ABSTRACT
Information about consumer’s web navigation trails are increasingly being collected, analyzed, and used to target them with advertisements. This information can be quite personal, indicating individuals’ likes and dislikes, as well as current and long term needs. While the ability to effectively target advertisements helps keep many sites and services on the web available freely, this practice has also raised privacy concerns. These concerns concern multiple factors: lack of transparency by the data aggregators and lack of control by the consumers. One viable approach that individuals can take to regain control is obfuscation, whereby real user requests are masked via the injection of noisy requests. In this paper, we describe a theoretical model and design for a web browser extension that relies on a trusted third party to generate fake HTTP requests (dummies). The dummy requests are generated as \( k \) different users’ profiles surfing in parallel with the actual user. The value of \( k \) can be adjusted by the user to achieve the level of obfuscation they are comfortable with.

Categories and Subject Descriptors
K.4.1 [Computers and Society]: Public Policy Issues — Privacy

General Terms
Documentation, Security, Human Factors.

Keywords
Web Tracking, Private Web Navigation, Obfuscation, Behavioral Targeting.

1. BACKGROUND
There is a growing ability by on-line advertisers and web data aggregators to collect and store information about Internet users as they navigate the Internet [1], [2]. The information collected is increasingly personal covering likes and dislikes, as well as current and long term needs. What is currently known is that this information is used for targeting advertisements and to enhance the quality of service. In the past, web companies were only able to track people’s actions on their own sites, but this has changed. Giant Internet companies are now able to track users’ movements across almost many popular websites, thus creating vast amounts of data about users [1]. One of the potentially lucrative uses of this information is “behavioral targeting”, it is the process of determining people’s likes based on where they go on the Internet, then presenting them with advertisements that are most likely to impact them. Behavioral targeting, (BT), is formally defined in [3] as follows: “Behavioral targeting is a method of advertising seeking out consumers based on where they go and what they do as indicators of what their interests are most likely to be. BT is mostly applied in online ad targeting, where consumers’ behaviour on a relevant website is tracked to determine their interests, then ads are served to that person relevant to their interests”. Hence BT aims at reducing advertisement wastage by targeting ads to interested consumers. The BT industry is still in its early stages; it is not well defined or standardized [4]. However, recent investments in BT by leading online companies (such as Google, AOL, and Yahoo!) demonstrate the growing interest in BT and a significant shift to online advertising has been forecast.

The practice of storing and analyzing users’ web activities has opened privacy debates on the definition of privacy, and whether it is a right, and raising difficult questions about who owns this information, and who decides what should be done with it. On the practical side, we also see a growing concern on the possible harmful effects of data collection fueled by the privacy breaches and mistakes done by these giant aggregators [5].

In this paper, we describe the theoretical model and design of a web browser extension that relies on a trusted third party to generate fake HTTP requests (dummies). The dummy requests are generated as \( k \) different users’ profiles surfing in parallel with the actual user. The dummy profiles should be “real enough” to confuse the aggregator observing the user’s activities. In other words, each fake profile should be consistent and mimic a real user’s online behavior.

2. PROBLEM DEFINITION
Traditional BT tracks online users using some form of categorization [4]: We will refer to the entity, or set of entities, that perform this data collection and targeting as an “aggregator”.

An aggregator identifies a set of categories that is relevant to their campaigns; users are associated with a given category if they demonstrate interest in it, by visiting websites, clicking on ads, or typing keywords related to this category. Examples of categories include home buyers and luxury car lovers. These categories will assist the marketer in directing their ad campaign to the right audience. Once classified into a category, a user remains associated with it for a period of time that is a function of the category, for example a homebuyer remains on the market for an average of four months. This period of time is referred to as the
“period of Desire”. The categories can be classified into two
types: situational and long term. Users are interested in situational
categories for a limited time such as home buyers, while long term
categories reflect permanent or long term interest in a subject such
as hobbies, or chronic diseases. For example, BlueKai, a leader in
online data aggregation, sorts consumers into 30,000 pre-defined
categories such as “light spenders and safety-net seniors” [6].

An individual \( u \) who is surfing the web creates a profile \( P_u \)
consisting of all the webpages accessed along with the time spent
per page. An aggregator observing the user’s activities will have
access to all or parts of the profile, the aim of the aggregator is to
figure out the users’ interests in order to classify them into the
predefined categories.

In what follows, we assume the model of an aggregator that tracks
users online and stores all of their Internet activities in a database
for a fixed period of time \( T \). The aggregator uses this information
associated with users with a set of predefined categories, users are
associated with a category for a period of time that depends on the
category itself. For example, a user \( u \) demonstrates interest in the
chronic disease “diabetes type 1” by visiting websites \( X, Y \) and
\( Z \) affiliated with the disease. The aggregator then stores that
user’s profile \( X, Y, Z \), for a period of time \( T \), and associates
the user with the category “type 1 diabetes” for a period of time
that is a function of the category (since type 1 diabetes is a
chronic disease, the user might be associated with the category for
good).

Our objective is to confuse the aggregator. This means making it
difficult for the aggregator to allocate the correct category to an
individual user based on their surfing habits (i.e., the probability
of such an allocation would be low). Successful obfuscation
would reduce the cost effectiveness of tracking individual users
and may encourage aggregators to opt-in (or use some other
model to compensate) individual consumers in exchange for
removing the obfuscation.

In order to confuse the aggregator, we would like to generate
dummy webpage requests from the user’s machine while the user
is web surfing. The idea is to generate \( k \) fake surfing profiles
\( D_1, \ldots, D_k \) that would be added to the real profile \( P_u \). The set
consisting of the real user’s profile and the dummy profiles ( \( D_1, \ldots, D_k \) and \( P_u \) ) will be labeled as \( \varphi_u \). To identify the
correct profile \( P_u \), the aggregator will have to tell the real
webpage accesses apart from the fake ones, or to correctly
reconstruct the profiles \( D_1, \ldots, D_k \) and \( P_u \), then identify the
correct profile \( P_u \). Our aim is to make the aggregator’s task
difficult by making the dummy profiles look like real ones. To
achieve that, the dummy profiles will be generated using a
combination of real users’ profiles:

A group of participating users voluntarily and regularly submit
their profiles to a trusted entity referred to as the “market”. These
users will be referred to as market suppliers (or simply suppliers).
This pool of information will provide the market with the needed
information to analyze and predict suppliers’ surfing habits and
thus to create stochastic models, one for each supplier, that
describe their surfing habits, and that predicts their future
interests. The stochastic models are updated regularly to reflect
the supplier’s changing/new interests.

Now, a person \( u \) wishing to obfuscate his/her Internet
navigation, registers with the market as a user. Consequently, the
market associates \( k \) stochastic processes (belonging to \( k \)
different suppliers) \( S_1, \ldots, S_k \) to \( u \). These processes will be
used to generate the dummy profiles \( D_1, \ldots, D_k \). The different
stochastic models \( S_1, \ldots, S_k \) should be chosen such that they
have some degree of variability among themselves and the actual
user’s profile, \( P_u \). A measure for the degree of variability will be
presented later.

3. PREVIOUS WORK

Previous obfuscation-based methods focused on private web
search [7]–[13]. The basic methodology consisted of generating
dummy search queries. The dummy queries might not reflect the
users’ interests. As a result they will introduce noise to the user’s
profile. These methods do not require a 3rd party to cooperate and
they operate on a user’s machine only.

A popular tool for web search obfuscation is TrackMeNot (TMN).
TMN tries to obfuscate users’ queries by generating dummy
search queries using a Firefox browser [12]. When installed, TMN
has an embedded list of query terms chosen from publicly
available lists of popular search queries. This list is then
continuously modified with terms from the pages surfed by the
user. TMN observes its user’s online behavior and mimics his/her
search habits (such as timing and search engines used). However,
TMN does not offer any theoretical guarantee of successful
obfuscation of search terms, in other words, the authors did not
define a way to measure whether the obfuscation was successful
or not. Therefore, it is not known how well TMN actually works.

The problem with these types of tools is that they are designed to
obfuscate web search terms and thus to prevent web search
service providers from accurate inference on users’ search
profiles. In fact, these tools obfuscate search terms and never
follow up with links from the search pages. This means that a
non-search website visited, or an ad clicked on, would always
originate from the user. Since search engines are usually used as
means to the get to a webpage, most of the interests of a surfer
would be known from the non-search websites visited. Since web
aggregators are now able to track users’ movements across almost
all popular websites, a different approach should be taken, one
that takes into consideration the whole profile of the user.

Other non-obfuscation based approaches have also been taken to
address this problem. These approaches include the open source
software Tor [14]. Tor encrypts users’ actions then sends them
through a distributed network of proxies that are run by
volunteers. The path between the proxies is random, and being
encrypted means that no proxy can learn the content of the
message and associate it with the correct sender [14]. The biggest
concern with using Tor regularly is usability, since Javascript and
Flash are disabled when using Tor. Moreover, in [15] the authors
launched an attack on the Tor network and were able to identify
the IP address of a Tor user and associate it with a secure
application.

There also exists opt out measures that a user can benefit from,
these are: disabling 3rd party cookies, and Javascript, and refusing
4. DEFINITIONS AND MOTIVATION

4.1 Preliminary Definitions

Definition 1. A trail is an ordered set of websites along with some timing information: \((w_1, w_2, \ldots, w_n)(t_{1,2}, t_{2,3}, \ldots, t_{n-1,n})\), where each \(w_j\) represents a webpage, and the following property holds: for any website \(w_i, i > 1\), there exists another website \(w_j\) for some \(j < i\), such that \(w_i\) is accessed from within \(w_j\). \(t_{j,i}\) refers to the time spent from accessing \(w_j\) until accessing \(w_i\). In other words, a trail is a tree like structure of websites along with the time spent in each website, such that each node is accessed from the parent node. The websites, \(w_j\), are assumed to be URLs with no headers associated with them.

Note that, because of the property above, a search term can only be the first term of a trail. However, the first term in a trail need not be a search term, it can be a visit to any website.

In what follows, we assume that each trail belongs to exactly one category. The assumption was designed to simplify the presentation, and it does not limit the generalizability of the model (in fact, a straightforward way to generalize the model is by assuming that each category represents several interests. In other words a category would represent a set of inter-related interests that are implied by the same trail. For example a category could represent: interested in natural remedies and healthy food and yoga).

Definition 2. A User Profile: is a set of trails originating from the same user.

Definition 3. A user session (or session profile): is the set of trails executed by the user during an online session.

In what follows, we assume that the aggregator and the market both have the same set of predefined categories \(C = c_1, \ldots, c_h\).

4.2 The Basic Model

4.2.1 Overview

The market associates \(k\) stochastic processes (belonging to \(k\) different suppliers) \(S_1, \ldots, S_k\) to each user \(u\). Each stochastic process \(S_i\) describes/predicts the navigation of supplier \(i\) between the different categories: \(c_1, \ldots, c_h\). Hence, the nodes of the process are the different categories \(c_1, \ldots, c_h\), and the edges describe a possible move between two categories along with the probability of the move. When user \(u\) starts surfing the net, the stochastic processes \(S_1, \ldots, S_k\) will be used to predict the category \(c_{i_1}^{1}, \ldots, c_{i_k}^{k}\) that the suppliers \(1, \ldots, k\) will be interested in. In general, at appropriate time \(t\), \(S_i\) is used to determine the next category \(c_{i_j}^{j}\) that supplier \(i\) will most likely be interested in next, given its history. Now, given the next category \(c_{i_j}^{j}\) that the dummy profile \(D_i\) should pursue, the market chooses a trail that belongs to category \(c_{i_j}^{j}\) and sends it to the user for execution (following the timing indicated in the trail).

Note that the trails of the dummy profile \(D_i\) may not belong to the same supplier; however the long term and short term interests in the different categories simulate that of a real user. The market may wish to always assign trails to the dummy profile \(D_i\) that belong to the supplier \(i\), however that depends on the availability of trails from supplier \(i\). For the time being, we assume that the
set of trails belonging to the same dummy profile share common surfing habits [17], [18].

4.2.2 Motivation
Without relying on a trusted third party, generating fake user queries has been shown to be ineffective [19]. The main problem is that the online behavior of users follows certain patterns. Thus an adversary can employ advanced data mining techniques to construct the real user trails and discard the dummy requests [20]. A good solution would be to generate dummy trails that mimic real users’ trails in every user’s online session. However, even though this will reduce the probability of knowing the real requests from the dummy ones in one online session, the long term surfing pattern can be studied to filter out dummy trails from real ones (for example an adversary can compare the current user’s profile with the profile collected the day before). Thus it is important to generate dummy trails that not only mimic a real user in one online session, but also one that follows consistent long term surfing pattern. For this reason, our tool associates, for every user $u$, $k$ fixed stochastic processes that belong to $k$ distinct market suppliers.

On the other hand, if the generated dummy profiles are very close to the real one (in terms of the categories represented in these profiles), then the interests of the user might be exposed making it easier to categorize them. Thus our design of the model takes into consideration a degree of variability among the stochastic processes assigned to a user.

Moreover, if the same dummy trail is assigned to more than one user, then an aggregator comparing different users’ profiles might be able to figure out the dummy profile. Hence, each user will be assigned distinct stochastic processes, and no dummy trail should be stored at the aggregator’s end for more than one user.

Now, for every user wanting to join the market, the market needs to have enough resources (unassigned suppliers, enough usable trails) before accepting the request. However, in what follows, we assume that the resources of the market are infinite.

Table 1 shows an example of a user’s obfuscated online session with $k=2$ (the session consists of three trails). The probability of an adversary disclosing the true trail using only the current internet session (and thus knowing the user’s current interests) is

$$\frac{1}{k+1} = 1/3.$$

<table>
<thead>
<tr>
<th>Time slot</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>User</td>
<td>Weather</td>
<td>Weather</td>
<td>Weather</td>
</tr>
<tr>
<td></td>
<td>Network</td>
<td>Network</td>
<td>Network</td>
</tr>
<tr>
<td></td>
<td>Ottawa</td>
<td>Ottawa</td>
<td>Montreal</td>
</tr>
<tr>
<td>Dummy 1</td>
<td>CBC News</td>
<td>Local News</td>
<td>Corruption</td>
</tr>
<tr>
<td></td>
<td>for Montreal</td>
<td></td>
<td>Commission</td>
</tr>
<tr>
<td>Dummy 2</td>
<td>YouTube</td>
<td>Annoying</td>
<td>Justin Beiber</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Orange Video</td>
<td>Video</td>
</tr>
</tbody>
</table>

Table 1: An example of a user’s obfuscated online session with $k=2$.

5. THE MODEL DETAILS
In what follows, we present the different elements of the model in more detail.

5.1 The Aggregator
The aggregator is assumed to be a curious entity with access to all users’ Internet queries. The goal of the aggregator is to infer important private information about the users from their Internet profiles. The information that is of interest to the aggregator pertains to the classification of the users in a set of predefined categories $C$. For this reason, the aggregator records all users’ queries, and tries to classify the users into different categories using the information collected.

So the aggregator operates as follows:

- The aggregator observes users’ activities (like websites visited, ads clicked, words searched) and classifies users into different categories based on their online behavior: A visit to a website, puts the user in the category that this site belongs to.
- The aggregator attributes a category $c \in C$ to a user for a period of time (Desire period) that is a function of the category: $T_c$.
- In addition to storing affiliations of users with categories, the aggregator stores users’ trails for a fixed period of time $T$. In other word, each trail remains stored in the aggregator’s database for $T$ time units, after which it is discarded.

5.2 The Market
A Market of $m$ users and $s$ suppliers exist:

1. The suppliers supply the market with their trails continuously. The market stores the suppliers’ trails for $T'$ time units (where $T' > T$) or until they expire (when one or more URL in the trail ceases to exist or is changed). The Market benefits from having an unlimited database of trails, hence The bigger $T'$ the better (limited by space availability).

   For every supplier affiliated with the market, the market creates a stochastic process that describes his/her navigation between the different categories. This process can be used to predict the next category $c_t$ that the supplier will be interested in. Hence, the nodes of the process are the different categories, and the edges describe a possible move between two categories. The market uses the navigation history of its suppliers (the set of trails in the market associated with the supplier) to create these processes. The stochastic process is continuously updated with the suppliers’ new surfing data.

2. An interested user $u$ registers with the market in order to obfuscate his/her Internet activities. The Market grants the user’s request if it has $k$ distinct, unassigned and varied stochastic processes (the notion of variability will be defined later).

   Once a user is added, the market will obfuscate the user’s profiles by supplying the user with real trails submitted by the suppliers. In the following subsections, we explain the process in more details.
5.2.1 Identifying the next category for a dummy profile

Every time a user \( u \) is added to the market, the market associates \( k \) stochastic processes (belonging to \( k \) different suppliers \( u_1, \ldots, u_k \)) \( S_1, \ldots, S_k \) to \( u \). These processes will be associated with \( u \) until their originators (the supplier) drop out of the market, at which time a new supplier/stochastic process is selected. These processes are used to generate \( k \) dummy profile sessions \( D_1, \ldots, D_k \) while the user is surfing. The process goes as follows: When the user \( u \) starts an online session, its profile session \( P_u \) is empty, the profile starts filling up as the user is surfing. At the same time, the market uses the user’s associated stochastic processes to create the dummy profile sessions \( D_1, \ldots, D_k \) in parallel:

At the beginning of the user’s session, the \( k \) stochastic processes are used to predict the first category for each of the dummy profiles (say \( c_1, \ldots, c_k \) respectively), then the market selects trails \( T_1, \ldots, T_k \) that belong to \( c_1, \ldots, c_k \) respectively. Each of these trails is then executed on the user’s computer (according to the timing provided in each trail). If a trail \( T_j \) is fully executed after \( t \) time units, and if the profile of \( u \) looks like: \( P_u, D_1, \ldots, D_k \), then \( S_j \) is used to determine the next category \( c_j \) that \( u \) will most likely be interested in next, given its history \( D_j \), and a new trail is chosen accordingly. Note that because trails tend to be unique, a dummy trail cannot be used again (by two different users or by the same user) until \( \tau \) time units passed (the time needed for a trail to clear from the aggregators database).

The choice of the right trails from a given category is explained in more details in the next subsection.

5.2.2 Choosing the next trail for a dummy profile

Once the next category \( c_j \) in a dummy profile is determined, we need to choose a trail \( T \) from the market that belongs to category \( c_j \) and add it to \( \mathcal{Q}_u \). However this is not a straightforward task due to several reasons:

- Some URLs are unique as they embed a unique number in them:
  - One example is when a supplier \( u \), visiting website \( W_1 \) clicks on some ad which redirects him to a website \( W_2 \), and \( W_2 \) has a URL with a unique identifier, this identifier embeds information about \( W_1 \) in it (this is usually helpful for performing statistics on how ads were accessed). This unique identifier cannot normally appear in another profile for a different person, say \( u' \), unless \( W_2 \) was shared between \( u \) and \( u' \).
  - Another more dramatic example is when the unique number represents information that uniquely identifies the originator of the URL (i.e. supplier \( u \)).

To overcome the above cases, we assume that the market has a URL Scrubbing tool. This tool removes all additional and inessential information from the URL:

- Trails are sequences of websites with timing spent in every site, and as such they can be unique. So if a trail appears in several users’ profiles, its uniqueness will suggest that the first user who issued the trail is the correct user, and others most likely have it as a dummy trail. However, as suggested above, because the market does not allow two profiles to have the same trail within \( \tau \) time interval, the aggregator will not be able to use the uniqueness of the trails to rule out the dummy ones.

- Search terms, at the beginning of some trails, say \( W_1 \) in \( W_1 \rightarrow W_2 \rightarrow \ldots \rightarrow W_h \), can contain personally identifying information (for example searching for names or addresses), to overcome that, the market removes the search terms at the beginning of every trail, then the market looks for a keyword that is common in \( W_2 \) and that would have a link to \( W_h \) in the first page of the search results, if found, the market uses the same search engine to initiate a new URL \( W_1' \). Otherwise, the search is considered personal and the trail is discarded.

- The market needs to discard all trails with websites that require log-ins.

- Some websites require personal information, such as, Mapquest, websites that sell products, charity websites and so on. The trails embedding such websites need to be discarded from the Market.

Once the trails are cleaned, they are distributed into their corresponding categories. Hence, the Market will have \( h \) sets of trails, each set belonging to a different category. Then a trail \( T \) from category \( c_j \) is provided to \( u \).

It is important to note that the market needs to analyze the surfing habits of its suppliers (in terms of, for example, timing and browser used) and divide these suppliers into different sets depending on their surfing habits (suppliers with similar surfing habits would belong to the same set). Thus, when assigning a trail \( T \) to a dummy profile \( D_j \), the Market can stick to trails originating from suppliers that share \( i \)’s surfing habits (i.e. trails from suppliers that belong to the same set as \( i \)). We will not discuss this division in the current paper, however we will assume that such a division exists and is implemented by the Market.

After the trail \( T \in c_j \) is added to a user’s profile \( P_u \), the Market labels it as inactive for \( \tau \) time units (the time the trail will spend in the aggregator database), and consequently cannot be used in another dummy profile until it is reactivated.
Given a user $\mathbf{u}$, the associated suppliers $\mathbf{u}_1, \ldots, \mathbf{u}_k$ need to have some degree of variety among themselves and with $\mathbf{u}$. The purpose is to associate more categories per $\mathbf{u}_i$. To achieve that, we ask that the set $S_1, \ldots, S_k, S_u$ (where $S_u$ is the stochastic process associated with user $\mathbf{u}$) be $(1, J)$-varied. The definition is presented next, but first we assume that the categories are divided into two distinct sets: situational and long term (as described earlier).

**Definition 5.** Given $k$ stochastic models $S_1, \ldots, S_k$, let $C_1 \ldots C_k$ be the set of long-term categories represented in the states of $S_1 \ldots S_k$ respectively. Then $S_1, \ldots, S_k$ are said to be $(1, J)$-varied if the set: $\Omega = \{C_1 \cup C_2 \cup \ldots \cup C_k \mid \Omega \geq 1 \}$ and the categories in $\Omega$ belong to $J$ different processes $S_{i_1}, \ldots, S_{i_j}$.

Stochastic processes as well as other processes and tools that predict users’ behaviors have been used in the areas of context prediction and web prefetching. In the next section, we summarize the techniques used in both areas, and suggest an algorithm that uses a high order Markov Model: Prediction by Partial Matching as a good solution in our context.

6. **STOCHASTIC PROCESSES**

6.1 **Prediction and Prefetching**

Context prediction is the task of predicting the next user action based on previous actions. For example in mobile advertisement, predicting the future location of a user will enhance the relevance of the advertisement. We are interested in algorithms that allow non-numerical contexts (website categories) and that do not rely on time-dependant typical user patterns (for example, people sleep at night, and return home in the evening). The most prominent algorithms that satisfy these requirements use Markov models to predict the user’s next action [21], [22]. These algorithms differ in the order of the Markov model used (some use variable order models).

On the other hand, web prefetching is a method to reduce the wait time for webpage access. It is the art of predicting the next possible page to be requested in the near future. There exist three types of prefetching depending on where the technique is implemented: server, proxy or client side. The server-based prefetching predicts future accesses to a given server from all users, the proxy-based prefetching predicts future accesses for a group of users across all servers, and client-based prefetching predicts future accesses of one user across all web servers.

There exists a large variety of web prediction algorithms, these can be classified into two categories, content prefetching, which are predictions made from analyzing the content of the current pages, and history-based which predicts future accesses from past ones (history).

In this problem, we are interested in history-based algorithms that are implemented on the user’s side. The reason is that we want to model a user’s future requests, not as web accesses but as contexts, in other words, we want to predict the next category that the user will be interested in pursuing, given the user’s previous context (category navigation history). The main algorithms that fit this description can be classified into two main categories: algorithms that are based on Markov Models and algorithms that use data mining techniques. The Markov based algorithms are effective in prefetching, they operate by matching the user’s current access sequence with the user’s historical access sequences [23], low order Markov models suffer from non-accuracy, while higher Markov models have good accuracy but suffer from state space complexity. Some authors have proposed “prediction by partial matching” (PPM) algorithm [24], which is one of the most successful algorithms in the web prefetching domain as well as in context prediction [22], [23], [24], [25], [26]. It uses a high-order Markov model that has the form of a tree. Each branch of the tree represents a (partial) sequence of the user’s requests. The main drawback with PPM is that as more web pages are requested by the user, the graph grows linearly in size, refer to [27] for a detailed overview of the algorithm. In fact if $k$ is the maximum allowable branch height, and if $C$ is the set of categories, then the worst case complexity of the PPM model is $O(|C|^k)$.

Recent trends in Web Prefetching combine the use of high order (or variable order) Markov models with data mining techniques such as clustering and association rules. The objective is to benefit from the prediction accuracy of Markov models while reducing its state space complexity [28].

Assuming that $C$ and $k$ are bounded, the PPM model can be used to predict the user’s next category. In the next subsection, we present the model in more details.

6.2 **PPM**

A PPM is a set of Markov trees (a special kind of graph) whose nodes are categories and whose branches are ordered sequences of categories. The tree is used to record the categories visited by the user, in the order visited by the user. Each node has a counter that indicates the number of times the sequence of categories leading up to the node has been executed. Figure 2 shows how to build a PPM. If a user starts his session by requesting a trail from category $C_1$, then a root node with category $C_1$ is created. If the next category visited by the user is $C_2$, then a node with $C_2$ is created and added to the PPM as a child node of $C_1$, and another node $C_2$ is created as a new root node. If category $C_3$ is subsequently visited, then a node with $C_3$ is created and added to the branch $C_1 \rightarrow C_2$, and another node is created and added as a child node of $C_2$, and another one is created as a new root node. If in another session the user requests $C_1$ followed by $C_4$, then a new node with category $C_4$ is created as a new child of the root node $C_1$, and another node representing $C_4$ is created as a root node.
To avoid a state space explosion, a height limit $m$ is imposed on the trees. In other words, a very long sequence of categories is recorded as a set of subsequences. Given the category sequence of a user session, PPM is used to match each suffix of the last $m$ categories with the branches of the tree, starting with the longest suffix (last $m$ categories, followed by last $m-1$ categories...). Once a match is found, the branch is used to predict the next category. For more information, the reader is referred to [24], [27].

7. LIMITATIONS

The analysis presented here does not address the impact of the user-market communication on the overall speed and performance of the web requests. On low bandwidth connections our approach may impact the user’s web navigation experience.

It is plausible that web site owners and service providers may react to a scheme such as ours by removing or restricting free content. For example, if obfuscation can successfully blur user categories then another model to finance web content and services may be needed. Such actions may be targeted specifically at only those users employing obfuscation schemes. Therefore, a desirable feature of an obfuscation scheme is to hide that one is being used from the aggregators.

The market is assumed to have unlimited resources (for example, in terms of unlimited trails, and unlimited space for creating stochastic processes) and that it knows the categories held by the aggregator. In practice, this means that the market would need the same capability as the aggregators themselves. Moreover, the market is assumed to have a URL scrubber.

Another consideration regarding the market is who would finance its construction and operation. Users may be the most incentivized to do so, or it can be a differentiating feature for web browser vendors and they would finance it. This is an area which requires additional research.

Another assumption we make is that trails are held by the aggregator for a limited and fixed time $\tau$. Although that is plausible, the exact time is not necessarily known by the market. Although regulations on the destruction of web surfing logs may set an upper limit.

8. ACKNOWLEDGMENTS

Our sincere thanks to Stuart Shapiro for his valuable insights on initial discussions of this work.

9. REFERENCES


Figure 2: An example of a PPM.


