# Data Mining of User Navigation Patterns

José Borges and Mark Levene

Department of Computer Science, University College London,
Gower Street, London WC1E 6BT, U.K.
{j.borges, mlevene}@cs.ucl.ac.uk

**Abstract.** We propose a data mining model that captures the user navigation behaviour patterns. The user navigation sessions are modelled as a hypertext probabilistic grammar whose higher probability strings correspond to the user's preferred trails. An algorithm to efficiently mine such trails is given. We make use of the $N$gram model which assumes that the last $N$ pages browsed affect the probability of the next page to be visited. The model is based on the theory of probabilistic grammars providing it with a sound theoretical foundation for future enhancements. Moreover, we propose the use of entropy as an estimator of the grammar's statistical properties. Extensive experiments were conducted and the results show that the algorithm runs in linear time, the grammar's entropy is a good estimator of the number of mined trails and the real data rules confirm the effectiveness of the model.

## 1 Introduction

Data Mining and Knowledge Discovery is an active research discipline involving the study of techniques which search for patterns in large collections of data. Meanwhile, the explosive growth of the World Wide Web (known as the web) in recent years has turned it into the largest source of available online data. Therefore, the application of data mining techniques to the web, called *web data mining*, was the natural subsequent step and it is now the focus of an increasing number of researchers.

In web data mining there are currently three main research directions: (i) mining for information, (ii) mining the web link structure, and (iii) mining for user behaviour patterns. Mining for information focuses on the development of techniques to assist users in processing the large amounts of data they face during navigation and to help them find the information they are looking for, see for example [11]. Mining the link structure aims at developing techniques to take advantage of the collective judgement of web page quality in the form of hyperlinks, which can be viewed as a mechanism of implicit endorsement, see [5]. The aim is to identify for a given subject the authoritative and the hub pages. Authoritative pages are those which were conferred authority by the existing links to it, and hubs are pages that contain a collection of links to related authorities. Finally, the other research direction, which is being followed by an increasing number of researchers, is mining for user navigation patterns. This research focuses on

techniques which study the user behaviour when navigating within a web site. Understanding the visitors navigation preferences is an essential step in improving the quality of electronic commerce services. In fact, the understanding of the most likely access patterns of users allows the service provider to customise and adapt the site's interface for the individual user [16], and to improve the site's static structure within the underlying hypertext system [19].

When web users interact with a site, data recording their behaviour is stored in web server logs, which in a medium sized site can amount to several megabytes per day. Moreover, since the log data is collected in a raw format it is an ideal target for being analysed by automated tools. Currently several commercial log analysis tools are available [23]; however, these tools have limited analysis capabilities producing only results such as summary statistics and frequency counts of page visits. In the meantime the research community has been studying data mining techniques to take full advantage of information available in the log files. There have so far been two main approaches to mining for user navigation patterns from log data. In the first approach log data is mapped onto relational tables and an adapted version of standard data mining techniques, such as mining association rules, are invoked, see for example [8]. In the second approach techniques are developed which can be invoked directly on the log data, see for example [2] or [21].

In this paper we propose a new model for handling the problem of mining log data which directly captures the semantics of the user navigation sessions. We model the user navigation records, inferred from log data, as a hypertext probabilistic grammar whose higher probability generated strings correspond to the user's preferred trails. Therefore, a compact and self contained model of user interaction with the web is provided. There are two contexts in which such model is potentially useful. On the one hand, it can help the service provider to understand the users needs and as a result improve the quality of its service. The quality of service can be improved by providing adaptive pages suited to the individual user, by building dynamic pages in advance to reduce waiting time, or by providing a speculative service which sends, in addition to the requested document, a number of other documents that are expected to be requested in the near future. On the other hand, such a model can be useful to the individual web user by acting as a personal assistant integrated with his/her web browser.

In fact, if the browser keeps a user's log file which characterises his/her interactions with the web, a hypertext probabilistic grammar can be incrementally built. Such a grammar would be a representation of the user's knowledge of the web which can act as a memory aid, be analysed in order to infer the user preferred trails for a given subject, or work as a prediction tool to prefetch interesting pages in advance.

Section 2 presents the proposed hypertext grammar model, while Section 3 presents the results of the performed experiments. Section 4 discusses related work and Section 5 presents a preliminary discussion of recent improvements to the model. Finally, in Section 6 we give our concluding remarks and discuss further work.

## 2 Hypertext Probabilistic Grammars

A log file can be seen as a per-user ordered set of web page requests from which it is possible to infer the user navigation sessions. In this work we simply define a *user navigation session* as a sequence of page requests such that no two consecutive requests are separated by more than $X$ minutes, where $X$ is a parameter. In [4] the authors proposed for $X$ the value of 25.5 minutes which corresponds to $1\frac{1}{2}$ standard deviations of the average time between user interface events. Since then many authors have adopted the value of 30 minutes. We note however, that more advanced data preparation techniques, such as those described in [9], could be used in a data pre-processing stage to fully take advantage of all the information available in the log files.

The user navigation sessions inferred from the log data are modelled as a *hypertext probabilistic language* generated by a *hypertext probabilistic grammar* (or simply HPG) [14] which is a proper subclass of probabilistic regular grammars [24]. A HPG is a probabilistic regular grammar which has a one-to-one mapping between the set of non-terminal symbols and the set of terminal symbols. Each non-terminal symbol corresponds to a web page and a production rule corresponds to a link between pages. Moreover, there are two additional states, $S$ and $F$, which represent the start and finish states of the navigation sessions.

From the set of user sessions we obtain the number of times a page was requested, the number of times it was the first state in a session, and the number of times it was the last state in a session. The number of times a sequence of two pages appears in the sessions gives the number of times the corresponding link was traversed. The probability of a production from a state that corresponds to a web page is proportional to the number of times the corresponding link was chosen relative to the number times the user visited that page.

The probability of a production from the start state is proportional to the number of times the corresponding state was visited, implying that the destination node of a production with higher probability corresponds to a state that was visited more often. We define $\alpha$ as a parameter that attaches the desired weight to a state being the first in a user navigation session. If $\alpha = 0$ only states which were the first in a session have probability greater than zero of being in a production from the start state, on the other hand if $\alpha = 1$ all state visits are given proportionate weight. Note that when $\alpha > 0$ every grammar state has an initial probability greater than zero. The probabilities of the productions from the start state correspond to the vector of initial probabilities, $\pi$, and the probabilities of the other productions correspond to the transition matrix of a Markov chain [12].

In the example shown in Figure 1 we have 6 user sessions with a total of 24 page requests, wherein state $A_1$ was visited 4 times, 2 of which are the first state in a user session, therefore, since $\alpha = 0.5$ we have $\pi(A_1) = \frac{0.5 \cdot 4}{24} + \frac{0.5 \cdot 2}{6} = 0.25$. Figure 2 shows the grammar inferred from the given set of trails for $N = 1$ and $\alpha = 0.5$. (We freely utilise in our figures the duality between grammars and automata [13].)

| Session ID | User trail |
|---|---|
| 1 | $A_1 \rightarrow A_2 \rightarrow A_3 \rightarrow A_4$ |
| 2 | $A_1 \rightarrow A_5 \rightarrow A_3 \rightarrow A_4 \rightarrow A_1$ |
| 3 | $A_5 \rightarrow A_2 \rightarrow A_4 \rightarrow A_6$ |
| 4 | $A_5 \rightarrow A_2 \rightarrow A_3$ |
| 5 | $A_5 \rightarrow A_2 \rightarrow A_3 \rightarrow A_6$ |
| 6 | $A_4 \rightarrow A_1 \rightarrow A_5 \rightarrow A_3$ |

**Fig. 1.** An example set of user's trails.

In a HPG the probability of the first derivation step is defined as the *support threshold*, $\theta$, and is not factored into the derivation probability. Thus, the probabilities of the productions from the start state are used to prune out the strings which may otherwise have high probability but correspond to a subset of the hypertext system rarely visited. Moreover, a string is included in the grammar's language if its derivation probability is above the *cut-point*, $\lambda$, where the cut-point corresponds to the grammar *confidence* threshold. The values of the support and confidence thresholds give the user control over the quantity and quality of the trails to be included in the rule set. The strings generated by the grammar correspond to user navigation trails, and the aim is to identify the subset of these strings that best characterise the user behaviour when visiting the site. Parameters such as *confidence* and *support* are defined as preference measures which rank the generated strings. The algorithm used to mine rules having confidence and support above the specified thresholds is a special case of a directed graph Depth-First Search which performs an exhaustive search of all the strings with the required characteristics, cf. [2].

In Figure 3 we show the rules obtained with different model configurations. From the first two sets of rules we can see how both the number of rules and its average length decreases when the cut-point increases. Note that although $A_3$ was never the first state in a navigation session both rule-sets include rules starting with it. This occurs because $A_3$ has a high frequency of traversal and the value of parameter $\alpha = 0.5$ gives a positive weight to that. The third rule-set has

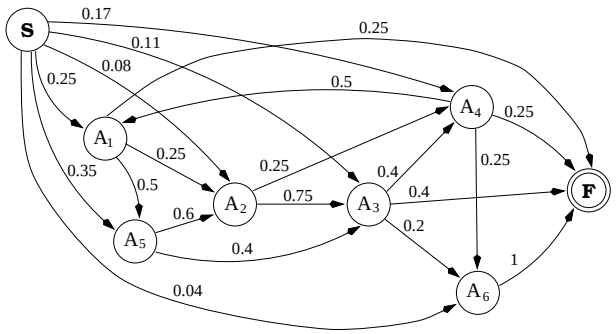


**Fig. 2.** The hypertext grammar for $N = 1$ and $\alpha = 0.5$.

a higher support threshold and as a consequence the rules starting in state $A_3$ are excluded. Finally, rule-set 4 corresponds to a grammar inferred while given the same weight for all traversals, i.e. $\alpha = 1.0$, and consequently rules starting at $A_2$ and $A_3$ are included even though these states were never the first in a navigation session.

| rule-set 1 | | rule-set 2 | | rule-set 3 | | rule-set 4 | |
|---|---|---|---|---|---|---|---|
| $\alpha = 0.5$ | | | | | | $\alpha = 1.0$ | |
| $\theta = 0.1$ | | | | $\theta = 0.15$ | | $\theta = 0.1$ | |
| $\lambda = 0.2$ | | $\lambda = 0.3$ | | $\lambda = 0.3$ | | $\lambda = 0.3$ | |
| rule | conf. | rule | conf. | rule | conf. | rule | conf. |
| $A_1A_2$ | 0.25 | $A_1A_5A_2$ | 0.3 | $A_1A_5A_2$ | 0.3 | $A_1A_5A_2$ | 0.3 |
| $A_1A_5A_3$ | 0.2 | $A_3A_4$ | 0.4 | $A_4A_1$ | 0.5 | $A_2A_3A_4$ | 0.3 |
| $A_1A_5A_2A_3$ | 0.23 | $A_4A_1$ | 0.5 | $A_5A_3$ | 0.4 | $A_3A_4$ | 0.4 |
| $A_3A_4A_1$ | 0.2 | $A_5A_3$ | 0.4 | $A_5A_2A_3$ | 0.45 | $A_4A_1$ | 0.5 |
| $A_3A_6$ | 0.2 | $A_5A_2A_3$ | 0.45 | | | $A_5A_3$ | 0.4 |
| $A_4A_1A_5$ | 0.25 | | | | | $A_5A_2A_3$ | 0.45 |
| $A_4A_6$ | 0.25 | | | | | | |
| $A_5A_3$ | 0.4 | | | | | | |
| $A_5A_2A_3$ | 0.45 | | | | | | |

**Fig. 3.** The rules obtained with various model configurations.

### 2.1 The *N*gram Model

We use the $N$gram concept [6] to determine the assumed user memory when navigating within the site, where $N$, $N \geq 1$, is called the history depth. Therefore, when the user is visiting a page it is assumed that only the $N$ previously visited pages influence the link he will choose to follow next. The intuition is that the user has a limited memory of the previously browsed pages and that the next choice depends only on the last $N$ pages browsed. In an $N$gram model each of the states of the HPG correspond to a sequence of $N$ pages visited.

The drawback of the $N$gram model is the increase in the number of states as the history depth increases. For a web site with $n$ pages and for a given history depth $N$ the expected number of grammar states is $n.b^{(N-1)}$, where $b$ is the average number of outlinks in a page. As such, there is a clear trade-off between the modelled user memory (measured by the history depth) and the grammar complexity (measured in the number of states).

Finally, the hypertext grammar model is incremental in the sense that when more log data becomes available it can be incorporated in the model without the need of rebuilding the grammar from scratch. Hence, the HPG is a compact way of keeping the browsing history since the grammar size depends only on the number of pages visited and not on the amount of log data available.

In Figure 4 we present the algorithm used to incrementally build a HPG. We let $T$ be the set of user navigation sessions and $|T|$ its cardinality, $T_i$ be the

$i^{th}$ session in the set and $T_i[j]$ be the $j^{th}$ page of session $i$. We also let $N$ be the history depth of the grammar, and *StateSet* be the set of grammar states; the function *StateSet.update*() updates *StateSet* with the last inferred state. A *State* corresponds to $N$ consecutive page requests $State[1], \ldots, State[N]$; the function *shiftLeft*() performs the assignment $State[i] := State[i+1]$, for $i = 1, \ldots, N$–1. In addition, we let *Productions* be the set of all grammar productions and the function *Productions.update*() updates the set with the last inferred production.

```
Algorithm 1 (Build_Grammar (T, N))
1. begin
2.   for i = 1 to |T|
3.     for k = 1 to N do
4.       State[k] = Ti[k];
5      end for
6.     StateSet.update(State);
7.     for j = N + 1 to |Ti|
8.       LeftState = State;
9.       State.shiftLeft();
10.      State[N] = Ti[j];
11.      StateSet.update(State);
12.      Productions.update(LeftState,State);
13.    end for
14.  end for
15.end.
```

**Fig. 4.** Incremental algorithm to build the hypertext probabilistic grammar.

### 2.2 The Entropy of a HPG

We propose to use the *entropy* of the HPG as an estimator of the statistical properties of the language generated by the grammar. The entropy is a measure of the uncertainty in the outcome of a random variable and in this context the sample space is the set of all strings generated by the grammar. If we had no information at all about the user interaction with the web the most rational thing would be to assume that all pages had the same probability of being visited and all its links have the same probability of being chosen. The entropy is maximum in this case and there is no point in looking for patterns or trying to predict a random behaviour. On the other hand, if the entropy is close to zero the user behaviour has few uncertainties, and a small set of short rules should contain enough information to characterise his/her behaviour. In conclusion, the intuition behind the estimator is that if the entropy of a probabilistic grammar is close to zero there should be a small set of strings with high probability and if

the entropy is high then there should be a large number of strings with similar and low probability.

Assuming a transition with probability one from state $F$ to $S$ a HPG corresponds to an irreducible and aperiodic Markov chain with a stationary distribution vector $\mu$ and transition matrix $A$. Thus, we can estimate the entropy with the following expression $H = H(\pi) + (-\sum_{ij} \pi_i A_{ij} log\ A_{ij})$ where $H(\pi)$ is included to take into account the randomness of the choice of the initial page, see [10] for detail on the entropy of a Markov chain. Note that we use the vector of initial probabilities $\pi$ as an estimator of the stationary vector $\mu$, since it is proportional to the number of times each state was visited. The value of $H$ can be normalised to be in the range between 0 and 1 by considering its ratio with the corresponding random grammar. The random grammar consists in a grammar with the same structure but in which all states have their out-links probabilities according to a uniform distribution.

Such a measure which is an estimator of the statistical distribution of the grammar string probabilities can be useful in helping the user in the specification of the support and confidence thresholds.

## 3 Experimental Evaluation

To assess the performance and the effectiveness of the proposed model experiments were conducted with both random and real data. Tests with random data provide the means of evaluating many different topologies and configurations of a HPG and tests with real data allow us to verify whether or not the model is potentially useful in practice.

The method used to create the random data consisted of four consecutive steps: (i) given the required number of states and the average branching factor randomly create a directed graph, (ii) for each grammar state assign outgoing links weights according with the chosen probability distribution, (iii) verify if the resulting grammar has the required properties, that is, if every state has a path to $F$ and if not add a link to $F$, (iv) normalise the grammar's weights, that is, calculate the production probabilities.

For the experiments with real data we used log files obtained from the authors of [17]. The log files contain two months of usage from the site `http://www.hyperreal.org/music/machines/`. It should be noted that the data was collected while caching was disabled and that we used the data without cleaning it. We divided each month into four subsets, each corresponding to a week, and for each subset we built the corresponding HPG for several values of the history depth. One of the weeks was discarded because it presented characteristics significantly different from the others, namely in the number of states inferred. This last fact indicates that the data should be cleaned; however, since data cleaning was not the aim of the work together with the belief that the probabilistic nature of the model could partially overcome the dirtiness of the data we decided to use the data without cleaning.

### 3.1 Performance

The first objective of the experiments was to evaluate the algorithm performance and its scalability. The experiments were conducted for several model configurations where the grammar's size, $n$, varied between 100 and 4000 states, the confidence threshold varied between 0.1 and 0.5. Note that the grammar size depends only on the number of pages in the hypertext system and not on the number of user sessions in the log data. In order to have a measure of support which allows us to compare the results for grammars with different sizes we have decided to define support in a way that takes into account the number of states $n$, that is $\theta = \frac{x}{n}$ where $x$ is an integer value. In our experiments we have adopted the value $\theta = \frac{1}{n}$ which means that only trails which have the first state with a frequency of traversal above the average will pass the test. For each configuration 150 runs were performed and the average results taken.

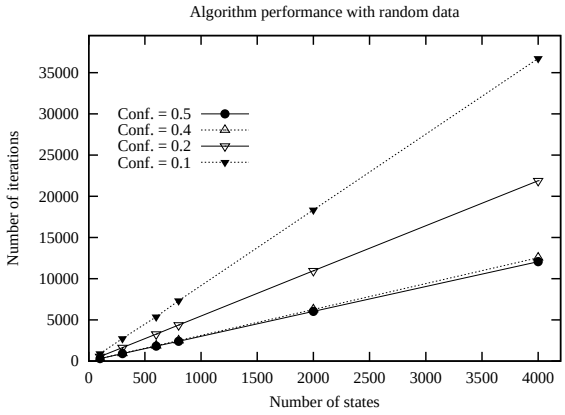


**Fig. 5.** Performance with random data.

Figure 5 shows the variation of the average number of iterations (for the 150 runs) with the grammar size for a fixed confidence. The experiments were performed for various values of the confidence threshold and the results suggest that there is a strong linear correlation between the grammar size and the number of iterations. From Figure 5 it can be also assessed how the performance decreases for lower values of the confidence threshold. The experiment's results showed that the performance when measured in CPU time follows a similar trend.

Figure 6 shows the performance analysis for the experiments with the real log files. It can be seen that it follows a trend similar to the verified for random data. Note, however, that while with real data each point in the plot represents a single run with random data each point corresponds to the average number of iterations for 150 runs. Moreover, in Figure 6 the points for $n > 4000$ correspond to runs of a $N$gram configuration.

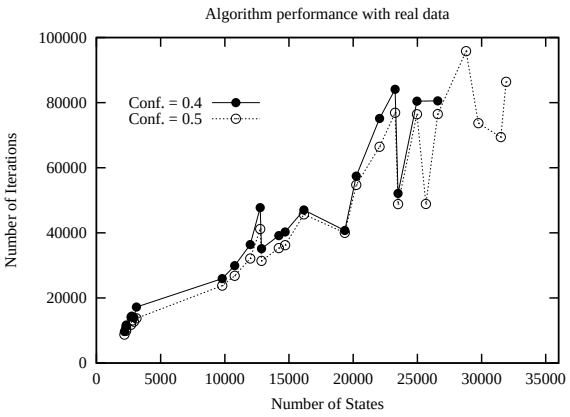


**Fig. 6.** Performance with real data.

Figure 7 shows how the number of rules obtained varies with both the size of the grammar and the confidence threshold. In fact, for a given confidence threshold the number of rules increases linearly with the number of states. On the other hand, for a given grammar size our experiments have also shown that the number of rules increases exponentially with the decrease of the confidence threshold.

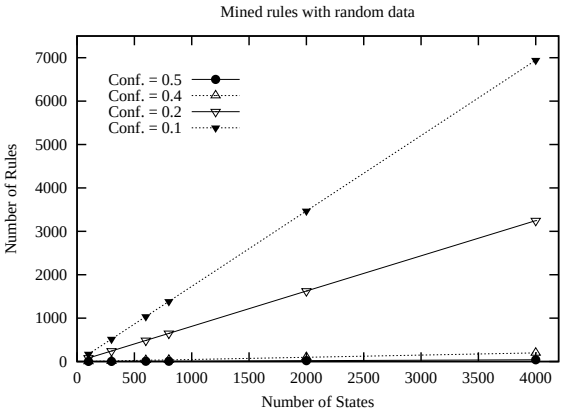


**Fig. 7.** The number of rules with the grammar size.

Figure 8 shows how the size of the longest rule obtained varies with the confidence threshold. It can be concluded that for high values of confidence we have as a result a small number of very short rules and for small values of confidence we obtain a large set of longer rules. Therefore, there is a clear trade-

off between obtaining a manageable number of rules and the desire of having rules with a significant length.

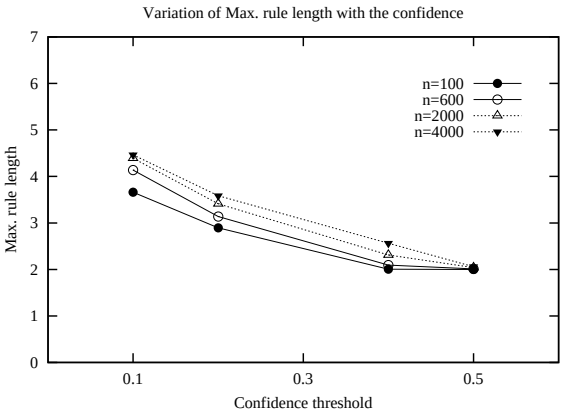


**Fig. 8.** The size of the longest rule with the confidence.

Figure 9 gives some examples of the rules mined from the real data. For each rule we give its probability and the average number of traversals of its links. Figure 10 shows the top-ten pages in the web site, which correspond to the states with the highest initial probability.

| Rule 1 | Conf.= 0.43 | Avg. No. Traversals = 44.5 | 4 pages |
|---|---|---|---|
| /<br>/ecards/<br>/ecards/cards/<br>/ecards/cards/96301 | | | |
| Rule 2 | Conf.= 0.53 | Avg. No. Traversals = 11 | 5 pages |
| /<br>/manufacturers/<br>/manufacturers/Music-and-More/<br>/manufacturers/Music-and-More/VF11/<br>/manufacturers/Music-and-More/VF11/info/VF11.review | | | |
| Rule 3 | Conf.= 0.56 | Avg. No. Traversals = 64 | 5 pages |
| /categories/drum-machines/samples/<br>/categories/drum-machines/samples/deepsky_kicks/<br>/categories/drum-machines/samples/deepsky_kicks/README/<br>/categories/drum-machines/samples/deepsky_kicks/<br>/categories/drum-machines/samples/ | | | |

**Fig. 9.** Example of rules mined from the real data.

### 3.2 Entropy

In order to verify the utility of the grammar's entropy as an estimator of the statistical properties of the grammar's language we calculated for each grammar

| The Top-Ten pages | | |
|---|---|---|
| Rk. | URL | Prob. |
| 1 | / | 0.088 |
| 2 | /manufacturers/ | 0.050 |
| 3 | /samples.html | 0.026 |
| 4 | /samples.html?MMAgent | 0.023 |
| 5 | /manufacturers/Roland/ | 0.020 |
| 6 | /links/ | 0.018 |
| 7 | /Analogue-Heaven/ | 0.017 |
| 8 | /categories/software/Windows/ | 0.016 |
| 9 | /search.html | 0.015 |
| 10 | /categories/software/ | 0.014 |

**Fig. 10.** Top Ten Pages.

size and confidence threshold the correlation between: (i) the entropy and the number of rules, (ii) the entropy and the number of iterations, and (iii) the entropy and the average rule length. Moreover, to assess the effect that both the confidence and support thresholds have on the grammar's entropy we decided to also measure the entropy of a grammar inferred from the set of mined rules, which we call the *posterior grammar*. The posterior grammar is no more than a HPG whose initial input trails are a set of rules mined from its parent grammar. We define PER as the entropy rate of the posterior grammar, and ER is the entropy of the parent grammar but using the vector of initial probabilities of the posterior grammar as the estimator for the stationary vector in the parent grammar.

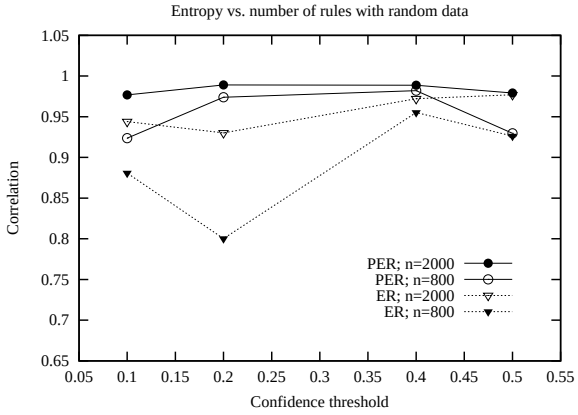


**Fig. 11.** Results with random data.

Figure 11 shows the results for grammars with 800 and 2000 states. Each point in the plot represents for a given configuration (size and confidence) the correlation between the number of rules mined and the entropy of the grammar. The results show that both PER and ER present a high correlation (above

0.8 with the number of rules but that PER is a better estimator. The results regarding ER suggest that we can obtain a good estimator for the number of rules from the original grammar provided we find a good estimator for the stationary vector. In fact, the results analysis suggest that the stationary vector has a fundamental importance in the overall quality of the estimator. The experimental results have also shown that the entropy is not a good estimator of the number of iterations or of the average rule length.

We note that the use of estimators from the posterior grammar is justified as an intermediate step in the search for the best way to obtain an estimator from the original grammar. Moreover, in a context where the user has a model that is periodically incremented with new log data the estimators obtained from the posterior grammar in one day can be used as the a priori estimators for the next day's grammar.

Figure 12 shows how the number of mined rules in the real data relates to the entropy in the posterior grammar, note that the logarithm of the number of rules was taken; a similar pattern was obtained for the random data. Note that while in Figure 11 each point represents the correlation between the grammar entropy and the number of rules for a specific grammar topology (for 150 runs), the plot in Figure 12 gives the overall variation of the number of rules with the PER and ER wherein each point corresponds to a single run for a specific topology. In conclusion, our experiments provide strong evidence that while the overall variation presents a regular *log*-linear trend, for a specific grammar topology there is a strong linear correlation between the number of rules and the grammar entropy.

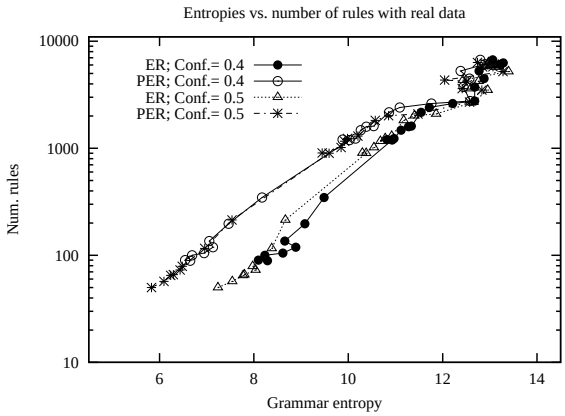


**Fig. 12.** Correlation between entropy and number of rules.

### 3.3 Ngram Model

The first order Markov chain provides an approximate model of user navigation patterns which can be improved by the $N$gram model noting that as we increase $N$ the entropy decreases. One question that arises when using the $N$gram model is whether there is a method to determine the order of the Markov chain that best models a given set of user navigation sessions. As was stated in Section 2.1 there is a tradeoff between the model complexity (measured by its number of states) and its accuracy. Moreover, if the order of the chain is too high the model is uninteresting due to the fact that user navigation sessions within a web site are typically short on average and also because the probability of a very long trail being repeated in the future is not very high.

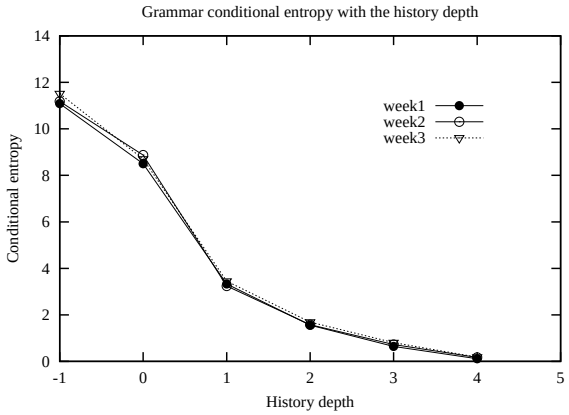


**Fig. 13.** Variation of the conditional uncertainty with the history depth.

In [18] the authors use the information theoretic measures of entropy and conditional entropy to analyse the variation of the conditional entropy with the increase of the grammar's order. A similar approach was described in [7] together with the specification of a statistical test to assess if the decrease in the conditional entropy is significant. The intuition is that if the decrease in the value of conditional entropy is not significant then the gain in the model accuracy by moving up in the chain's order is not sufficient to compensate for the additional complexity of the model. In practice the test is difficult to apply because of the extremely large number of degrees of freedom. Note that there is an amount of decrease that is due to the sample size; in fact, the smaller the sample size the faster we expect the conditional entropy to drop. In Figure 13 we show the variation of the conditional entropy with the history depth for the real data used in our experiments. The big drop from $d = 0$ to $d = 1$ shows that there is a big gain in accuracy when considering that the probability of choosing a page depends on the previous page as opposed to considering that

the probability of a page is independent of the path. For $d = -1$ the plot shows $log(n)$ which corresponds to a model where the choice probabilities are uniformly distributed.

In Figure 14 we show the increase in the grammar's number of states with the history depth. The values obtained are much lower than the worst case, $n.b^{(N-1)}$, and that might be explained by the insufficient amount of data available.

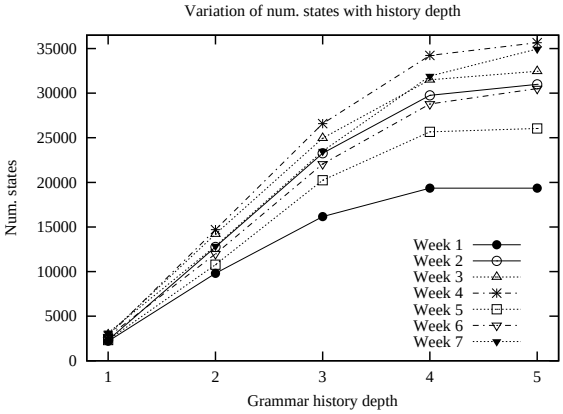


**Fig. 14.** Variation of the grammar size with the history depth.

## 4 Related Work

Some authors have recently proposed the user of a first order Markov model for predicting user requests on the WWW. In [1] a first order Markov model is proposed to implement a prefetching service to reduce server load. The model is build from past usage information and the transition probabilities between pages are proportional to the number of times both pages were accessed in a predefined time window. The experiment results show the method to be effective in reducing both the server load and the service time. A method is proposed in [15] wherein a dependency graph is inferred and dynamically updated as the server receives requests. There is a node for every requested page and an arc between two nodes exists if the target node was requested within $x$ accesses after the source node, the arc's weight is proportional to the number of such requests. The simulations performed with log data show that a reduction in the retrieval latency can be achieved. In [18] the authors present a study of the quality of a $k^{th}$ order Markov approximation as a model of predicting user surfing patterns. Using information theoretic measures they conclude that the best compromise between the model accuracy and its complexity occurs with the first order Markov model. Moreover, they show that the model probabilities are more stable over time for lower orders models.

The use of data mining techniques to analyse log data was first proposed by [8] and [25]. In [8] the log data is converted into a set of maximal forward references, a form which is amenable to being processed by existing association rules techniques. Two algorithms are given to mine the rules, which in this context consist of large itemsets with the additional restriction that the references must be consecutive in a transaction. In [25] is proposed a method to classify web site visitors according to their access patterns. Each user session is stored in a vector that contains the number of visits to each page and an algorithm is given to find clusters of similar vectors. The method does not take into account the order in which the page visits took place.

In [16] the authors challenged the AI community to use the the log data to create adaptive web sites and in [17] they present a technique which automatically creates index pages from the log data, i.e., pages containing collections of links which the user navigation behaviour suggests are related. In our previous work, [2], we proposed to model the log data as a directed graph with the arcs weights interpreted as probabilities that reflect the user interaction with the site, and we generalised the association rule concept. The authors of [20] propose to use log data to predict the next URL to be requested so the server can generate in advance web pages with dynamic content. A tree which contains the user paths is generated from the log data and an algorithm is proposed to predict the next request given the tree and the current user session. In [22] the authors propose a log data mining system composed of an aggregation module and a data mining module for the discovery of patterns with predefined characteristics. The aggregation module infers a tree structure from the data in which the mining is performed by a human expert using a mining query language.

Finally, [26] propose the integration of data warehousing and data mining techniques to analyse web records, and [9] study cleaning and preparation techniques which convert log data into user navigation sessions in a form amenable to processing by the existing data mining techniques.

## 5 Heuristics for Mining High Quality Patterns

We are currently working on the specification of new algorithms to improve the quality of the results relative to the deterministic DFS used in the experiments reported herein. In fact, as was shown in section 3.1, the exhaustive computation of all grammar strings with probability above the cut-point has the drawback of potentially returning a very large number of rules for small values of the cut-point. Note that if the user wants to find longer rules the threshold needs to be set low. This fact led us to the study of heuristics which allow us to compute a subset of the rule set while being able to control both its size and quality. In the following sections we briefly describe the ideas behind the heuristics we are developing; a complete description can be find in [3].

### 5.1 The Iterative Deepening Fine Grained Heuristic

Our first approach is based on the iterative deepening concept wherein the rule set is incrementally augmented until it is close enough to the desired set. We call this method the *iterative deepening fine grained* approach. A measure of the distance between the currently explored and the final rule set is provided by the error, and by setting its desired value the user has control over the number of rules to be returned. Informally, the error is defined as the amount of probability left to explore, and it can be estimated by the sum of the probabilities of the trails that are not yet final. Moreover, a criterion is provided to decide which trails to augment at each stage in order to maintain a high quality rule set.

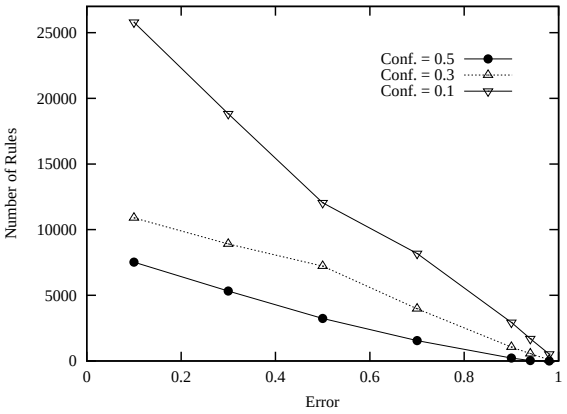


**Fig. 15.** Results of the iterative deepening fine grained heuristic.

Figure 15 shows the variation of the number of rules with the user specified error for a random grammar with 10000 states and average branching factor of 3, similar results were obtained for other configurations. Note that when the error is zero the resulting rule set is the same as the one given by the exhaustive computation done by the DFS. Since the variation is close to linear the proposed heuristic gives the user good control over the number of rules. A more detailed explanation together with extensive experimental results can be found in [3].

### 5.2 The Inverse Fisheye Heuristic

The second approach aims at providing the user with a method that gives a relatively small set of long rules. We propose the use of a dynamic threshold which imposes a very strict criterion for small rules and becomes more permissible when the trails get longer. We call this method *inverse fisheye* and with it, the early stages of the exploration are performed with the threshold value set high implying that only the best trails will pass the evaluation. In the subsequent

stages the threshold has its value progressively reduced according to one of the available criterions allowing the trails to be further explored. Moreover, the user has to specify the maximum exploration depth since there is no guarantee that the algorithm terminates, specially when the cut-point decreases at a higher rate than the trail probability.

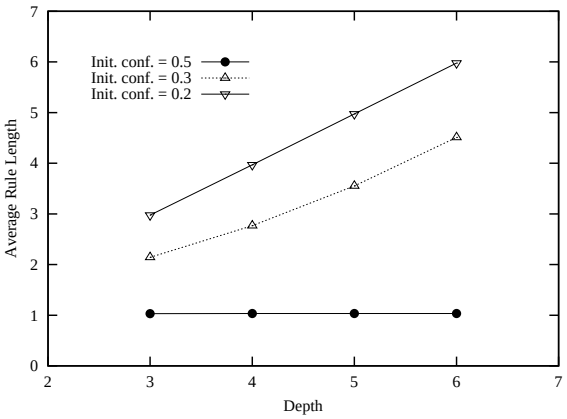


**Fig. 16.** Results of the inverse fisheye heuristic.

Figure 16 shows how the average rule length varies with the exploration depth for a random grammar with 1000 states and average branching factor of 5, similar results were obtained for other configurations. The preliminary results suggest that by setting the initial confidence and the intended depth of exploration the user can have control over both the size of the rule set and its average rule length. Again, more detail of the heuristic and the experimental results can be found in [3].

## 6 Concluding Remarks

We have proposed a model of hypertext to capture user preferences when navigating through the web. We claim that our model has the advantage of being compact, self contained, coherent, and based on the well established work in probabilistic grammars providing it with a sound foundation for future enhancements including the study of its statistical properties. In fact the size of the model depends only on the size of the web site being analysed and not on the amount of data collected.

The set of user navigation sessions is modelled as a hypertext probabilistic grammar, and the set of strings which are generated with higher probability correspond to the navigation trails preferred by the user. An algorithm to efficiently mine these strings is given. Extensive experiments with both real and random

data were conducted and the results show that, in practice, the algorithm runs in linear time in the size of the grammar. Moreover, the entropy of the posterior grammar is shown to be a good estimator of the number of rules output from our algorithm, and the experiments with real data confirm the effectiveness of our algorithm. Our model has potential use both in helping the web site designer to understand the preferences of the site's visitors, and in helping individual users to better understand their own navigation patterns and increase their knowledge of the web's content.

A brief introduction was provided to the heuristics we are currently studying which aim to improve the quality of mining for user navigation patterns. Moreover, we intend to further study the use of information theoretic measures and their relation to the model properties. We are also planning to incorporate relevance measures of web pages in order to assist the user in locating useful information.

## References


1. Azer Bestavros. Using speculation to reduce server load and service time on the www. In *Proc. of the fourth ACM International Conference on Information and Knowledge Management*, pages 403–410, Baltimore,MD, 1995.
2. J. Borges and M. Levene. Mining association rules in hypertext databases. In *Proc. of the fourth Int. Conf. on Knowledge Discovery and Data Mining*, pages 149–153, August 1998.
3. José Borges and Mark Levene. Heuristics for mining high quality user web navigation patterns. Research Note RN/99/68, Department of Computer Science, University College London, Gower Street, London, UK, October 1999.
4. Lara D. Catledge and James E. Pitkow. Characterizing browsing strategies in the world wide web. *Computer Networks and ISDN Systems*, 27(6):1065–1073, April 1995.
5. Soumen Chakrabarti, Byron E. Dom, David Gibson, Jon Kleinberg, Ravi Kumar, Prabhakar Raghavan, Sridhar Rajagopalan, and Andrew S. Tomkins. Mining the link structure of the world wide web. *IEEE Computer*, 32(8):60–67, August 1999.
6. E. Charniak. *Statistical Language Learning*. The MIT Press, 1996.
7. Christopher Chatfield. Statistical inferences regarding markov chain models. *Applied Statistics*, 22:7–20, 1973.
8. M.-S. Chen, J. S. Park, and P. S. Yu. Efficient data mining for traversal patterns. *IEEE Trans. on Knowledge and Data Eng.*, 10(2):209–221, March/April 1998.
9. R. Cooley, B. Mobasher, and J. Srivastava. Data preparation for mining world wide web browsing patterns. *Knowledge and Information Systems*, 1(1):5–32, February 1999.
10. T. Cover and J. Thomas. *Elements of Information Theory*. John Wiley & Sons, 1991.
11. M. Craven, D. DiPasquo, D. Freitag, A. McCallum, T. Mitchell, K. Nigam, and S. Slattery. Learning to extract symbolic knowledge from the world wide web. In *Proc. of the 15th National Conf. on Artificial Intelligence*, pages 509–516, July 1998.
12. W. Feller. *An Introduction to Probability Theory and Its Applications*. John Wiley & Sons, second edition, 1968.
13. J. Hopcroft and J.Ullman. *Introduction to Automata Theory, Languages, and Computation*. Addison-Wesley, 1979.
14. M. Levene and G. Loizou. A probabilistic approach to navigation in hypertext. *Information Sciences*, 114:165–186, 1999.

15. Venkata N. Padmanabhan and Jeffrey C. Mogul. Using predictive prefetching to improve world wide web latency. *Computer Communications Review*, 26, 1996.
16. Mike Perkowitz and Oren Etzioni. Adaptive web sites: an AI challenge. In *Proc. of Fifteenth International Joint Conference on Artificial Intelligence (IJCAI-97)*, pages 16–21, Nagoya, Japan, August 1997.
17. Mike Perkowitz and Oren Etzioni. Adaptive sites: Automatically synthesizing web pages. In *Proc. of the Fifteenth National Conference on Artificial Intelligence (AAAI-98)*, pages 727–732, Madison, Wisconsin, July 1998.
18. Peter L.T. Pirolli and James E. Pitkow. Distributions of surfers' paths through the world wide web: Empirical characterizations. *World Wide Web*, 2:29–45, 1999.
19. L. Rosenfeld and P. Morville. *Information Architecture for the World Wide Web.* O'Reilly, 1998.
20. S. Schechter, M. Krishnan, and M. D. Smith. Using path profiles to predict http requests. *Computer Networks and ISDN Systems*, 30:457–467, 1998.
21. M. Spiliopoulou, L. C. Faulstich, and K. Wilkler. A data miner analyzing the navigational behaviour of web users. In *Proc. of the Workshop on Machine Learning in User Modelling of the ACAI99*, Greece, July 1999.
22. Myra Spiliopoulou and Lukas C. Faulstich. WUM: a tool for web utilization analysis. In *Proc. of the International Workshop on the Web and Databases (WebDB'98)*, pages 184–203, Valencia, Spain, March 1998.
23. R. Stout. *Web Site Stats: tracking hits and analyzing traffic.* Osborne McGraw-Hill, 1997.
24. C. S. Wetherell. Probabilistic languages: A review and some open questions. *Computing Surveys*, 12(4):361–379, December 1980.
25. T. W. Yan, M. Jacobsen, H. Garcia-Molina, and U. Dayal. From user access patterns to dynamic hypertext linking. In *Proc. of the 5th Int. World Wide Web Conference*, pages 1007–1014, 1996.
26. O. R. Zaïane, 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.*, pages 12–29, Santa Barbara, CA, April 1998.