Mining Navigation Patterns with Hypertext Probabilistic Grammars

José Borges and Mark Levene
Department of Computer Science
University College London
Gower Street
London WC1E 6BT, U.K.
Email:{j.borges, mlevene}@cs.ucl.ac.uk

RN/99/08

Abstract

The continuous growth in the size and use of the World Wide Web is creating difficulties in both the design of Web sites to suit a variety of different users and in the navigation through very large web structures of pages and links. We propose a data mining model that captures the user navigation behaviour patterns. The set of user navigation sessions, which characterise the interaction with the Web pages visited, are modelled as a Hypertext probabilistic grammar, whose higher probability derivated strings correspond to the user's preferred trails. An algorithm to efficiently mine such trails is given. In addition, we make use of the N-gram model which assumes that, when the user is browsing a given page, only the last n pages browsed affect the probability of the next page to be visited. Moreover, we propose the use of entropy as an estimator of the statistical properties of the grammar. Extensive experiments with both real and random data were conducted and the results show that, in practice, our algorithm runs in linear time in the size of the grammar. Our experiments also show that the entropy of the grammar is a good estimator of the number of mined trails, i.e. the mined rules, and the results from the experiments with the real data confirm the effectiveness of our model.

Keywords. Web data mining, hypertext, trails, probabilistic grammars.

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 two main research directions being followed: (i) mining for information and (ii) mining for user behaviour patterns. Mining for information focuses on the development of techniques to assist users in processing the large amounts of data faced during navigating and help them find the information they are looking for; see for example (Craven *et al.* 1998). On the other hand, mining for user navigation patterns focuses on techniques which study the user behaviour when navigating within a Web site. Understanding the visitors navigation preferences is an essential step both in the process of customising the site interface for the individual user (adaptive web sites as suggested in (Perkowitz & Etzioni 1997)) and of improving the site's static structure of the underlying hypertext system (Rosenfeld & Morville 1998).

When Web users interact with a site, data recording their behaviour is stored in Web server logs. In a medium size site the log files can amount of 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. There are currently available several commercial log analysis tools, (Stout 1997); however, these tools have limited analysis capabilities producing only results such as summary statistics and frequency counts of page visits.

There have been so far two approaches to mining for user navigating patterns. In the first approach the log data is mapped into relational tables and then standard data mining techniques are invoked (Chen, Park, & Yu 1998). In the second approach new models and techniques have been developed which can be invoked directly on the log data (Borges & Levene 1998).

In this paper we adopt the second approach by specifying a new model for handling the problem which directly captures the semantics of the user navigation sessions. We propose to model the user navigation records as a *hypertext probabilistic language* (Levene & Loizou 1999) generated by a *hypertext probabilistic grammar*, which is within the class of *regular probabilistic grammars*. A log file can be seen as a per-user ordered set of Web page requests from which we can infer the user navigation sessions and afterwards, the hypertext probabilistic grammar. A session is a sequence of requests such that no two consecutive requests are separated by more than X minutes, where X is a parameter. We then build a N-grammar, where n is called the history depth, which determines the assumed user memory when navigating within the site. If $n = x$ we are assuming that when the user is inspectioning a given page only the last x previously visited pages influence the link he/she will choose to follow next. Assuming n, we construct grammars such that each of its states correspond to n pages visited in sequence. In Figure 1 we show the grammar with $n = 1$ built from the set of navigation trails (sessions) shown in the figure. This example will be used throughout the paper, where the details will be clarified. (We freely utilise the duality between grammars and automata in our figures (Hopcroft & Ullman 1979))

ID	Trail
1	$A_1 \rightarrow A_2 \rightarrow A_3 \rightarrow A_4$
2	$A_1 \rightarrow A_2 \rightarrow A_3$
3	$A_5 \rightarrow A_2 \rightarrow A_4$
4	$A_5 \rightarrow A_2 \rightarrow A_3 \rightarrow A_6$
5	$A_5 \rightarrow A_2 \rightarrow A_4 \rightarrow A_6$
6	$A_1 \rightarrow A_2 \rightarrow A_3$

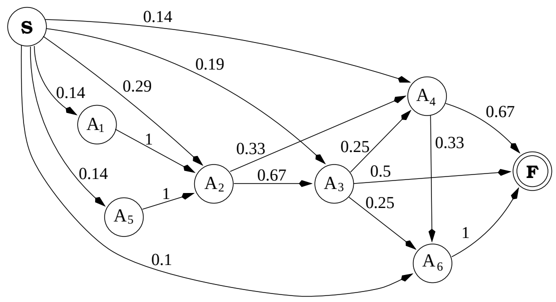

Figure 1: A set of trails and the corresponding Hypertext grammar.

In the inferred grammar, the probability of a production is proportional to the number of times the corresponding sequence of page requests occurs in the log files. The strings generated by the grammar correspond to the user navigation trails, and the aim is to identify the subset of these strings that better characterises the user behaviour when visiting the site. Parameters such as *confidence* and *support* are defined as preference measures which rank the generated strings. Moreover, we present an algorithm to efficiently mine the set of strings with higher probability. Finally, we study the applicability of the information theoretic measure of entropy (Cover & Thomas 1991), as a parameter allowing us to estimate the expected statistical properties of the results.

Experiments with both real and random data were performed and the results show that the algorithm has linear time performance in practice, that the grammar entropy is a good estimator of the number of mined rules, and, in addition, the results from the real data confirm the effectiveness of the model.

The rest of the paper is organised as follows. Section 2 presents the proposed Hypertext model and Section 3 presents the data mining algorithms which construct the grammar and mine it for trails. In Section 4 the experimental results obtained with both real data and synthetic data are presented and Section 5 discusses related work. Finally, Section 6 presents our concluding remarks and plans for future work.

2 Hypertext Model

2.1 Hypertext Databases

We formalise the essential characteristics of navigation in a Hypertext system, (Nielsen 1990), via a Hypertext database which is a directed graph whose nodes represent units of information and whose arcs allow a user to follow links between the information units.

Definition 1 (Hypertext database) A Hypertext database $\mathcal{H}$ is an ordered pair, $< \mathcal{S}, \mathcal{R} >$, where $\mathcal{S}$ is an information repository and $\mathcal{R}$ is a reachability relation.

$\mathcal{S} = \{s_1, \ldots, s_N\}$ consists in a finite set of states each corresponding to a page in the Hypertext system, and $\mathcal{R}$ is binary relation in $\mathcal{S}$ such that whenever $(s_i, s_j) \in \mathcal{R}$ there is a link in the Hypertext system from s_i, the *anchor* node, to s_j, the *destination* node.

The reachability relation, $\mathcal{R}$, represents all the links a user can traverse when browsing the Hypertext database, inducing the set of trails that can be followed, where a trail consists in a finite sequence of states corresponding to a walk in $\mathcal{R}$.

Note that in a Hypertext database every state is a potential source and destination of a trail, i.e., a user is free to start or finish the navigation in any page of the Hypertext system depending on the topology of $\mathcal{R}$.

2.2 Regular Grammars

We now give the necessary background concerning regular grammars; refer to (Hopcroft & Ullman 1979) for more detail.

An *alphabet* is a finite nonempty set of symbols, such as $A = \{a_1, \ldots, a_n\}$. As usual, A^* denotes the set of all finite length strings over A, including the empty string ϵ, and A^+ denotes the set $A^* - \{\epsilon\}$. A *language* L over A is any subset of A^*. A *grammar* is a generative device capable of generating all the strings in a language and nothing more.

Definition 2 (Regular grammar) A regular grammar (or simply a grammar) is a 4-tuple $G = < V, \Sigma, S, P >$, where:

1. V is a finite alphabet of nonterminals, $\{S, A_1, \ldots, A_n\}$.
2. Σ is a finite alphabet of terminals, $\{a_1, \ldots, a_n\}$; $V \cap \Sigma = \emptyset$.
3. $S \in V$ is a unique start symbol.
4. P is a finite set of production rules with the general form $A_i \rightarrow a$ or $A_i \rightarrow aA_j$, where $A_i, A_j \in V$ and $a \in (\Sigma \cup \epsilon)$.

In a grammar G, a one-step derivation of string w_2 from string w_1, $d : w_1 \Rightarrow w_2$, occurs when a production is applied to w_1 to obtain w_2. A *derivation*, $d : w_1 \Rightarrow^* w_n$, is a finite sequence of one-step derivations that derives w_n from w_1. A *sentential form* is any derivation from the unique starting symbol S. The language generated by a grammar G is the set of all sentential forms composed only of terminal symbols, $L(G) = \{w \in \Sigma^* | S \Rightarrow^* w\}$. A grammar G is *ambiguous* if there is a string $w \in L(G)$ such that w has at least two distinct derivations from the starting symbol S. Otherwise, G in *unambiguous*.

A sentential form in a regular grammar has at most one nonterminal symbol; therefore, a production of type $A_i \rightarrow a_j$ is called a *final production* because it terminates the derivation process, and a production of type $A_i \rightarrow a_j A_j$ is called a *transitive production*. The length of a derivation, D, is the number of productions applied in the string derivation, which in a regular grammar corresponds to the length of the generated string.

The productions are assigned a two-index numbering, $P_{i,j}$, to facilitate their reference; the first index corresponds to the nonterminal in its left-hand side and the second to its position in the group of productions with the same left-hand side.

2.3 Probabilistic Regular Grammars

In a probabilistic grammar (Wetherell 1980) each production has an attached probability; therefore, each string in the grammar language have an associated probability.

Definition 3 (Probabilistic grammar) A probabilistic grammar, G_p, is a 2-tuple $< G, \mathcal{P} >$ where G is a grammar and $\mathcal{P}$ is an ordered set of probabilities, $p_{i,j} \in \mathcal{P}$, that are assigned by a one-to-one mapping to the production rules $P_{i,j} \in P$. In the following we let $p(P_{i,j}) = p_{i,j}$.

A probabilistic grammar G_p is proper if $\{\forall i | 1 \leq i \leq |V|\}, \sum_{\{j|P_{i,j} \in P\}} p_{i,j} = 1$.

The probability of a derivation, d, is given by the product of the probabilities of the productions used in the string derivation, $p(d) = \prod_{1 \leq k \leq D} p(P_k)$, where $p(P_k)$ is the probability of the k^{th} production, $1 \leq k \leq D$. The probability of a string is given by the sum of the probabilities of all its derivations in the grammar, $p(w) = \sum_{d \in G} p(d), w \in L(G)$.

The skeleton of a probabilistic grammar G_p is the grammar G having the production $X \rightarrow \beta$ if and only if G_p has the production $X \rightarrow \beta$ with probability $p > 0$.

2.4 Hypertext Probabilistic Grammars

We now give the definition of a Hypertext Grammar (HG) which is in the class of regular grammars. A Hypertext grammar will be our representation of the Hypertext database.

Definition 4 (Hypertext grammar) A Hypertext grammar, G_h, representing a Hypertext database, $G_h = \mathcal{H} =< \mathcal{S}, \mathcal{R} >$, is a fourtuple $< V, \Sigma, S, P >$ where:

1. $V = \{S, A_1, \ldots, A_{|\mathcal{S}|}, F\}$ is a finite alphabet of $|\mathcal{S}| + 2$ nonterminal symbols. S is a unique start state, F is a unique final state, and there is a one-to-one and onto mapping from $V - \{S, F\}$ to $\mathcal{S}$.
2. $\Sigma = \{a_1, \ldots, a_{|\mathcal{S}|}\}$ is a finite alphabet of terminal symbols such that there is a one-to-one and onto mapping from Σ to $\mathcal{S}$, or equivalently from Σ to $V - \{S, F\}$.
3. P is a finite set of productions of the form $X \rightarrow yY$ with $X, Y \in V$ and $y \in \Sigma \cup \epsilon$ such that: (i) $S \rightarrow a_i A_i \in P$, $\forall A_i \in V - \{S, F\}$ (ii) $A_i \rightarrow a_j A_j \in P$ if and only if $(s_i, s_j) \in \mathcal{R}$ (iii) $A_i \rightarrow F \in P$, $\forall A_i \in V - \{S, F\}$ and finally (iv) $F \rightarrow \epsilon \in P$.

Each state $A_i \in V - \{S, F\}$ corresponds to a hypertext page and the productions of type (ii) represent the links between pages (the reachability relation). Both S and F are artificial states and while productions of type (i) mean that the navigation can start in any given page, productions of type (iii) mean that the user can stop the navigation in any page. Finally, the production (iv) derives the final string which only includes terminal symbols.

It can be proved that every Hypertext grammar is unambiguous because the one-to-one and onto mapping from Σ to $V - \{S, F\}$ and the fact that every transitive production is of the type $X \rightarrow a_i A_i, X \in V - \{F\}, A_i \in V - \{S, F\}$ and $a_i \in \Sigma$, implying that there is only one possible derivation for each string in the grammar.

It is shown in (Levene & Loizou 1999) that the class of Hypertext languages, generated from Hypertext grammars, is closed under substrings, join and intersection but is not closed under union, complement or concatenation.

While a Hypertext grammar models a Hypertext database a Hypertext Probabilistic Grammar (or simply HPG) models the user's interaction with it, the grammar probabilities are determined from statistical information collected from past usage of the Hypertext system. The probability of a production is assigned in such a way that it is proportional to the frequency with which the corresponding link was traversed; see Section 3.1 for the algorithm to infer a probabilistic hypertext grammar from log data.

We now give the definition of a Hypertext probabilistic grammar where $|A_i|, |A_i^s|$ and $|A_i^f|, A_i \in V - \{S, F\}$ are respectively the number of times A_i occurs in the user navigation sessions, the number of times A_i is the first state in a navigation session, and the number of times A_i is the last state in a navigation session; $|(A_i, A_j)|$ gives the number of times the respective link was traversed. Moreover, α is 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 a probability greater than zero of being in a production from the start state S, on the other hand when $\alpha = 1$ all state visits are given proportionate weight. Figure 1 shows the assigned production probabilities for the set of trails in the example with $n = 1$ and $\alpha = 1$, where n is the history depth.

Definition 5 (Hypertext probabilistic grammar) A Hypertext probabilistic grammar with *cut-point* $\lambda \in [0, 1)$ representing a Hypertext database, $\mathcal{H} = <\mathcal{S}, \mathcal{R}>$, is a 2-

tuple $G_{hp} =< G_h, \mathcal{P} >$ where G_h is a Hypertext grammar and $\mathcal{P}$ is an ordered set of probabilities, $p_{i,j} \in \mathcal{P}$, that are assigned by a one-to-one mapping to the productions $P_{i,j} \in P$. The production probabilities are assigned according to the expressions:

1. $p(S \to a_i A_i) = \alpha \frac{|A_i|}{\sum_{j=1}^{|V|} |A_j|} + (1-\alpha) \frac{|A_i^s|}{\sum_{j=1}^{|V|} |A_j^s|}$,
2. $p(A_i \to a_j A_j) = \frac{|(A_i, A_j)|}{\left(\sum_{\{x|(A_i,x) \in \mathcal{R}\}} |(A_i,x)|\right) + |A_i^f|}$,
3. $p(A_i \to F) = \frac{|A_i^f|}{\left(\sum_{\{x|(A_i,x) \in \mathcal{R}\}} |(A_i,x)|\right) + |A_i^f|}$.

In a HPG a string is included in the grammar language if its derivation probability is above the cut-point, where the cut-point can be viewed as the grammar *confidence* threshold.

In Definition 5 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. Therefore, in a HPG the probability of the first derivation step is defined as the *support threshold*, θ, and is not factored into the derivation probability (see Section 2.3), which is $p(d) = \prod_{2 \leq k \leq D} p(P_k)$, where $p(P_k)$ is the probability of the k^{th} production, $1 \leq k \leq D$. Thus, the probabilities of the productions from the start state are used to prune out the strings which would otherwise have high probability but correspond to a subset of the Hypertext system rarely visited.

Note that a string can be included in the grammar language without its derivation reaching state F. We stipulate that the sub-trails traversed on the way to F should also belong to the language since they represent trails navigated by the user; therefore, a string whose generation terminates in the state F contains additional information, meaning that the user not only traversed the corresponding trail but also finished the session in the last state of the derivation.

Definition 6 (Hypertext probabilistic language) The language generated by a G_{hp} is $L(G_{hp}) = \{w \mid w \in \Sigma^*, \mathcal{S} \Rightarrow^* w \text{ and } p(w) > \lambda, p(P_1) > \theta \text{ with support } \theta \in [0,1)\}$.

The set of trails in Figure 1 induce the HPG $G_hp =< V, \Sigma, S, P >$ where $V = \{S, A_1, \ldots, A_6, F\}$, $\Sigma = \{a_1, \ldots, a_6\}$. The productions and respective probabilities are as shown in Figure 2. Figure 3 gives the grammar strings for $\lambda = 0.3$ and $\theta = 0.1$.

Production	Prob.	Production	Prob.	Production	Prob.
$S \to a_1 A_1$	0.14	$A_1 \to a_2 A_2$	1	$A_1 \to F$	0
$S \to a_2 A_2$	0.29	$A_2 \to a_3 A_3$	0.67	$A_2 \to F$	0
$S \to a_3 A_3$	0.19	$A_2 \to a_4 A_4$	0.33	$A_3 \to F$	0.5
$S \to a_4 A_4$	0.14	$A_3 \to a_4 A_4$	0.25	$A_4 \to F$	0.67
$S \to a_5 A_5$	0.14	$A_3 \to a_6 A_6$	0.25	$A_5 \to F$	0
$S \to a_6 A_6$	0.1	$A_5 \to a_2 A_2$	1	$A_6 \to F$	1

Figure 2: Grammar productions probabilities.

For the sake of completeness we now give the definition of *induced grammar* which is the non-probabilistic grammar that generates the strings of a HPG with a cut-point.

Definition 7 (Induced grammar) Let G_h be a Hypertext grammar then $\varphi(G_h) =< V, \Sigma, S, P' >$ is the induced grammar, where $P' = P \cup \{X \to \epsilon \mid X \in V - \{S, F\}\}$.

2.5 The N-Gram Model

From the set of user navigation sessions it is possible to determine the probability with which a user will browse a page given the sequence of previously visited pages in the session. Thus, $P(S_m|S_n, S_{n-1}, \ldots, S_1)$ represents the probability that page S_m will be chosen next given that pages $S_n, S_{n-1}, \ldots, S_1$ were browsed previously; $S_n, S_{n-1}, \ldots, S_1$ is the *history* of the session and n is the *history depth*.

One model widely used in statistical language modelling for speech recognition is the N-gram model (Charniak 1996). This model makes the assumption that only the previous n symbols have a direct effect on the probability of the next symbol to appear. A common n is 2, a Trigram, meaning that $P(S_m|S_n, S_{n-1}, \ldots, S_1) = P(S_m|S_n, S_{n-1})$. In the context of user navigation sessions, the intuition is that the user has a limited memory of the previously browsed pages, or in other words, that it is reasonable to assume that the choice of the page to browse next does not depend on the complete history of the session but only on the last n pages browsed.

String	Confidence	String	Confidence	String	Confidence
$A_1A_2A_3$	0.67	A_2A_4	0.33	$A_5A_2A_4$	0.33
$A_1A_2A_4$	0.33	A_4A_6	0.33		
A_2A_3	0.67	$A_5A_2A_3$	0.67		

Figure 3: Grammar strings for $n = 1$.

In the N-gram model each state corresponds to a navigation trail having n pages; therefore, the grammar does not generate strings w of which $|w| < n$. Moreover, n has to be smaller than the largest trail in the trail set because trails with length smaller than n cannot correspond no any grammar state.

String	Confidence
$A_1A_2A_3$	1
$A_2A_4A_6$	0.5
$A_5A_2A_3$	0.33
$A_5A_2A_4A_6$	0.33

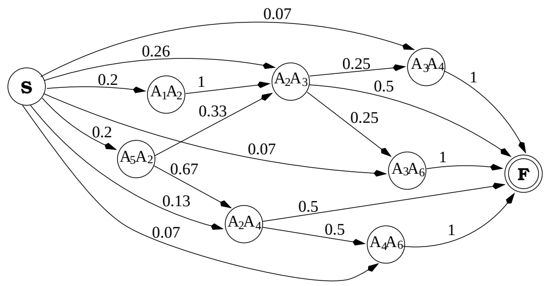

Figure 4: Grammar and its strings for $n = 2$.

The drawback of the N-gram model is the the exponential increase in the number of its states as the history depth increases. In a language whose alphabet has l symbols the N-gram model can lead to a grammar with l^n states; however, in practice the number of states is limited by the Hypertext database topology and the worst case can only occur if the reachability relation corresponds to a complete directed graph. In conclusion, there is a clear trade-off between the size of the user memory modelled (measured by the history depth) and the grammar complexity (measured by number of states). Figure 4 shows the grammar with $n = 2$ and the rules for $\lambda = 0.3$ and $\theta = 0.1$.

2.6 Information Theoretic Measures and Markov Chains

Consider a random variable Y with sample space $\{y_1, \ldots, y_n\}$, where $p(y_i) = p_i$ denotes its corresponding probabilities. The *entropy* of Y is defined as $H(Y) = -\sum_{i=1}^{n} p(y_i) log p(y_i)$ (Cover & Thomas 1991). The entropy measures the uncertainty in the outcome of a random variable. If the $H(Y) = 0$ it means that the outcome is certain; if $H(Y) = log(n)$ it means that we have complete uncertainty and all the outcomes have equal probability.

In a probabilistic grammar the sample space is the set of all generated strings leading to the next definition of grammar entropy according to (Soule 1974).

Definition 8 (Grammar entropy) The *entropy of a probabilistic grammar*, $H(G_p)$, is defined as $H(G_p) = -\sum_{w \in L(G_p)} p(w) log\, p(w)$.

The intuition is that if the entropy of a probabilistic grammar is close to zero then there should be a small set of strings with high probability; if the entropy is high then there should be a large number of strings with similar (and low) probability.

The analogy between a probabilistic regular grammar and a finite Markov chain is very useful for the computation of statistical properties of the language such as its entropy. In fact, the derivation of a string by a probabilistic grammar is a Markovian stochastic process since the choice of the next production depends only on the unique nonterminal symbol present in the sentential form.

We have considered two hypotheses when modelling a Hypertext grammar as a Markov chain by adding a further production $F \rightarrow S$ to the grammar: (i) if $p(F \rightarrow S) = 0$ the chain is absorbing in state F; (ii) if $p(F \rightarrow S) = 1$ the chain is irreducible (Feller 1971). Note that there is always a path to F.

In (i) a HPG corresponds to an *absorbing Markov chain* where the state F is a unique absorbing state. The probabilities of the productions of type $S \rightarrow a_i A_i$ correspond to the vector of initial probabilities, π, and the probabilities of productions $A_i \rightarrow a_j A_j$ characterise the square transition matrix A of order $|V - \{S, F\}|$; η is a vector with the same order as A and where $\eta_i = H(A_i)$ is the entropy of the random choice among all the productions which have nonterminal A_i in its left side, including productions of type $A_i \rightarrow F$. In (Soule 1974) the entropy of probabilistic grammar with a unique absorbing state is defined as the vector $H(G_p) = (I - A)^{-1}\eta$.

In (ii) the chain is the same but $p(F \rightarrow S) = 1$ makes the chain irreducible. (Cover & Thomas 1991) defines the *entropy rate* of a irreducible Markov chain with a stationary distribution vector μ and transition matrix A as $H_r(G_p) = -\sum_{ij} \mu_i A_{ij} log A_{ij}$, where the entropy rate is the average entropy per step in a string derivation.

In this paper we consider the entropy of a HPG as: (i) $H(G_{hp}) = H(\pi) + \pi(I - A)^{-1}\eta$ which corresponds to the entropy vector defined by Soule averaged by the vector of initial probabilities plus the entropy of vector π or (ii) $H_r(G_{hp}) = H(\pi) + (-\sum_{ij} \pi_i A_{ij} log\ A_{ij})$ which is the Cover definition plus the entropy of vector π which is taken as an estimator of μ. We include $H(\pi)$ in the definitions in order to take into account the randomness of the choice of the initial page. We use the vector of initial probabilities as an estimator of the stationary vector, since π is proportional to the number of times each state was visited assuming that the Markov chain is aperiodic.

3 Data Mining Algorithms

3.1 Building the grammar from Log Data

The Hypertext grammar that characterises the user interaction within a Web site is inferred from the log data. A log file can be seen as an ordered set of page requests from which we can infer the user navigation sessions. A session is a sequence of requests such that no two consecutive requests are separated by more than X minutes, where X is a parameter of the model. A session corresponds to a trail in the Hypertext database.

We let T be the set of Hypertext database trails, $|T|$ be the number of trails, T_i be the i^{th} trail in the set and $T_i[j]$ be the j^{th} state of trail i. We also let d be the history depth of the grammar, and $StateSet$ be an entity that represents the set of grammar states; the function $StateSet.update()$ updates $StateSet$ with the last inferred state. A $State$ corresponds to d consecutive page requests $State[0], \ldots, State[d]$, (see Section 2.5); the function $shiftLeft()$ performs the assignment $State[i] := State[i+1]$, for $i = 1, \ldots, d-1$. In addition, $Productions$ is the set of all grammar productions and the function $Productions.update()$ updates the set with the last inferred production. We now give the algorithm used to build a HPG.

Algorithm 1 (Build_Grammar (T, d))

```
begin
  for i = 1 to |T|
    for k = 1 to d do State[k] = T_i[k] end for
    StateSet.update(State);
    for j = d + 1 to |T_i|
      LeftState = State;
      State.shiftLeft();
      State[d] = T_i[j];
      StateSet.update(State);
      Productions.update(LeftState, State);
    end for
  end for
end.
```

3.2 Depth-First Search Algorithm

The algorithm proposed to mine for the grammar strings with probability above the confidence and support thresholds is a special case of the directed graph Depth-First Search (DFS) (Tarjan 1972) and is similar to that presented in (Borges & Levene 1998). This algorithm essentially performs an exhaustive search of all the strings with the required characteristics.

We let π be the vector of initial probabilities, $trail$ be the string being generated (sentential form) and $\mathcal{C}_{trail}$ be its confidence; $trail \Rightarrow trail'$ is the one-step derivation of $trail'$ from $trail$. We also let Y be the unique non-terminal symbol in the sentential form, $(Y \rightarrow a_i A_i \in P)$ be all those productions whose left side is Y, and $p(Y \rightarrow a_i A_i)$ is its probability. Finally, we let $Rules$ be the set of mined rules.

Algorithm 2.1 (Modified_DFS(λ,θ))

begin
  for $i = 1$ to $|V| - 2$
    if $\pi_i > \theta$ **then**;
      Explore(i, π_i);
    end if
  end for
end.

Algorithm 2.2 (Explore($trail$, $\mathcal{C}_{trail}$))

begin
  for each $(Y \rightarrow a_i A_i \in P)$
    if $(p(Y \rightarrow a_i A_i) \times \mathcal{C}_{trail} > \lambda)$ **then**
      $trail \Rightarrow trail'$
      Explore($trail'$, $\mathcal{C}_{trail'} \times p(Y \rightarrow a_i A_i)$);
    else
      $Rules := Rules \cup trail$;
    end if
  end for
end.

4 Experimental Evaluation

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

4.1 Experiments with Random Data

Algorithm 3 presents our method of generating the random data. The parameters needed to be specified are: the number of states in the Hypertext grammar (N), the average number of out-going links per state (O), and the probability distribution and respective

Algorithm 3 (Random_Data(N,(O)))

1. create a complete directed graph with N states;
2. traverse each of the $N(N-1)$ arcs and for each one determine if it will be in the grammar or not with a probability of $\frac{O}{N}$;
3. for each existing arc assign its weight according to the chosen probability distribution (the arc's weight corresponds to the number of times the corresponding hypertext link was traversed);
4. for each state assign the number, t, of times a session terminated in it according to the probability distribution;
5. for each state such that $t = 0$ verify if there is a path to a state with $t \neq 0$, i.e., if there is a path to the final node. If not then generate another value $t > 0$.

parameter of the link's weights.

In the experiments with random data we had two main objectives: (i) to evaluate the performance of the algorithms and (ii) to evaluate how the grammar entropy could be used as estimators of rules characteristics. The experiments were conducted for various grammar configurations. Since the model in (Borges & Levene 1998) was not sensitive to variations in the average density, $\frac{O}{N}$, of the graph and was also not sensitive to the probability distribution of the weights of the links we decided to maintain these parameters fixed in this experiments. The size varied between 100 and 4000 states, the confidence threshold values varied between 0.1 and 0.5, and the support was fixed to $\frac{1}{N}$ for each grammar size. For each configuration more than 150 runs were performed.

The results show that for a given confidence the average number of iterations (for the 150 runs) varies linearly with the number of grammar states (the grammar size), see Figure 5; the CPU time follows a similar trend. Figure 6 shows how the average number

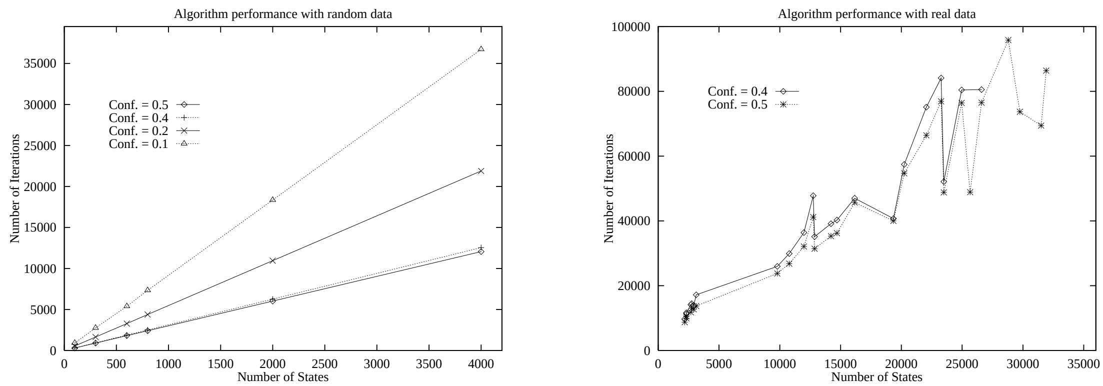

Figure 5: Variation of the number of iterations with the grammar size.

rules obtained (again for the 150 runs) varies with the size of the grammar.

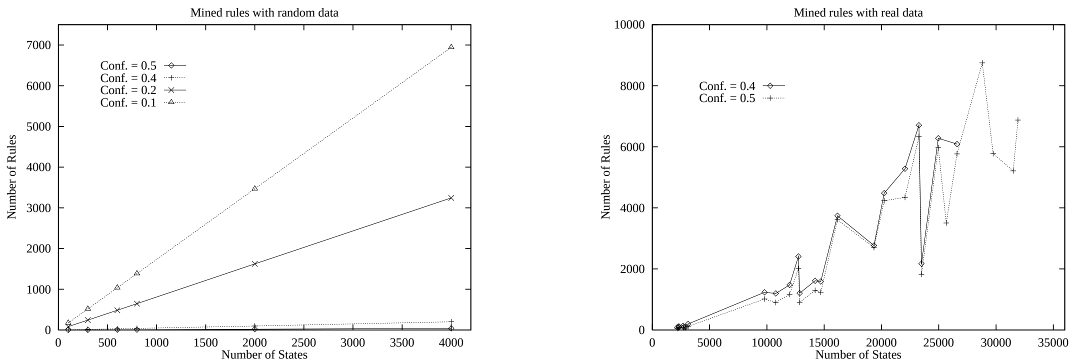

Figure 6: Variation of the number of rules with the grammar size.

In order to verify the quality of both definitions of grammar entropy, see Section 2.6, as estimators of the characteristics of rules mined we calculated for a given grammar size and confidence thresholds 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.

In order to test to what extent both the confidence and support thresholds were affecting the entropy of the results we measured the entropy parameters in a grammar inferred from the set of mined rules, the *posterior grammar*, G'_p. Figure 7 shows the results for grammars with 800 and 2000 states where PER is the Cover entropy of the posterior grammar, PSE is the Soule entropy of the posterior grammar, and ER is the Cover entropy of the original grammar but using the vector of initial probabilities of the posterior grammar, π', as the estimator for the stationary vector in the original grammar. The results show that PER is a better estimator of the number of rules than PSE. Moreover, PSE has the drawback of needing to invert a matrix which is compu-

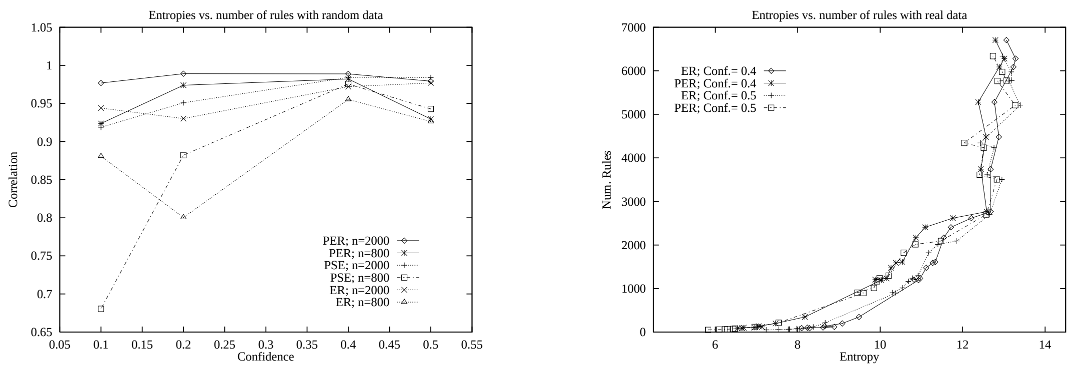

Figure 7: The grammar entropy versus the number of rules.

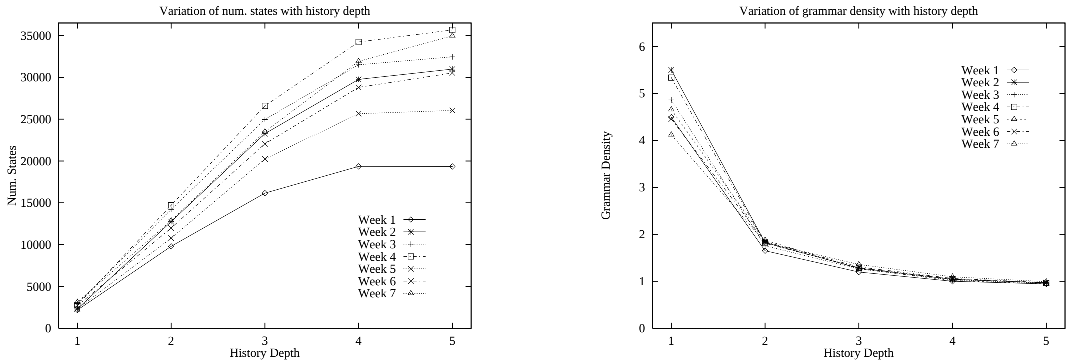

Figure 8: Variation of the grammar characteristics with the history depth.

tationally demanding. The results regarding ER suggest that the original grammar is a good estimator for the number of rules to mine provided we find a good estimator for the stationary vector. The experimental results also show that the entropy is not an estimator of the number of iterations or of the average rule length.

4.2 Real Data

In this section we describe the experiments performed with real log files. The data was obtained from the authors of (Perkowitz & Etzioni 1998) and can be found in `http://www.cs.washington.edu/research/adaptive/download.html`.

The log data contains two months of usage from the site `http://www.hyperreal.-org/music/machines/`. It should be noticed that: (i) the data was collected while the cache was disabled and (ii) 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 Hypertext grammar for several values of the history depth. One of the weeks was discarded because it presented characteristics significantly different from all the other seven. This last fact indicates that the data should be cleaned; however, because 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 for the other weeks to use the data as it was.

Figure 8 shows the variation of both the number of states (grammar size) and the grammar density with the history depth. The results show that the size of the N-gram model increases at a rate much slower than the worst case and that the grammar size

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

Rule 1	Conf.= 0.43	Avg. Num. Traversals = 44.5
/ /ecards/ /ecards/cards/ /ecards/cards/96301		

Rule 2	Conf.= 0.53	Avg. Num. Traversals = 11
/ /manufacturers/ /manufacturers/Music-and-More/ /manufacturers/Music-and-More/VF11/ /manufacturers/Music-and-More/VF11/info/VF11.review		

Rule 3	Conf.= 0.56	Avg. Num. Traversals = 64
/categories/drum-machines/samples/ /categories/drum-machines/samples/deepsky_kicks/ /categories/drum-machines/samples/deepsky_kicks/README/ /categories/drum-machines/samples/deepsky_kicks/ /categories/drum-machines/samples/		

Figure 9: Example of mined rules.

stabilises for history values of order 5. Figure 8 also shows that the grammar density decreases as the history depth increased, as expected. Figure 7 shows how the number of mined rules in the real data relate to the entropy in the posterior grammar; a similar pattern was obtained for the random data. Note that while in the random data plot each point represents the correlation between the grammar entropy and the number of rules for a specific grammar topology (for 150 runs), the plot of real data gives the overall variation of the number of rules with the grammar entropy and each point corresponds to a single run for a specific topology. Our experiments provide strong evidence that while the overall variation presents a regular but non-linear trend, for a specific grammar topology there is a strong linear correlation between the number of rules and the grammar entropy.

Figure 9 gives some examples of the rules mined from the real data as well as the top-ten pages in the Web site, which correspond to the states with the highest initial probability. In Figure 5 it can be seen that the algorithm's performance with real data follows a linear trend as does the algorithm with random data. Figure 6 shows the number of rules mined in the real grammars confirming the results obtained with the random data and showing that although the average pattern is linear there is a significant variation in the results; therefore, the grammar entropy has an important role as the estimator of the number of rules for a specific grammar.

5 Related Work

The use of data mining techniques to analyse log data was first proposed by (Chen, Park, & Yu 1998) and (Yan *et al.* 1996). In (Chen, Park, & Yu 1998) a technique for mining of user access navigation patterns is proposed. The log data is converted into a form amenable by the existing data mining techniques and two algorithms are proposed to deal with the specificity of the rules in this context. (Yan *et al.* 1996) propose a method in which for each user session, inferred from log data, a vector is created that stores the number of visits in each page. A algorithm is then applied to find clusters of similar vectors.

In (Perkowitz & Etzioni 1997) the authors formally challenged the AI community to use the the log data to create adaptive Web sites and in (Perkowitz & Etzioni 1998) they present their own approach, which consists in using log data to automatically create index pages, i.e., pages containing collections of links which the user navigation behaviour suggests are related.

In our previous work, (Borges & Levene 1998), we propose to model the log data as directed graph with the arcs weights being probabilities that reflect the user interaction with the site and we generalised the association rule concept.

(Schechter, Krishnan, & Smith 1998) proposes the use of log data to predict the next URL to be requested by a user so the server can generate dynamic content in advance, and reduce the latency of the request. 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 (Spiliopoulou & Faulstich 1998) the authors propose a log data mining system composed of an aggregation module and a mining module. The aggregation module prepares the data for mining by constructing a tree, where trails with the same prefix are merged, and in which the mining is performed by a human expert using a mining query language which allows him to specify the characteristics of the required patterns. Finally, (Zaïane, Xin, & Han 1998) propose the use of data warehousing and data mining techniques to analyse Web records and (Cooley, Mobasher, & Srivastava 1999) study cleaning and preparation techniques to convert log data into user navigation sessions in a form amenable by the existing data mining techniques.

6 Concluding Remarks and Future Work

We have proposed a model of Hypertext to capture user navigation preferences when navigating through the Web. We claim that our model presents the advantage of being self contained, coherent, and based on the well established work in probabilistic grammars which provides it with a sound foundation for future enhancements of the model.

The set of user navigation sessions is modelled as a Hypertext probabilistic grammar, and the 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 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 model. 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 Web's content.

We are currently experimenting with a randomised algorithm for mining rules as a more efficient alternative to the DFS algorithm. We intend to study further the use of information theoretic measures and their relation to the properties of the model. We are also studying the effect of adding relevance measures to the Web pages in order to assist the user in locating useful information.

References

Borges, J., and Levene, M. 1998. Mining association rules in hypertext databases. In *Proc. of the fourth Int. Conf. on Knowledge Discovery and Data Mining*, 149–153.

Charniak, E. 1996. *Statistical Language Learning.* The MIT Press.

Chen, M.-S.; Park, J. S.; and Yu, P. S. 1998. Efficient data mining for traversal patterns. *IEEE Trans. on Knowledge and Data Eng.* 10(2):209–221.

Cooley, R.; Mobasher, B.; and Srivastava, J. 1999. Data preparation for mining world wide web browsing patterns. *Knowledge and Information Systems* 1(1).

Cover, T. M., and Thomas, J. A. 1991. *Elements of Information Theory.* John Wiley & Sons.

Craven, M.; DiPasquo, D.; Freitag, D.; McCallum, A.; Mitchell, T.; Nigam, K.; and Slattery, S. 1998. Learning to extract symbolic knowledge from the world wide web. In *Proc. of the 15th National Conf. on Artificial Intelligence.*

Feller, W. 1971. *An Introduction to Probability Theory and Its Applications.* John Wiley & Sons, second edition.

Hopcroft, J. E., and Ullman, J. D. 1979. *Introduction to Automata Theory, Languages, and Computation.* Addison-Wesley.

Levene, M., and Loizou, G. 1999. A probabilistic approach to navigation in hypertext. *Information Sciences* 114:165–186.

Nielsen, J. 1990. *Hypertext & Hypermedia.* Academic Press.

Perkowitz, M., and Etzioni, O. 1997. Adaptive web sites: an AI challenge. In *Proc. of Int. Joint Conf. on Artificial Intelligence.*

Perkowitz, M., and Etzioni, O. 1998. Adaptive sites: Automatically synthesizing web pages. In *Proc. of the 15th National Conf. on Artificial Intelligence.*

Rosenfeld, L., and Morville, P. 1998. *Information Architecture for the World Wide Web.* O'Reilly.

Schechter, S.; Krishnan, M.; and Smith, M. D. 1998. Using path profiles to predict http requests. *Computer Networks and ISDN Systems* 30:457–467.

Soule, S. 1974. Entropies of probabilistic grammars. *Information and Control* 25:57–74.

Spiliopoulou, M., and Faulstich, L. C. 1998. WUM: A web utilization miner. Int. Workshop on the Web and Databases.

Stout, R. 1997. *Web Site Stats: tracking hits and analyzing traffic.* Osborne McGraw-Hill.

Tarjan, R. 1972. Depth-first search and linear graph algorithms. *SIAM Journal on Computing* 1(2):146–160.

Wetherell, C. S. 1980. Probabilistic languages: A review and some open questions. *Computing Surveys* 12(4):361–379.

Yan, T. W.; Jacobsen, M.; Garcia-Molina, H.; and Dayal, U. 1996. From user access patterns to dynamic hypertext linking. In *Proc. of the 5th Int. World Wide Web Conference*, 1007–1014.

Zaïane, O. R.; Xin, M.; and Han, J. 1998. Discovering web access patterns and trends by applying olap and data mining technology on web logs. In *Proc. Advances in Digital Libraries Conf.*, 12–29.