:     $A[j] \leftarrow B[j]$, for each $j = 0,\ldots,k+1$.
10: Let $j^*$ be an index that maximizes $A[j^*]$.
11: Let $i$ be equal to the number of states.
12: while $i \geq 0$ do
13:     Let $k^* \leq j^*$ be such that $A[k^*] = A[j^*] \cdot P_{j^*-k^*}(v_i)$.
14:     Choose a history of length $j^* - k^*$ for state $v_i$.
15:     $j^* \leftarrow k^*$
16:     $i \leftarrow i - 1$
```

The algorithm itself is a modification of classical dynamic programming algorithms. Thus, we do not provide a detailed analysis here and sketch its correctness instead.

After having iterated over states $v_1,\ldots,v_i$ at Line 4, $A[j]$, $j = 1,\ldots,K$, will contain the maximum product of probabilities $P_{j_1}(v_1)$, …, $P_{j_i}(v_i)$, with the constraint that $j_1 + \cdots + j_i = j$. A standard dynamic programming induction can be employed to show that at Line 10, the value of $A[j^*]$ is the maximum possible likelihood, given the total order constraint. The last part of the algorithm just unwinds the computation and reconstructs an order assignment that guarantees the maximum likelihood $A[j^*]$.

5.2 The compressed MLE problem

Here, we prove that the compressed MLE problem is NP-hard. Our proof works regardless of the order of the (possibly, variable order) Markov chain. The NP-hardness proof works as long as each state in the chain has positive order.

LEMMA 6. *The compressed MLE problem, for any order $k \geq 1$, is NP-hard.*

PROOF SKETCH. We reduce from the *Edge-Partition into Triangles* problem, shown to be NP-hard by Holyer [11]. Given an undirected graph $G = (V, E)$, the problem asks whether the edge set E, $|E| = m$, can be partitioned into $\frac{m}{3}$ triangles.

For each $e = \{v, w\} \in E$ we create two traces (x_e, x_v) and (x_e, x_w). Let $\mathcal{T}$ be the set of traces. We ask whether there exists a Markov chain for $\mathcal{T}$, using at most $\frac{m}{3} + 2$ distinct probability distributions over the out-neighbors, with likelihood at least $p = (3m)^{-2m}$.

First, suppose that a partition of E into triangles exists, i.e., let $S \subseteq \binom{V}{3}$, $|S| = \frac{m}{3}$, and for each $e \in E$ there exists $s \in S$ such that $e \subseteq s$. The out-distribution of the reset state R will be uniform over $\{x_e \mid e \in E\}$, i.e., for each $e \in E$, the probability of transitioning from R to x_e will be $\frac{1}{m}$.

Furthermore, for each $s \in S$, and for each $e \in E$ such that $e \subseteq s$, we assign to x_e the uniform out-distribution with support $\{x_v \mid v \in s\}$.

[1] We recall that in a higher order Markov chain each distinct "history" can be seen as a state.

Finally, for each $v \in V$, the state x_v will transition to R with probability 1.

An easy calculation shows that the probability that the above Markov chain produces the input traces $\mathcal{T}$ is exactly p. Also, the number of distinct out-distributions is $1 + |S| + 1 = \frac{m}{3} + 2$.

On the other hand, it can be shown that every Markov chain M satisfying the requirements and guaranteeing a likelihood of at least p, must contain exactly $\frac{m}{3}+2$ different out-distribution: one having support $\{R\}$, one having support $\{x_v \mid v \in V\}$, and each of the remaining $\frac{m}{3}$ having some support $S_i \in \binom{V}{3}$. Furthermore, for each $e \in E$, there must exist exactly one S_i containing it; these properties imply that the S_i's induce a partition into triangles of the edges of G. □

6. THE STATIONARITY OF THE MLE MARKOV CHAIN

In this section we characterize the stationary distribution of chains derived from trails and its connection to a prefetching problem.

Again, assume that we are given a sequence of traces $\mathcal{T} = \{T_1, \ldots, T_t\}$, and that we compute via Equation (1) the Maximum Likelihood Estimate M for $\mathcal{T}$. Let ℓ_i be the number of times that state x_i was visited in the input traces. Let ℓ_R be the number of traces. Finally, let $L = \ell_R + \sum_{i=1}^{n} \ell_i$ be the total number of visits to states in the input traces.

LEMMA 7. *Let the vector π be $\pi = \left(\frac{\ell_R}{L}, \frac{\ell_1}{L}, \frac{\ell_2}{L}, \ldots, \frac{\ell_n}{L}\right)$. Then, $\pi M = \pi$.*

PROOF. Observe that ℓ_i, $i = 1, \ldots, n$, is equal to

$$\ell_i = C_{x_i \to R} + \sum_{j=1}^{n} C_{x_i \to x_j} = C_{R \to x_i} + \sum_{j=1}^{n} C_{x_j \to x_i}.$$

Furthermore, ℓ_R is equal to

$$\ell_R = C_{R \to R} + \sum_{j=1}^{n} M_{R \to x_j} = C_{R \to R} + \sum_{j=1}^{n} M_{x_j \to R}.$$

Therefore,

$$\ell_R + \sum_{i=1}^{n} \ell_i = \sum_{i=1}^{t} (|T_i| + 1)$$

Let $\tau = \pi \cdot M$. Consider the x_i-coordinate of τ, for $i = 1, \ldots, n$. We have

$$\begin{aligned}\tau_{x_i} &= \frac{\ell_R}{L} \cdot M_{R \to x_i} + \sum_{k=1}^{n} \left(\frac{\ell_k}{L} \cdot M_{x_k \to x_i}\right) \\ &= \frac{\ell_R}{L} \cdot \frac{C_{R \to x_i}}{C_{R \to R} \sum_{j=1}^{n} C_{R \to x_j}} \\ &+ \sum_{k=1}^{n} \left(\frac{\ell_k}{L} \cdot \frac{C_{x_k \to x_i}}{C_{x_k \to R} + \sum_{j=1}^{n} C_{x_k \to x_j}}\right) \\ &= \frac{C_{R \to x_i}}{L} + \sum_{k=1}^{n} \left(\frac{C_{x_k \to x_i}}{L}\right) = \frac{\ell_i}{L} = \pi_{x_i}.\end{aligned}$$

The same derivation gives $\tau_R = \pi_R$. Therefore, $\pi \cdot M = \tau = \pi$, and the claim is proved. □

We observe that the Markov chain M is irreducible. This will allow us to claim that the π of Lemma 7 is the only stationary distribution of M.

OBSERVATION 8. *The Markov chain M is irreducible.*

PROOF. Since a state is part of M iff it was reached by at least one input trace starting from R and ending in R, and since each input trace has positive probability of being followed in M, it holds that M is irreducible. □

COROLLARY 9. *The vector π of Lemma 7 is the unique stationary distribution of M.*

Now consider a Markov chain M of order k on states $S = \{x_1, \ldots, x_n\}$, plus the "reset" state R. Such a chain can be seen as a first order Markov chain M' on state space $S' = \{R\} \cup \bigcup_{i=1}^{k} S^i$

A sequence of traces generated by M, can be interpreted as a sequence of traces generated by M'. By Corollary 9, there exists a stationary distribution π' for M'.

Recall that $\pi'(\sigma)$, for some $\sigma \in S'$, is the fraction of time that is spent on the multi-state σ by a random walk. Analogously, if we let $\overline{\pi}(x_i) = \sum_{\substack{\sigma \in S' \\ \text{the last state of } \sigma \text{ is } x_i}} \pi'(\sigma)$, we have that $\overline{\pi}(x_i)$ is the fraction of time that is spent on state x_i in a random walk over the kth order Markov chain M.

Let ℓ_σ be the number of times that the multi-state σ in the Markov chain M' is visited by the input traces, let L be equal to the sum of the ℓ_σ's plus the number of traces, and let ℓ_i be the number of times that state x_i in M is visited by the input traces. Then

$$\sum_{\substack{\sigma \in S' \\ \text{the last state of } \sigma \text{ is } x_i}} \ell_\sigma = \ell_i.$$

Therefore, we obtain that the fraction of time spent on state x_i in a random walk over the kth order Markov chain M is equal to:

$$\overline{\pi}(x_i) = \sum_{\substack{\sigma \in S' \\ \text{last state of } \sigma \text{ is } x_i}} \pi'(\sigma) = \frac{\sum_{\substack{\sigma \in S' \\ \text{last state of } \sigma \text{ is } x_i}} \ell_\sigma}{L} = \frac{\ell_i}{L}.$$

6.1 The prediction problem

We now use what we have developed so far in this section to solve the following prediction problem: suppose the user's browser is able to ask a content provider which page it should prefetch so to maximize the probability that, when the user clicks on a new link, the browser will be able to show the new page without performing other network operations. Which page should be suggested?

Given a Markov chain, the best page to prefetch is easy: given the user history up until that point, prefetch the state (page) that has largest probability of being clicked on (breaking ties arbitrarily).

With the following observation, we obtain the stationary efficiency of the best algorithm with a given stationary Markov chain. By stationary efficiency we mean the (asymptotic) fraction of times at which the page that was prefetched happens to be the one that the user clicked on. Again we state our result in terms of a first order Markov chain, but, as already noted, the higher order case reduces to the first order case.

LEMMA 10. *If π is the unique stationary distribution of M, then the efficiency of the best prefetching algorithm for M is*

$$\sum_{x} \left(\pi(x) \cdot \max_{y} M(x, y)\right).$$

PROOF. The probability of prefetching the right page, if we are at state x is exactly $\max_y M(x, y)$. At stationarity, we will spend a fraction $\pi(x)$ of the time at x. Hence, the statement follows. □

We observe that, if M is a maximum likelihood estimate obtained from a set of traces $\mathcal{T}$, then the terms $\pi(x)$ and $M(x, y)$ can be computed directly from the traces.

7. DATA

We use four data sets for our experiments. The first two deal with user behavior patterns across different pages in a website whereas the last two deal with user behavior patterns on a single page such as the search results page (SERP) or a content page. In all our data, we append a generic `reset` state to each trail and prepend a sequence of length k `reset` states so that the trails are all connected to one another, the underlying chain is ergodic, and `reset` will help "forget" the history across different trails.

Yahoo. This dataset, called Yahoo, is an anonymized sample of all user transitions that occurred in all Yahoo! websites. We restrict our attention to US-generated traffic and to the top 59 Yahoo! sites including `yahoo.com`, Mail, News, Sports, Finance, and so on; we also include a catchall `outside` state to capture transitions that leave the Yahoo! websites. The data was collected in July 2009 and consists of a large set of randomly sampled 1.1 billion cookies. The average length of the trails is around 46. A record for a cookie is of the form $\langle a, t, b\rangle_{i=1}$ where a, b are the names of the Yahoo! sites and t is the time at which the user left state a and entered state b, measured in seconds; the information about a was obtained using the HTTP referrer. We break the record for a cookie into trails whenever consecutive elements of the record cannot be pieced together, i.e., $b_{i-1} \neq a_i$ or either a_i or b_i is the `outside` state. Note that this can happen due to one of several reasons: the referrer string was not recorded properly, the user typed a URL into the browser address location, or a bookmark was used to directly jump into a website.

New York Times. This dataset, called NYTimes, consists of a sample of user transitions that occurred in New York Times (`nytimes.com`) and recorded using the Yahoo! browser toolbar from September 2011. The data consists of about 25,000 user trails, where each trail is identified by its anonymized cookie. The average length of the trails is around 9. A record for a cookie is similar to Yahoo, except that a and b are URLs in NYT. We map these URLs into one of 40 topics, where these topics were manually selected from the New York Times website and by looking at the URLs themselves. The topics will be the states of the Markov chain. Example topics are Science, Politics, Sports, World, etc. We used simple hand-crafted URL-based mapping rules to map the URLs to one of these 40 topics; by this process, more than 95% of the URLs were successfully mapped to a valid topic. The remaining were mapped into a generic `other` state. As in Yahoo, we also have an `outside` state to capture transitions from or to non-NYT sites.

Mousetracking. This dataset, called MouseTrack, consists of events such as mouse scroll, focus, and click, captured for a random sample of users visiting the Yahoo! SERP. This capturing was enabled by appropriately instrumenting the SERP and using the JavaScript mouseover and mouseout events on specific DOM elements in the SERP. The number of states in this dataset is 270 and includes states such as res::i (the ith search result), logo::logo (the Yahoo! logo), ads_horiz_bot (the horizontal ads at the bottom), etc. Note that these states are automatically extracted from the name of the corresponding DOM element in the raw HTML. The data was recorded for 10 days in August 2011 and consists of about 2.34M trails. The average length of each trail is around 7. The trail consists of elements of the form $\langle t_e, t_\ell, a\rangle$, where a is the state, and t_e is the time when the mouse entered the DOM element an t_ℓ is the time when the mouse left the DOM element (or the DOM element was clicked). We use the timestamps to construct the actual trail. Note that unlike the previous two data sets, this dataset captures the user behavior on a single page. Also, by the construction of SERP, a majority of the user movements have an orientation (top to bottom) and mostly self-avoiding (i.e., states are not typically revisited).

Eyetracking. This dataset, called EyeTrack, consists of eye gaze movements collected as part of a controlled experiment involving about 32 participants. In each treatment of the experiment, 8 random news articles were rendered on a 2×4 grid, where the positions are numbered row-major from left to right. Each participant was exposed to about 18 treatments and their task was to click on one article of their choice to read. All the participant's activities, in particular, their eye movements and gaze patterns, were recorded using a Tobii 1750 Eye Tracker (sampling rate 50Hz, 17" monitor, 1024×768 display resolution). We parsed the raw eye tracker data to obtain pauses and abrupt changes in the eye position. We associate the eye position with one of the 8 cells in the grid (thus, the number of states in this dataset is 8). The resulting data consists of 521 trails, where each trail consists of elements of the form similar to MouseTrack. The average length of the trails is around 68. This dataset is closer to MouseTrack in the sense that it is derived from user behavior on a single page, but is different in that there is no obvious top to bottom or left to right orientation . In fact, as we will see, the lack of orientation is heavily reflected in the behavior.

8. EXPERIMENTS

In this section we present the results of our various algorithms and measurements on the four data sets that we discussed in Section 7. First we present the results on the order of the chain representing browsing behavior across multiple pages and next we present the results on the order of the chain that captures the behavior within a single page. Then, we focus on variable order Markov chains and the effect of representing and compressing the state space. We then investigate the robustness of findings by subjecting the data to several natural constraints and modifications. Finally, we conclude with an application of our methods: to predict the next state visited by the user.

We implemented the basic algorithm for Markov chain order estimation. Recall that this algorithm simply involves maintaining various counters to count the number of transitions, for a given length of the history. This algorithm is naturally parallelizable in Map-Reduce, which is very important for studying large data sets such as MouseTrack and Yahoo. We present our results by computing the MLE matrix for various order chains and then computing the log-likelihood of the input for each of these orders. We use $k = 1, \ldots, 5$ for Yahoo (due to the size of the data) and $k = 1, \ldots, 8$ for the other three data sets. While reporting the performance, to convey the main idea, we report the relative improvement over the log-likelihood fit at $k = 1$, i.e., the usual Markov chain. This way, we can clearly see the value in using a chain of higher order to describe the observed trails.

8.1 Multi-page browsing behavior

Figure 1 shows the relative log-likelihood improvement for order k Markov chains over the $k = 1$ chain. As we can see, the curves are concave and appear to saturate at $k = 3$ for Yahoo and $k \approx 5$ for NYTimes. The relative improvements are around 11% for Yahoo and $\approx 13\%$ for NYTimes. This suggests that the browsing behavior across websites is definitely not Markovian but can be captured reasonably well by a not-too-high order Markov chain. Thus, the cross-site browsing behavior appears to have limited but non-trivial history.

If we examine the popular higher-order states in both the data sets we find that they are quite intuitive. For example, in Yahoo, the (Mail, Mail) or (`yahoo.com`, Registration, Mail), or (Mail,

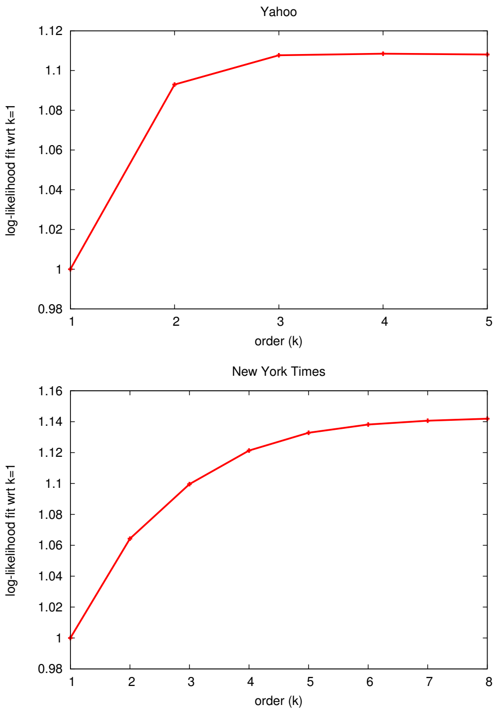

Figure 1: Log-likelihood fit for user visit patterns on Yahoo.

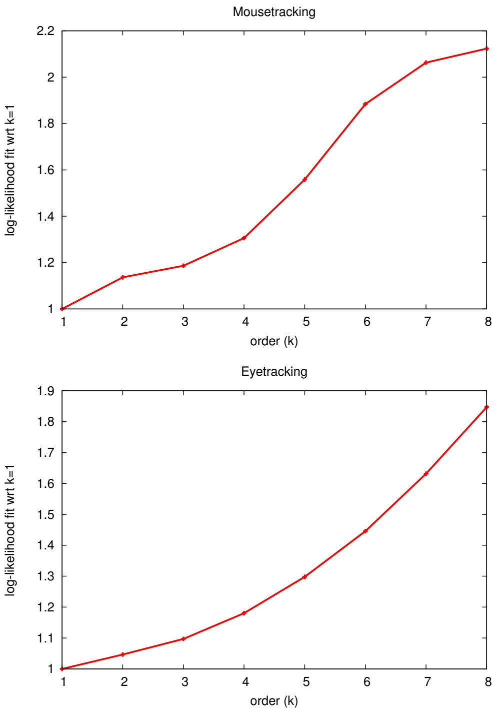

Figure 2: Log-likelihood fit for mouse movements in MouseTrack and eye movement in EyeTrack.

News) are popular higher-order states that determine the next transition. Likewise, for NYTimes, popular higher order states include (Business, Search), (NYregion, US, World), and (Opinion, Blog, Opinion).

8.2 Single-page browsing behavior

We then turn to the browsing behavior on a single page. To this end, we use the mousetracking data (MouseTrack) on SERP and the eyetracking data (EyeTrack) on news articles. Figure 2 shows the relative improvements in log-likelihood. As we see, unlike Section 8.1, the behavior is markedly opposite: the curves are convex for EyeTrack and convex up to order 6 for MouseTrack. This suggests that the single-page browsing behavior is not only highly non-Markovian but also cannot be represented by a low-order Markov chain. Thus, users clearly (perhaps subconsciously) remember their browsing pattern and the states they have visited.

Even though at a high level both EyeTrack and MouseTrack exhibit similar behavior, there are subtle differences. The plot for MouseTrack shows that the improvement is diminishing after $k \approx 7$. The reason is that users mostly visit the search results (which are the states) and it is reasonable that mouse movements on about 6 or 7 search results are probably sufficient to determine the user's next course of action. In contrast, the EyeTrack plot shows no signs of flattening. This is due to the inherent two-dimensional browsing task the users were subjected to. Middle states such as 2, 3, 5, 6 have to be revisited many times in order to move from the left side to the right side. So, a lot of history might be needed in order to determine the next course of action by the user.

8.3 Variable order models

In this section we study the performance of the algorithm for variable order estimation. We ran this algorithm on the four data sets and Figure 3 shows the results. To make comparison with the fixed order Markov chain easier, we interpret the x-axis as a fractional order, i.e., it is the sum of the history lengths of all states divided by the number of states. For completeness, we also include the range $k \in (0, 1]$ and show the performance of the fixed order Markov chain. Clearly, variable order Markov chains are very powerful and even with very limited total history, they can exceed the performance of fixed order chains, even with larger total history. This is only modestly true for EyeTrack, once again suggesting that the user behavior is more complicated in this case.

In course of building the variable order chains, it is illustrative to study which states benefit from having a lot of history. For EyeTrack, we see that the "middle" states 3, 2, 7, 6 benefit a lot from history. For MouseTrack, the search results (in increasing order from 1, $\ldots$, 10) benefit from history. For these two cases, the benefit is more polarized. In the optimal solution, these states demand a lot of history before other states get some amount of history. For Yahoo and NYTimes, the situation is quite different. The history gets spread evenly among the more popular states: e.g., Mail, News, Sports in Yahoo and World, US, Blog, Opinion in NYTimes. This once again suggests a marked behavioral difference between these two activities.

8.4 Transition table

In this section we study the transition table of a fixed order chain. First, we focus on the support sizes as a function of the order. Recall that an order k Markov chain can have $O(n^k)$-sized support; hence, it is useful to measure the support size of the kth order chain as a fraction of this maximum. Figure 4 shows the relative sizes. Clearly, MouseTrack is quite efficient in terms of support whereas EyeTrack requires relatively more values in the support. The support sizes for Yahoo and NYTimes are comparable and lie somewhere in between.

Next, we study the effect of pruning some of the entries in the transition table. This pruning is done at the counting stage, before

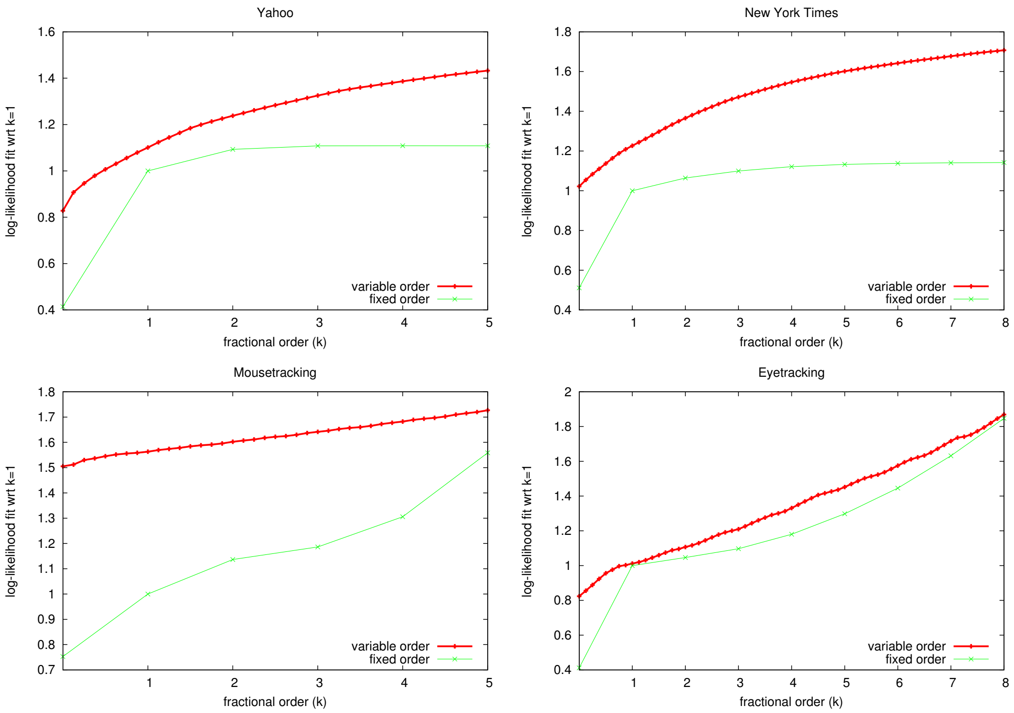

Figure 3: Best variable order chains obtained by the algorithm for various data sets.

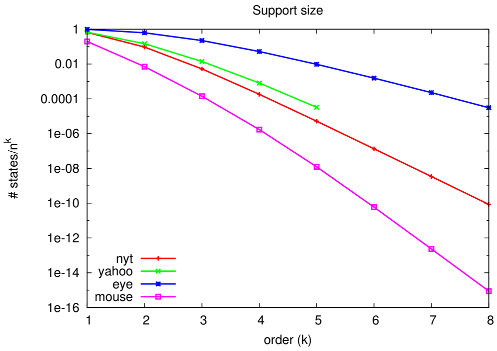

Figure 4: Support size of the higher-order chain relative to the maximum support size.

the transition matrix is normalized. For brevity, we only show the results for NYTimes; the other are similar. The top panel of Figure 5 shows the performance hit in log-likelihood when entries below a certain count are removed from the table (e.g., the curve for the legend > 4 denotes normalizing the matrix after removing all counts of at most 4). The bottom panel of Figure 5 shows the declining support size after pruning. It is clear from the figures that even aggressive pruning can result in significant space savings while not compromising adversely on the quality of the representation.

8.5 Robustness analysis

In this section we perform various analysis to study how robust are our findings.

Train-test split. First, we focus on computing the MLE estimator on a dataset that is different from the dataset on which the log-likelihood evaluation is done. We choose the two large data sets Yahoo and MouseTrack for this purpose. We split the data into two equal-sized partitions, train and compute the MLE on one partition, and evaluate it on the other partition. The results are shown in Figure 6. There is not much change in terms of the relative log-

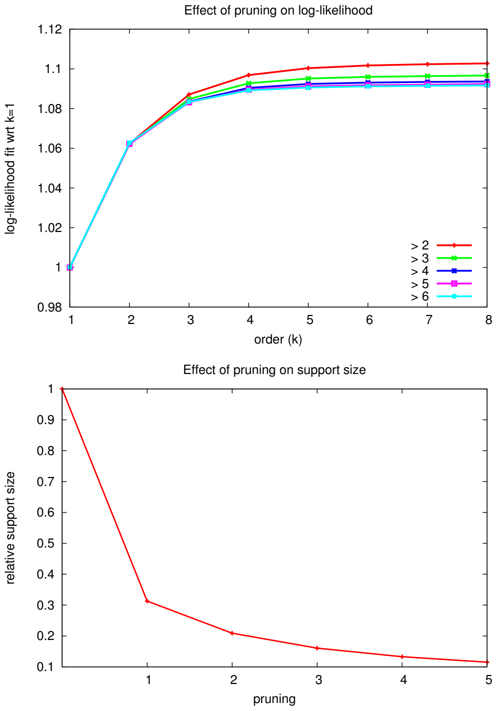

Figure 5: Effect of pruning on NYTimes.

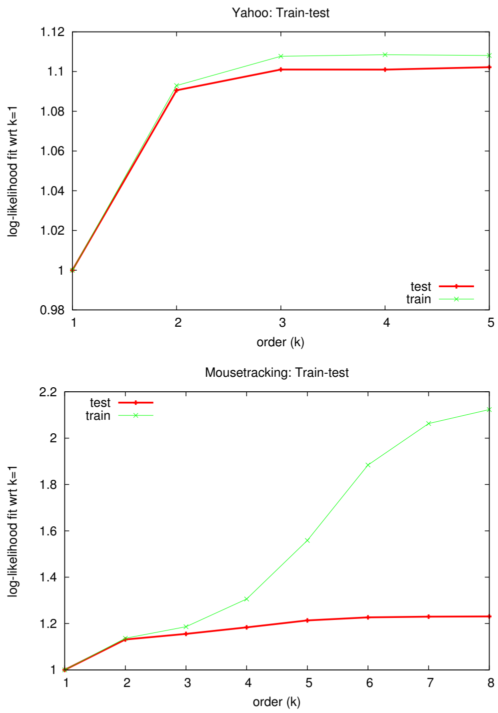

Figure 6: Train-test analysis for Yahoo and MouseTrack.

likelihood improvements over $k = 1$ between the original results and the new results for Yahoo (and NYTimes) but for MouseTrack (and EyeTrack), there was a marked difference for higher order states. This suggests that page browsing patterns, even at an aggregate level, are hard to learn and utilize. Exploring this discrepancy further is an interesting direction for future research.

Removing self-loops. Next, we focus on removing self-loops in the data and see how it would affect the findings. Self-loops are natural in all the data sets and in some applications, it is important to consider the process without self-loops since other stochastic models can be used to capture the dwell time on a particular website. Figure 7 shows the results. The improvements are almost halved for NYTimes and nearly unchanged for MouseTrack. The former is intuitive since users might browse similar categories repeatedly across different web pages. The latter happens since single-page browsing is less likely to have too many self-loops and hence the impact of removing them is minimal.

Removing short trails. Finally, we study the impact of removing trails that are too short. Note that by including trails that are very short, we are actually downplaying the performance of higher-order chains. Hence, if we remove them, we should see an improvement in their performance. Figure 8 shows the results for NYTimes and MouseTrack, after removing trails of length at most 5. The effect of having longer history is quite dramatic suggesting that longer trails can actually benefit a lot more from them. From a different point of view, we also study the impact of sessionizing: breaking up long trails into smaller trails if the consecutive time interval is more than 30 minutes. Figure 9 shows the effect of such a sessionization on Yahoo. The effect, as seen, is minimal: trails spanning more than a session would not have benefited from history in the first place and hence this is to be expected.

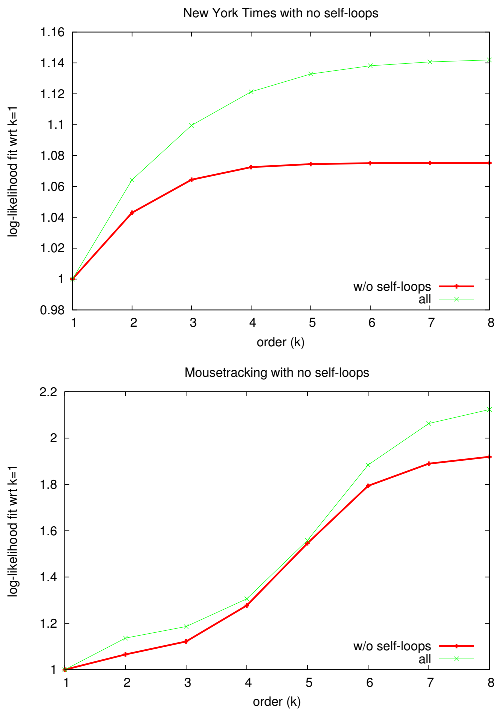

Figure 7: Fixed order chains with self-loops removed for NYTimes and MouseTrack.

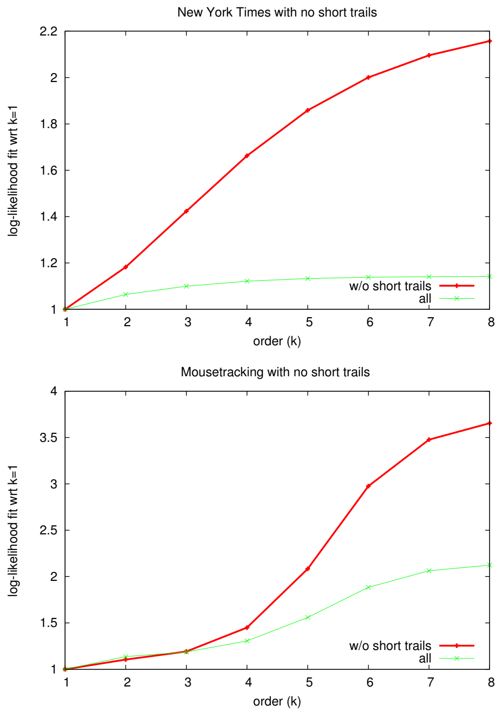

Figure 8: Fixed order chains with trails of length at most 5 removed for NYTimes and MouseTrack.

8.6 An application: Prediction

In this section we study a simple application of our findings so far: how much can the next state of the user be predicted with

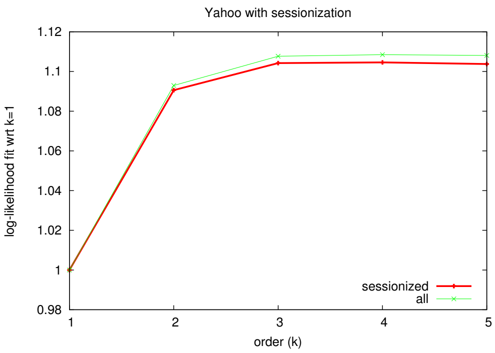

Figure 9: Fixed order chains with sessionized trails for Yahoo.

a higher-order Markov chain. To this end, we use the computed MLE matrix and the fact we proved about the stationary of the MLE Markov chain in order to compute the probability of predicting it accurately. For ease of interpretation, we present the results relative to the prediction probability for $k = 1$. Figure 10 shows the results. The improvements are significant (40% with order 2-3) for MouseTrack and very minimal for EyeTrack; once again, the two-dimensional browsing aspect of EyeTrack makes it hard to predict well. For NYTimes, we get around 10% improvement for $k = 3$, but for Yahoo, the behavior seems more intricate. This remains the subject of future investigation.

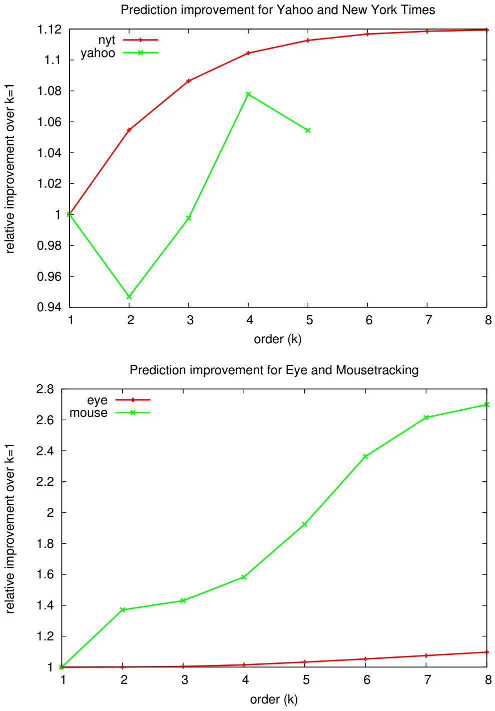

Figure 10: Improvements over prediction with order 1 Markov chain.

Acknowledgments

We thank Fernando Diaz, Sergiy Matusevych, and Vidhya Navalpakkam for providing us some of the data.

9. REFERENCES

[1] N. Archak, V. Mirrokni, and S. Muthukrishnan. Mining advertiser-specific user behavior using adfactors. In *19th WWW*, 2010.

[2] P. Boldi, F. Bonchi, C. Castillo, D. Donato, A. Gionis, and S. Vigna. The query-flow graph: Model and applications. In *17th CIKM*, 2008.

[3] J. Borges and M. Levene. Evaluating variable-length Markov chain models for analysis of user web navigation sessions. *IEEE TKDE*, 2007.

[4] P. Buhlmann and A. Wyner. Variable length Markov chains. *Annals of Statistics*, 1999.

[5] H. Cao, D. Jiang, J. Pei, E. Chen, and H. Li. Towards context-aware search by learning a very large variable length hidden Markov model from search logs. In *18th WWW*, 2009.

[6] N. Craswell and M. Szummer. Random walks on the click graph. In *30th SIGIR*, 2007.

[7] I. Csiszár and P. Shields. The consistency of the BIC Markov order estimator. *Annals of Statistics*, 2000.

[8] D. Dalevi, D. Dubhashi, and M. Hermansson. A new order estimator for fixed and variable length Markov models with applications to DNA sequence similarity. *Statistical Applications in Genetics and Molecular Biology*, 2006.

[9] B. Davison. Learning web request patterns. *Web Dynamics*, 2004.

[10] M. Deshpande and G. Karypis. Selective Markov models for predicting web page accesses. *ACM TOIT*, 2004.

[11] I. Holyer. The NP-completeness of some edge-partition problems. *SICOMP*, 1981.

[12] J. Kemeny and J. Snell. *Finite Markov Chains*. van Nostrand, 1960.

[13] R. Lempel and S. Moran. SALSA: The stochastic approach for link-structure analysis. *ACM TOIS*, 2001.

[14] Z. Li and J. Tian. Testing the suitability of Markov chains as web usage models. In *COMPSAC 2003*, 2003.

[15] L. Page, S. Brin, R. Motwani, and T. Winograd. The PageRank citation ranking: Bringing order to the web. Technical report, Stanford InfoLab, 1999.

[16] Y. Peres and P. Shields. Two new Markov order estimators. *Arxiv Preprint Math/0506080*, 2005.

[17] P. Pirolli and J. Pitkow. Distributions of surfers' paths through the World Wide Web: Empirical characterizations. *WWW*, 1999.

[18] J. Rissanen. A universal data compression system. *IEEE Trans. on Inf. Theory*, 1983.

[19] D. Ron, Y. Singer, and N. Tishby. The power of amnesia: Learning probabilistic automata with variable memory length. *Machine Learning*, 1996.

[20] R. Sarukkai. Link prediction and path analysis using Markov chains. *Computer Networks*, 2000.

[21] R. Sen and M. Hansen. Predicting web users' next access based on log data. *JCGS*, 2003.

[22] I. Zukerman, D. Albrecht, and A. Nicholson. Predicting users' requests on the WWW. In *7th UM*, 1999.