A Fine Grained Heuristic to Capture Web 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

In previous work we have proposed a statistical model to capture the user behaviour when browsing the web. The user navigation information, obtained from web logs, is modelled as a hypertext probabilistic grammar (HPG) which is within the class of regular probabilistic grammars. The set of highest probability strings generated by the grammar corresponds to the user preferred navigation trails. We have previously conducted experiments with a Breadth-First Search algorithm (BFS) to perform the exhaustive computation of all the strings with probability above a specified cut-point, which we call the *rules*. Although the algorithm's running time varies linearly with the number of grammar states, it has the drawbacks of returning a large number of rules when the cut-point is small and a small set of very short rules when the cut-point is high.

In this work, we present a new heuristic that implements an iterative deepening search wherein the set of rules is incrementally augmented by first exploring trails with high probability. A stopping parameter is provided which measures the distance between the current rule-set and its corresponding maximal set obtained by the BFS algorithm. When the stopping parameter takes the value zero the heuristic corresponds to the BFS algorithm and as the parameter takes values closer to one the number of rules obtained decreases accordingly.

Experiments were conducted with both real and synthetic data and the results show that for a given cut-point the number of rules induced increases smoothly with the decrease of the stopping criterion. Therefore, by setting the value of the stopping criterion the analyst can determine the number and quality of rules to be induced; the quality of a rule is measured by both its length and probability.

Keywords

Usage mining, navigation patterns, hypertext probabilistic grammars

1. INTRODUCTION

The rapid increase in the size of the web in recent years has transformed it into the largest resource of available online data. As this trend continues so do the difficulties the user faces in trying to find relevant and interesting information on the web. Moreover, navigation in large information structures often causes the user to become disoriented [10; 13], that is, when following links the user loses the context in which he is browsing and needs assistance in finding his way.

A research direction which deals with this problem and is being pursued by an increasing number of researchers is web usage mining [19]. This research field focuses on techniques which use log data to find patterns in the user navigation sessions. The navigation sessions take the form of sequences of links followed by the user, which we call *trails*, that characterise his web navigation behaviour. Understanding user navigation behaviour is an essential step in the process of customising web sites to the user's needs either by improving its static structure or/and by providing adaptive web pages [14]. Moreover, since 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 [20]; 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 the information available in such 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 or clustering techniques, are invoked, see for example [9] and [22]. In the second approach techniques are developed which can be invoked directly on the log data, see for example [1], [2], [18] and [17].

The work reported herein is part of a ongoing research with the long term goal of specifying a set of techniques to identify relevant web trails [1; 12; 23; 2; 3]. In [2] we presented a new model for handling the problem of mining log data wherein the web is modelled as a regular grammar whose states correspond to web pages and production rules to hyperlinks. The user navigation sessions are incorporated into the model in order to build a *hypertext probabilistic grammar* (or simply a HPG) and data mining techniques are used to find the higher probability strings which correspond to the user's preferred navigation trails. In [3] we have extended the HPG analysis capabilities by devising a heuristic to induce a set of strings whose average link probability is above a specified threshold. The HPG model provides a simple and sound tool to summarise the user interaction with the web and which is potentially useful both to the web site designer and to the individual user.

On the one hand, by using a HPG to analyse server logs, the web site designer can benefit from having a better understanding of the users' browsing behaviour which is characterised by the set of most popular trails. Understanding the user preferences is a step forward to provide adaptive web pages or to improve the site structure according to the business objectives. Such objectives can be, for example: to personalise web pages, to increase the average time a user spends in the site, or to introduce new pages in places which make them highly visible. To increase the average time a user stays on the site links can be created between pages of popular trails. A new product can be given good exposure by placing links to its pages from a popular trail that includes related products. Knowing the popular trails can also help to identify the most common points where users terminate their sessions so that the content on those points can be improved.

On the other hand, a HPG can be implemented as a browser plug-in to incrementally store and analyse the user's individual navigation history. Such a HPG would be a representation of the user's knowledge of the web and could act as a memory aid, be analysed in order to infer the user preferred trails, or work as a prediction tool to prefetch in advance pages the user may be interested in. By having a model of his individual knowledge of the web a user would be able to compare and/or exchange his model with those of his peers and, for example, identify peers' preferred trails which are unknown to him.

In [2] we described a Breadth-First Search (BFS) algorithm to find all the trails having confidence and support above a user specified threshold. Although such algorithm is very efficient it yields an unmanageable number of rules if the confidence is set too low and a small set of very short rules if the confidence is set too high. A large rule-set limits the ability of running the algorithm in main memory, with the consequent degradation of its performance, and also demands the existence of large storage space and database access time. This is particularly important when the analyst is interactively experimenting with various different model configurations. The referred drawbacks of the BFS led us to investigate heuristics which allow us to compute a subset of the rule-set such that, for a given confidence value, we are able to control both its size and rule length. The approach reported herein, which we call the *fine-grained* heuristic, implements an iterative deepening search wherein the set of rules is incrementally augmented by first exploring trails with high probability. A stopping parameter is provided which measures the distance between the current rule-set and its corresponding maximal set obtained by the BFS algorithm.

We note that the HPG model corresponds to a kth order Markov model, see [2]. Recently, several authors have also been proposing the use of Markov models to the study of users web navigation behaviour. In [16] the author proposed a system which is used to demonstrate the utility of Markov chain models in link prediction and path analysis on the web. Experimental results are reported which show that a Markov model can be useful both in the prediction of http requests and in the prediction of the next link to be requested. In [4] the authors propose a methodology for clustering individuals given the collections of their user navigation sessions and in [5] a methodology for visualising the navigation patterns characterising each cluster is presented. The navigation behaviour of the users is represented by a Markov model in which the pages are classified into predefined categories. As opposed to the HPG model, this model does not analyse the log data in its finest level of detail. However, the proposed techniques are potentially useful for a log data preprocessing stage in order to build a HPG representing the navigation behaviour of a group of users with similar interests. In addition, the HPG is incremental, extensible and provided with efficient algorithms to analyse its navigation trails.

Section 2 presents an overview of HPGs, while Section 3 presents the proposed heuristic and details the results of performed experiments. Finally, in Section 4 we give our concluding remarks.

2. THE HPG MODEL

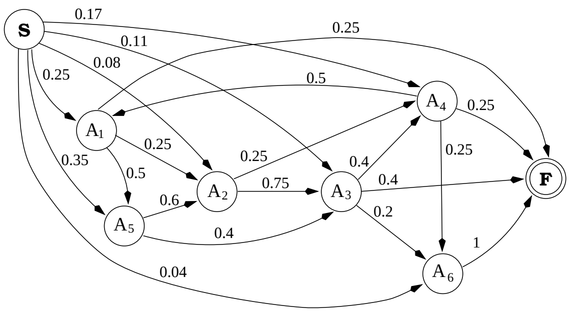
A log file consists in a per-user ordered set of web page requests from which it is possible to infer user navigation sessions. Since it is expected that a user visits a web site more than once, a user navigation session is usually defined as a sequence of page requests from the same user such that no two consecutive requests are separated by more than X minutes, where X is a given parameter. In [6] the authors proposed for X the value of 25.5 minutes, which corresponds to 1.5 standard deviations of the average time between user interface requests. Since then, many authors have adopted the value of 30 minutes for X. Techniques to infer user navigation sessions from log data are described in [11].

A collection of user navigation sessions is modelled as a *hypertext probabilistic language* [12] generated by a *hypertext probabilistic grammar* (or simply HPG) [2] which is a proper subclass of probabilistic regular grammars [21]. A HPG is a probabilistic regular grammar which has a one-to-one mapping between the sets of nonterminal and terminal symbols. Each nonterminal symbol corresponds to a web page and a production rule corresponds to a link between pages. Moreover, there are two additional artificial states, S and F, which represent the start and finish states of the navigation sessions. The probability of a grammar string is given by the product of the probabilities of the productions used in its derivation. We call the productions with S on its left-hand side *start productions* and we call the productions corresponding to links between pages *transitive productions*.

From the collection of user navigation 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. We define α as a parameter which attaches the desired weight of a state corresponding to the first page browsed in a user navigation session. If $\alpha = 0$ only states which were the first in an actual session have probability greater than zero of being in a start production. In this case, only those strings which start with a state that was the first in a session are induced by the grammar. In the other extreme, if $\alpha = 1$ the probability of a start production is proportional to the overall number of times the corresponding state was visited. In this case, the destination node of a production with higher probability corresponds to a state that was visited more often. The parameter α can take any value between 0 and 1, providing a balance between the two scenarios described above. As such, α gives the analyst the ability to tune the model for the search of different types of patterns in the user navigation sessions. Finally, the probability of a transitive production is assigned in such a way that it is proportional to the frequency with which the corresponding link was traversed.

ID	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	$A_4 \rightarrow A_1 \rightarrow A_5 \rightarrow A_3$

(a)



(b)

Figure 1: A set of trails and the corresponding HPG for $N = 1$ and $\alpha = 0.5$.

Figure 1 (a) gives an example of a collection of user sessions and Figure 1 (b) gives the corresponding HPG, represented by a transition diagram, for $\alpha = 0.5$. A HPG is a four-tuple $< V, \Sigma, S, P >$ where $V = \{S, A_1, A_2, A_3, A_4, A_5, A_6, F\}$ is the set of nonterminal symbols, $\Sigma = \{a_1, a_2, a_3, a_4, a_5, a_6\}$ is the set of terminal symbols, S is the unique start symbol, and P is the set of production rules. The production rules of the HPG in the example are given in Figure 2. The collection of navigation sessions in the example contains a total of 24 page requests, wherein, for example, state A_1 was visited 4 times, 2 of which as the first state in a user session. Therefore, for $\alpha = 0.5$ we have that the probability of its start production is $p(S \rightarrow a_1 A_1) = (0.5 \cdot 4)/24 + (0.5 \cdot 2)/6 = 0.25$. Moreover, page A_4 was visited 4 times, 1 of which as the last page in a session, once on the way to page A_6, and twice on the way to page A_1. Therefore, the probability of the transitive productions with A_4 on their left-hand side are $p(A_4 \rightarrow a_1 A_1) = 2/4$, $p(A_4 \rightarrow a_6 A_6) = 1/4$, and $p(A_4 \rightarrow F) = 1/4$.

start prod.		transitive prod.		final prod.	
$S \rightarrow a_1 A_1$	0.25	$A_1 \rightarrow a_2 A_2$	0.25	$A_1 \rightarrow F$	0.25
$S \rightarrow a_2 A_2$	0.08	$A_1 \rightarrow a_5 A_5$	0.5	$A_3 \rightarrow F$	0.4
$S \rightarrow a_3 A_3$	0.11	$A_2 \rightarrow a_3 A_3$	0.75	$A_4 \rightarrow F$	0.25
$S \rightarrow a_4 A_4$	0.17	$A_2 \rightarrow a_4 A_4$	0.25	$A_6 \rightarrow F$	1.0
$S \rightarrow a_5 A_5$	0.35	$A_3 \rightarrow a_4 A_4$	0.4		
$S \rightarrow a_6 A_6$	0.04	$A_3 \rightarrow a_6 A_6$	0.2		
		$A_4 \rightarrow a_1 A_1$	0.5		
		$A_4 \rightarrow a_6 A_6$	0.25		
		$A_5 \rightarrow a_2 A_2$	0.6		
		$A_5 \rightarrow a_3 A_3$	0.4		

Figure 2: The hypertext probabilistic grammar corresponding to the collection of trails in the example for $\alpha = 0.5$.

The strings generated by the grammar correspond to the user navigation trails, and the aim is to identify the subset of these strings, which correspond to the rules, that best characterise the user behaviour when visiting the site. (From now on we use the terms trail and string interchangeably.) A trail is included in the grammar's language if its derivation probability is above the *cut-point*, λ. The language cut-point is composed of two distinct thresholds $\lambda = \theta \cdot \delta$, where $\theta \in (0, 1)$ is the *support* threshold and $\delta \in (0, 1)$ the *confidence* threshold. The intuition behind the support is that it is the factor of the cut-point responsible for pruning out the strings whose first derivation step has low probability, corresponding to a subset of the hypertext system rarely visited. Note that there can be up to n start productions, where n is the number of grammar states; therefore, the probabilities of these productions are of a much smaller order than those of the transitive productions suggesting that the support value should take into account the number of states. In the experiments reported in this work we fixed the support at $\theta = 1/n$. The confidence is the factor of the cut-point responsible for pruning out strings whose derivation contains transitive productions with small probabilities. In similarity to the standard association rule mining the support threshold has the role of restricting the mining for rules to the strings which start at states that appear frequently in the navigation sessions. Moreover, the confidence threshold has the role of pruning out the strings with low probability that passed the support test. The values of the support and confidence thresholds give the user control over the number, the length and the probability of the trails to be included in the rule-set.

We note that the proposed model makes the assumption that the choice of the next page to visit depends only on the current page in the user's browser and that the HPG corresponds to a Markov chain. It follows that there can be strings in the grammar's language corresponding to trails that were not traversed as a whole by a user. Although this assumption is somewhat controversial it is severely limiting to model only the exact user sessions since the probability of long trails being followed several times in exactly the same way is low and, in addition the prediction is harder. Moreover, when a user is inspecting a web page and deciding which link to follow next only few if any of all the previously visited pages affect his decision. Consider, for example, a user browsing an online supermarket that has to decide which link to follow while inspecting a page with special offers. His decision will probably depend more on the quality of the offers themselves then on all the pages he visited beforehand. On the other hand, the probability of choosing a link is not completely independent of the browsing history. Therefore, a model that blends these two modes of behaviour provides a better representation of the user's navigation patterns. Thus, in [2] we make use of the Ngram concept [7], where $N, N \geq 1$, determines the user memory when navigating the web, implying that when visiting a page only the N previously visited pages influence the link the user will choose to follow next. In an Ngram each of the states correspond to a sequence of N pages visited leading to a tradeoff between the model accuracy and its complexity, since when the order of the model increases so does its size, measured by the number of states. The decision of which N to choose can be assisted by statistical techniques, such as detailed in [8].

In [2] we reported the results of the experiments with an algorithm which finds all grammar strings with probability above the specified cut-point. Although the algorithm, which is a modified breadth-first search, is very efficient it has the drawback of potentially returning a very large number of rules for small values of the cut-point and, in addition, the rules are too short when the cut-point is close to one. While Figure 2 shows the variation of the number of rules with the confidence threshold Figure 4 shows the variation of both the average (ARL) and maximum (MRL) rule length with the same threshold. In can be seen that in order to obtain long rules the threshold has to be set with a low value and that leads to a very large number of rules. Although it is possible to rank the rules in a large rule-set by their length and probability, or by some other criteria, in order to identify the best rules, the manipulation of a large rule-set limits the algorithm's ability to run in main memory. Moreover, the database access time needed to store a large rule-set can also be a severe limitation when the analyst is interactively performing consecutive tests with several different model configurations. These problems led us to the study of the heuristic presented in the next section which allows us to compute a high probability subset of the rule-set while being able to control both its size and the length of its rules.

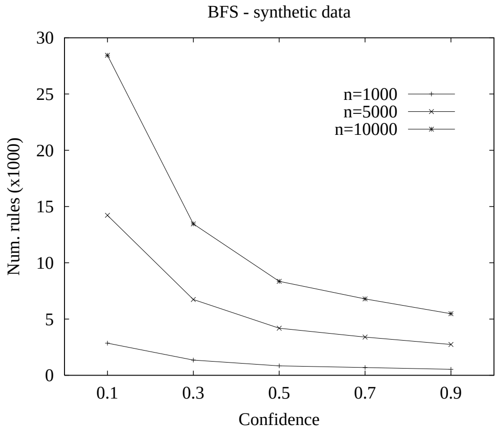


Figure 3: Characteristics of the rule-set induced with the BFS algorithm.

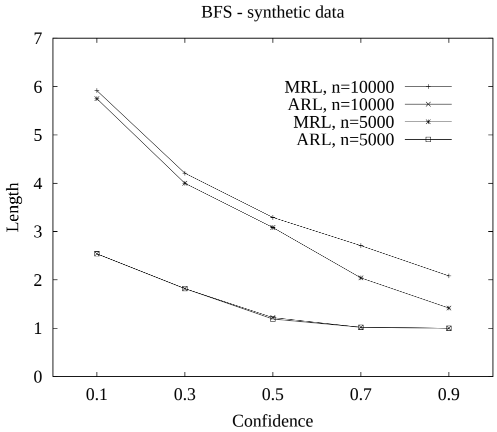


Figure 4: Characteristics of the rule-set induced with the BFS algorithm.

3. FINE-GRAINED HEURISTIC

3.1 Iterative Deepening Heuristic

The idea behind this heuristic is to implement an *iterative deepening* search (or simply ID) wherein the rule-set is incrementally built until it has the desired characteristics. To that effect, a *stopping parameter* is provided which measures the distance between the rule-set being built and the *maximal rule-set*. The maximal rule-set is the set of rules induced by the BFS algorithm, which includes all trails with probability above the cut-point and is used as reference for the heuristic's results. The analyst specifies the value of a *stopping criterion*, which works as a threshold for the stopping parameter. The search for rules stops when the value of the stopping parameter surpasses the value of the stopping criterion. By setting the value of the stopping criterion the analyst is given control over the number and length of the rules obtained; the rule-set obtained is a proper subset of the maximal set.

The ID heuristic works as follows, an *exploration tree* is built from the start state and in the first iteration only the trails with length one are explored. Having built the tree with depth one the value of the stopping parameter is computed and evaluated against the stopping criterion, specified by the analyst. If the stopping criterion is met the exploration stops, otherwise the depth of the tree is incremented by one and all the trails with length two are explored. The stopping parameter is then re-computed and compared again with the stopping criterion; the process continues until the stopping criterion is met.

When a trail is being explored every out-link from its last state is evaluated to determine if it is an *admissible expansion* or an *inadmissible expansion*. A trail expansion is admissible if will lead to a trail with probability above the cut-point and inadmissible if not. Moreover, a trail is *maximal* when it has no admissible expansions and a trail is a *candidate* when it has at least one admissible expansion.

In the sequel $\mathcal{L}^\lambda$ stands for the grammar's language with cut-point λ, and $|\mathcal{L}^\lambda|$ denotes its cardinality. The language $\mathcal{L}^\lambda$ is obtained by the BFS algorithm and includes only maximal trails, a sub-trail is implicitly defined since it is a prefix of a maximal trail. We let k be the current depth of the exploration tree, measured by the number of links, and $\mathcal{L}_k^\lambda$ be the set of trails in the exploration tree, each of length less or equal than k and with probability above the cut-point. The language $\mathcal{L}_k^\lambda$ includes both the maximal and candidate trails from exploration tree. Moreover, we let T^λ be a random variable denoting the length of a trail and $P(T^\lambda \leq k)$ be the probability of a trail in the language $\mathcal{L}^\lambda$ be no longer than k. Given $\mathcal{L}_k^\lambda$, $P(T^\lambda \leq k)$ is computed with a one-step lookahead that is performed to identify both the admissible and inadmissible expansion and consequently the maximal and candidate trails in $\mathcal{L}_k^\lambda$. The probability $P(T^\lambda \leq k)$ corresponds to the sum of the probabilities of the maximal trails and the inadmissible expansions of the candidate trails.

$P(\mathcal{L})$ stands for the sum of the probabilities of all trails in $\mathcal{L}$ where $\mathcal{L} \in \{\mathcal{L}^\lambda, \mathcal{L}_k^\lambda\}$. Note that $0 \leq P(\mathcal{L}_k^\lambda) \leq 1$ since the sum of the probabilities of a set of trails, not including sub-trails, in an exploration tree, which corresponds to a stochastic tree, is always less or equal to 1. In addition, $P(\mathcal{L}_k^\lambda) \geq P(T^\lambda \leq k)$ since while both sets include the maximal trails with length less or equal to k, $\mathcal{L}_k^\lambda$ includes the candidate trails which are prefixes (and therefore have higher probability) of their inadmissible expansions included in $T^\lambda \leq k$. Also, $P(\mathcal{L}_k^\lambda) \geq P(\mathcal{L}^\lambda)$ since every trail in $\mathcal{L}_k^\lambda$ is a proper prefix of a maximal trail in $\mathcal{L}^\lambda$. Finally, for k large enough $\mathcal{L}_k^\lambda = \mathcal{L}^\lambda$.

The ID heuristic iteratively computes $\mathcal{L}_k^\lambda$ for $k = \{1, 2, \ldots\}$ until $\mathcal{L}_k^\lambda$ is a good enough approximation of $\mathcal{L}^\lambda$. For a given k we measure how close $\mathcal{L}_k^\lambda$ is from $\mathcal{L}^\lambda$ by the value of the stopping parameter, which is defined as the remaining probability left to explore. The stopping parameter is computed with a one-step lookahead and is defined as:

$$\eta_k = P(\mathcal{L}_k^\lambda) - P(T^\lambda \leq k).$$

A better estimator for η_k can be computed with n-step lookahead, where if n is large enough $P(T^\lambda \leq n) = P(\mathcal{L}^\lambda)$; however, the use of a n-step lookahead would lead to a decrease in performance.

Note that the value of the stopping parameter is defined to be the sum of the probabilities of the admissible expansions of the candidate trails. The exploration stops when the stopping parameter is sufficiently small; we say that a number is sufficiently small if it is less than some predefined τ, $\tau > 0$, called the *stopping criterion*. At the end of the exploration the rule-set is given by $\mathcal{L}_k^\lambda$ since both the maximal and the candidate trails are considered rules when the stopping criterion is met. Finally, note that η_k/λ is an upper bound estimator of the number of trails that can still be mined.

Figure 5 presents an example of an exploration tree to clarify the concepts introduced in this section. In the figure a pair of numbers next to a link represents the probability of the link followed by the probability of the trail beginning in state S. For $k = 2$ and $\lambda = 0.15$, before performing the lookahead, the set of candidate trails is $\{A_1A_3, A_1A_4, A_2A_5\}$. When an one-step lookahead is performed from the last state of the candidate trail A_1A_3 it is verified that both A_3A_7 and A_3A_8 are inadmissible expansions, therefore, A_1A_3 is a maximal trail. Moreover, the lookahead identifies A_4A_{10} as an inadmissible expansion of A_1A_4 and A_4A_9 as an admissible expansion of the same trail so A_1A_4 is a candidate trail. Finally, both A_5A_{11} and A_5A_{12} are identified as admissible expansions of A_2A_5 meaning that A_2A_5 is also a candidate trail.

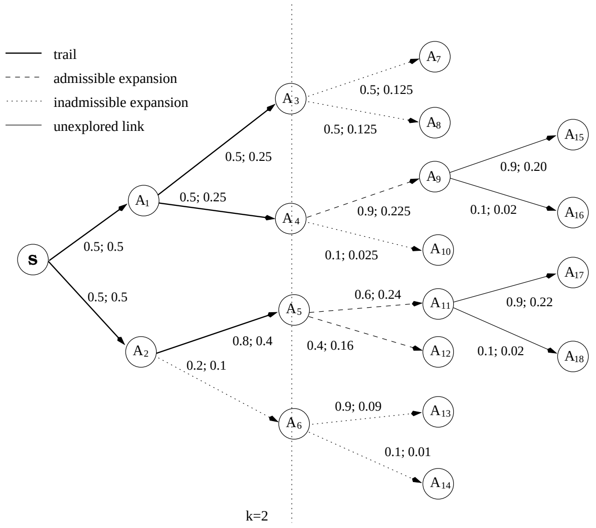


Figure 5: Example of an exploration tree obtained with the ID heuristic for $k = 2$ and $\lambda = 0.15$.

Once all the trails with length less or equal to $k = 2$ are classified we can compute the value of the stopping parameter. To that effect, $P(\mathcal{L}_2^\lambda)$ represents the probability of the set of maximal and candidate trails whose length is less or equal to 2, and has the value:

$$P(\mathcal{L}_2^\lambda) = P(A_1A_3) + P(A_1A_4) + P(A_2A_5) = 0.25 + 0.25 + 0.4 = 0.9.$$

In addition, $P(T^\lambda \leq 2)$, is the probability of the length of a maximal trail be less or equal than 2, and is given by the sum of the probabilities of both the maximal trails and the inadmissible expansions of the candidate trails.

$$P(T^\lambda \leq 2) = P(A_1A_3) + P(A_1A_4A_{10}) = 0.25 + 0.025 = 0.275.$$

Therefore, the value of the stopping parameter is:

$$\eta_2 = P(\mathcal{L}_2^\lambda) - P(T^\lambda \leq k) = 0.9 - 0.275 = 0.625$$

which corresponds to the sum of the probabilities of all admissible expansions of the candidate trails. Finally, since the language with cut-point $\lambda = 0.15$ is

$$\mathcal{L}^\lambda = \{A_1A_3,\ A_1A_4A_9A_{15},\ A_2A_5A_{11}A_{17},\ A_2A_5A_{12}\}$$

and $P(\mathcal{L}^\lambda) = 0.25 + 0.2 + 0.22 + 0.16 = 0.83$, therefore,

we can verify that:

$$\left(P(\mathcal{L}_2^\lambda) = 0.9\right) > \left(P(\mathcal{L}^\lambda) = 0.83\right) > \left(P(T^\lambda \leq 2) = 0.275\right).$$

3.2 Synthetic Data Experiments

We have conducted a set of experiments with the ID approach and although the heuristic is efficient the stopping criterion is not fine enough. Figure 6 shows the variation of the stopping parameter, η_k, with the exploration depth, and Figure 7 shows the variation of the percentage of rules obtained with the exploration depth. In both figures it is shown that almost 80% of the stopping parameter variation occurs for $k \leq 2$ meaning that it is difficult to control the size of the induced rule-set with the stopping criterion value. In fact, if the stopping criterion is set at 0.52 the tree is explored only until $k = 1$ and we obtain less than 20% of the rules, if the stopping criterion is set at 0.48 the final exploration depth is $k = 2$ and close to 80% of the rules are induced. The results are very consistent for different grammar sizes as we can see by the overlap of the lines in the plots. To overcome this problem we decided to improve the concept in a way which would lead to smaller increases in $|\mathcal{L}_k^\lambda|$; this technique is described in the next section.

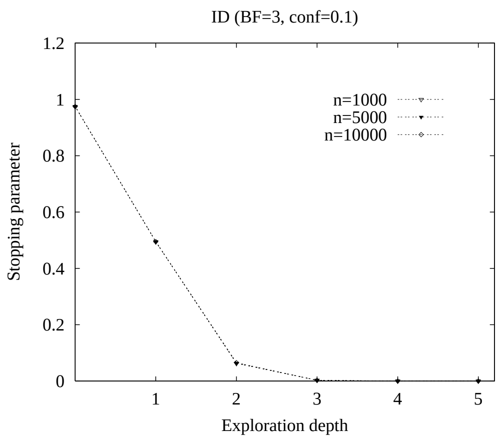


Figure 6: Variation of the stopping parameter and of the percentage of rules mined with the exploration depth.

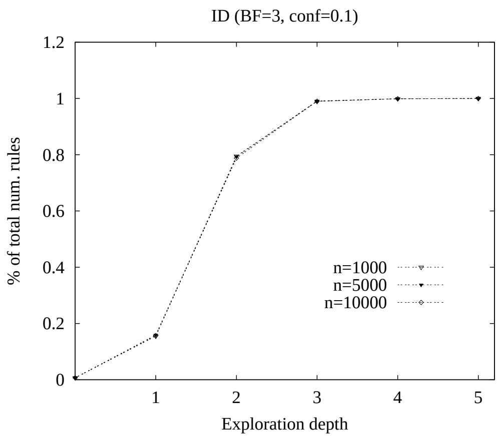


Figure 7: Variation of the stopping parameter and of the percentage of rules mined with the exploration depth.

3.3 Fine-Grained Iterative Deepening Heuristic

In this section we present the *fine-grained* iterative deepening heuristic (or simply FG) which aims at implementing a concept similar to that presented in Section 3.1 but in a way that gives more control over the number of rules obtained. As with the ID heuristic, an exploration tree with the trails evaluated is incrementally built from the start state; however, in this case the tree is characterised by its size, measured by the number of links composing the trails already explored. At each stage, a link is chosen to expand a trail from the tree and the value of the stopping parameter is computed. Thus, the size of the tree is augmented only by a single link at a time. If the value of the stopping parameter meets the stopping criterion the exploration stops, otherwise a new link is chosen to expand a trail in the tree and the process continues until the stopping criterion is met.

The FG heuristic explores the trails selectively, in contrast with the ID heuristic wherein all admissible expansions of a candidate trail is explored at the same time. When selectively exploring an admissible expansion of a candidate trail the other existing admissible expansions of the candidate will continue unexplored. As a result, there can be a prefix of the resulting trail with admissible expansions still needed to be explored. Thus, in order to formalise the description of the FG heuristic we need to introduce a new type of trail, the *candidate sub-trail*. An candidate sub-trail is prefix of a longer trail (either a maximal or a candidate trail) which has at least one admissible expansion still unexplored. And to recapitulate, a trail is maximal if it has no admissible expansions and a trail is a candidate when it has at least one admissible expansion but is not a prefix of any other candidate or maximal trail.

To clarify the candidate sub-trail concept we give an example from Figure 5. An one-step lookahead from state A_1 identifies both A_1A_3 and A_1A_4 as its admissible expansions. If A_1A_3 is chosen to be expanded first it will become a candidate trail and the trail A_1 will become a candidate sub-trail. Note that the trail A_1 has one unexplored admissible expansion, A_1A_4, and it is a sub-trail of the candidate trail A_1A_3. The fundamental difference between a candidate sub-trail and a candidate trail is that a sub-trail is always a prefix of either a maximal of a candidate trail. When the tree exploration stops, because the stopping criterion is met, the rule-set returned is composed of the maximal trails and the candidate trails; the candidate sub-trails are discarded since they are prefixes of one or more of the other trails.

In the FG heuristic we re-define k to be the total number of links that constitute the exploration tree and we let l_k^λ be a set of trails in the tree; l_k^λ includes both the maximal and the candidate trails in the exploration tree. We let $\mathcal{L}_k^\lambda$ be the family of all the l_k^λ sets, that is, $\mathcal{L}_k^\lambda = \{l_k^\lambda \mid \#l_k^\lambda = k\}$, where $\#l_k^\lambda$ represents the number of distinct links composing the trails in l_k^λ. (Note that there can be several sets of trails whose total number of links is k and that l_k^λ doesn't include the candidate sub-trails.) Moreover, we let $P(l_k^\lambda)$ be the sum of the probabilities of all trails in l_k^λ.

We let t^λ be a variable denoting the sum of the lengths of a set of trails. Moreover, for a given l_k^λ, $P(t^\lambda \leq k)$ represents the probability of a trail in l_k^λ being a maximal trail. $P(t^\lambda \leq k)$ is computed with a one-step lookahead from the candidate trails in order to identify both the maximal and candidate trails. Note that $P(t^\lambda \leq k)$ corresponds to the sum of the probabilities of the maximal trails and the inadmissible expansions of the candidate trails. In similarity to the ID heuristic, $0 \leq P(l_k^\lambda) \leq 1$, $P(l_k^\lambda) \geq P(t^\lambda \leq k)$, and $P(l_k^\lambda) \geq P(\mathcal{L}^\lambda)$.

Now, we let l_a^λ be the set of candidate sub-trails in l_k^λ and $P(l_a^\lambda)$ be the sum of the unexplored admissible expansions of the candidate sub-trails. Then, for a given k and l_k^λ the stopping parameter is defined as:

$$\eta_k = P(l_k^\lambda) - P(t^\lambda \leq k) + P(l_a^\lambda).$$

The search for rules stops when the stopping parameter takes a value sufficiently close to the stopping criterion, τ. The induced rule-set is the set of explored trails, l_k^λ, which contains both the maximal and candidate trails. Note that at each stage the value of the stopping parameter corresponds to the amount of probability left to explore, which is given by the sum of the probabilities of all the admissible expansions of both the candidate trails and the candidate sub-trails.

We propose two different criteria for deciding which trail to expand. We call the first the *best-trail criterion* as it chooses to expand the trail corresponding to the highest trail probability from among all the admissible trail expansions in l_k^λ. At each stage the trail chosen to be augmented will lead to the best possible new trail.

We call the second the *greedy criterion* as it chooses to expand the link with the highest probability from among all the admissible expansions in l_k^λ. The probabilities of the transitive productions are in general significantly greater than the probabilities of the start productions, especially when $\alpha > 0$ and all the grammar states have a positive initial probability. Therefore, for the greedy criterion we multiply the probability of a candidate link by the probability of the start production corresponding to the first state in the trail. This way the values of the candidate links are normalised in order to make them comparable to the probabilities from the start state. Otherwise, the admissible expansions of a candidate trail would, in general, have a much higher probability than the admissible expansions from the start state.

In Figure 5 the ID exploration tree for $k = 2$ and $\lambda = 0.15$ corresponds to a FG tree for $k = 5$ and the same cut-point; in fact, that is the tree obtained with the best-trail criterion. In this case we have that $l_5^\lambda = \{A_1A_3, A_1A_4, A_2A_5\}$, and

$P(l_5^\lambda) = P(A_1A_3) + P(A_1A_4) + P(A_2A_5) = 0.9$.

In addition, $P(t^\lambda \leq 5) = P(A_1A_3) + P(A_1A_4A_{10}) = 0.275$, $P(l_a^\lambda) = 0$ and the stopping parameter is

$\eta_5 = P(l_5^\lambda) - P(t^\lambda \leq 5) + P(l_a^\lambda) = 0.625.$

Note that in this particular case there are no candidate sub-trails. According to the best-trail criterion the next trail to be expanded is A_2A_5 to $A_2A_5A_{11}$, which is the one that has the highest probability among the trails corresponding to admissible expansions.

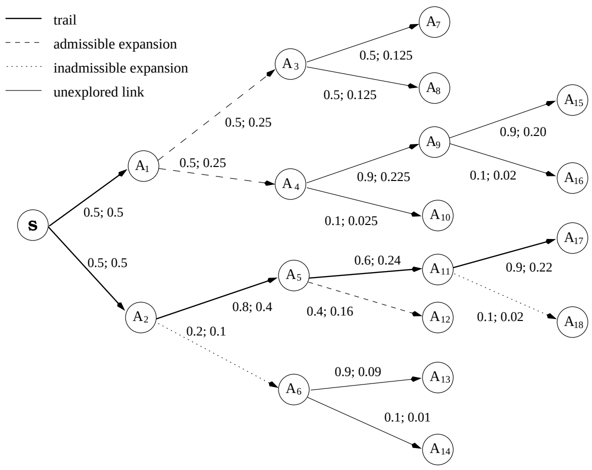


Figure 8: An example of an exploration tree obtained with the fine-grained heuristic when using the greedy criterion for $k = 5$ and $\lambda = 0.15$.

Figure 8 gives another example of an exploration tree obtained with the FG heuristic for $k = 5$ and $\lambda = 0.15$, in this case the greedy criterion was used. In this exploration tree we have that $l_5^\lambda = \{A_1, A_2A_5A_{11}A_{17}\}$ wherein A_1 is a candidate trail, and $A_2A_5A_{11}A_{17}$ is a maximal trail. The trail A_2A_5 is a candidate sub-trail since it has the unexplored admissible expansion A_5A_{12}. Therefore,

$P(l_5^\lambda) = P(A_1) + P(A_2A_5A_{11}A_{17}) = 0.72,$

$P(t^\lambda \leq 5) = P(A_2A_5A_{11}A_{17}) = 0.22,$

$P(l_a^\lambda) = P(A_2A_5A_{12}) = 0.16$, and

$\eta_5 = 0.72 - 0.22 + 0.16 = 0.66.$

According to the greedy criterion the next trail to be expanded is either A_1A_3 or A_1A_4.

When using the best-trail criterion the resulting rule-set contains the best set of trails that is possible to obtain for a given exploration tree size. Note that a maximal trail implicitly gives its set of sub-trails, where a sub-trail corresponds to a prefix of the maximal trail. Herein, we are defining the best set of trails to be the set which maximises the sum of the probabilities of both its maximal trails and its sub-trails. For example, in Figure 5 and for $k = 5$ the sum of the probabilities of all the trails in l_5^λ is:

$P(A_1)+P(A_1A_3)+P(A_1A_4)+P(A_2)+P(A_2A_5)=0.5+0.25+0.25+0.5+0.4=1.9.$

For $k = 5$ there is no other such set of trails with higher probability.

Proposition 1. For a given k, the best-trail criterion of the fine grained heuristic gives the l_k^λ whose sum of trail probabilities is maximal (including the probabilities of the sub-trails).

Proof. This property will be proved by induction.

For $k = 1$ the property holds trivially since l_k^λ contains a single trail which was chosen because it has the highest probability.

If for $k = n$ we have that l_k^λ contains the best set of trails, the property will continue to hold for $k = n + 1$ since the unexplored trail with the highest probability was added to l_k^λ in order to obtain l_{k+1}^λ. □

Figure 9 exemplifies the computation of the set of candidate trails CT, the set of candidate sub-trails CST, the set of rules RS (the maximal trails) and of the stopping parameter when the greedy version of the FG heuristic is applied to the grammar in Figure 8. The cut-point was set with the value $\lambda = 0.15$. For each member of the CST set the value inside the brackets gives the product of the probability of the trail's first link and the probability of the link corresponding to the admissible expansion, that is, the value for the greedy criterion. For a trail in the CT set each admissible expansion of the trail is represented inside brackets, the link corresponding to the expansion is given followed by its value for the greedy criterion. Finally, the symbol (*) indicates the trail chosen to be expanded in the next iteration which corresponds to that with the highest value for the greedy criterion.

Similarly, Figure 9 gives the evolution of the trail sets when the best-trail criterion is used. In this case a value inside the brackets represents the trail probability, since with this version the trail with highest probability is the one chosen to be expanded.

k	CST	CT	RS	η_k
0	$A_1(0.5)^* A_2(0.5)$	-	-	1
1	$A_2(0.5)^*$	$A_1(A_3, 0.25)(A_4, 0.25)$	-	1
2	-	$A_1(A_3, 0.25)(A_4, 0.25)$ $A_2(A_5, 0.4)^*$	-	0.9
3	-	$A_1(A_3, 0.25)(A_4, 0.25)$ $A_2A_5(A_{11}, 0.3)^*(A_{12}, 0.2)$	-	0.9
4	$A_2A_5A_{12}(0.2)$	$A_1(A_3, 0.25)(A_4, 0.25)$ $A_2A_5A_{11}(A_{17}, 0.45)^*$	-	0.9
5	$A_2A_5A_{12}(0.2)$	$A_1(A_3, 0.25)^*(A_4, 0.25)$	$A_2A_5A_{11}A_{17}$	0.66
6	$A_2A_5A_{12}(0.2)$ $A_1A_4(0.25)^*$		$A_2A_5A_{11}A_{17}$ A_1A_3	0.41
.	.	.	.	.

Figure 9: The contents of the sets of trails induced by the FG heuristic when using the greedy criterion and the cut-point $\lambda = 0.15$.

k	CST	CT	RS	η_k
0	$A_1(0.5)^*, A_2(0.5)$	-	-	1
1	$A_2(0.5)^*$	$A_1(A_3, 0.25)(A_4, 0.25)$	-	1
2	-	$A_1(A_3, 0.25)(A_4, 0.25)$ $A_2(A_5, 0.4)^*$	-	0.9
3	-	$A_1(A_3, 0.25)^*(A_4, 0.25)$ $A_2A_5(A_{11}, 0.24)(A_{12}, 0.16)$	-	0.9
4	$A_1A_4(0.25)^*$	$A_2A_5(A_{11}, 0.24)(A_{12}, 0.16)$	A_1A_3	0.65
5		$A_1A_4(A_9, 0.225)$ $A_2A_5(A_{11}, 0.24)^*(A_{12}, 0.16)$	A_1A_3	0.625
6	$A_2A_5A_{12}(0.16)$	$A_1A_4(A_9, 0.225)^*$ $A_2A_5A_{11}(A_{17}, 0.22)$	A_1A_3	0.605
.	.	.	.	.

Figure 10: The contents of the sets of trails induced by FG heuristic when using the best-trail criterion and the cut-point $\lambda = 0.15$.

We should note that the fine-grained heuristic was developed with the objective of giving the analyst as much control as possible over the number and quality of the mined rules by setting the value for the stopping criterion. When using the FG heuristic we expect the analyst to choose to explore a relatively small number of trails, otherwise he would use the BFS algorithm instead. Therefore, when devising the FG heuristic we were mainly concerned with providing a stopping criterion with tight control over the number of rules and the heuristic's performance wasn't considered a critical issue. There are, however, some simple modifications which can lead to improvements in the heuristic's performance. The first modification consists in exploring a specified number of links at a time instead of only one at a time, that is, to expand the X best trails, where X is a parameter specified by the analyst. The parameter X should be specified in a way that takes into account both the grammar size and the control the analyst wants to have over the number of rules; when X increases the control the stopping parameter has over the number of rules decreases but the performance improves. Note that when X corresponds to the number of admissible expansions the FG heuristic is equivalent to the ID heuristic.

The second modification consists in keeping separately two small sorted sets, one with the best elements from CT and the other with the best elements from CST; the size of these sets could be a parameter. In this case, the next best trail to be expanded could be found among the trails in those small sorted sets. Note that in the standard heuristic in order to find the next best trail both CST and CT sets have to be entirely traversed as it would be very demanding to keep such sets sorted. The smaller sorted sets would have to be refilled and ordered only when empty. In fact, whenever a new trail had to be inserted into one of the larger sets it would be verified if the trail was better than at least one trail in the corresponding sorted set. If the answer was positive than the new trail could be inserted or replace the worst trail in the sorted set (depending on whether or not the set was full).

3.4 Experimental Evaluation

To assess the effectiveness of the Fine-Grained heuristic experiments were conducted with both synthetic and real data. The synthetic data generation method consisted in randomly creating HPGs given a number of pages, an average number of out-links per page, and a probability distribution for the links' weights. The number of states, n, varied between 1000 and 20000 states, the confidence threshold between 0.1 and 0.9, and the support threshold was fixed with the value $1/n$ for each grammar size. For each configuration 30 runs were performed. The real log files were obtained from the authors of [15] and correspond to 2 months of log data which was divided into 8 weeks of usage data. The stopping criterion was set to vary in the range of 0.1 to 0.98.

As stated in Section 3.3, the main objective of the fine-grained heuristic is to provide tight control over the number of rules mined by setting the value of the stopping criterion. Note that if the stopping criterion takes the value 0 the rule-set induced by the heuristic is exactly the same as that given by the breadth-first search algorithm.

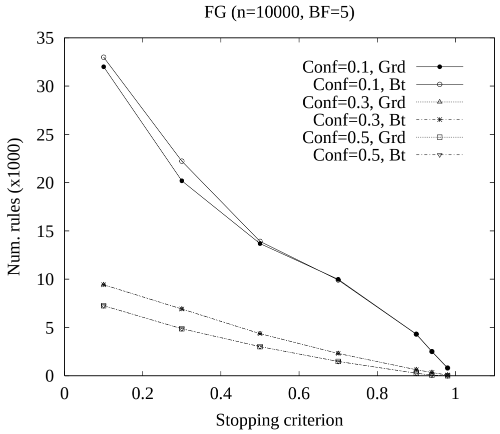


Figure 11: The variation of the number of rules with the fine-grained stopping criterion when applied to the synthetic data.

Figure 11 shows the variation of the number of rules obtained with different values of the stopping criterion for a set of randomly generated grammars with 10000 states and an average branching factor of 5 out-links per page. The results show that the number of rules induced decreases smoothly with the value of the stopping criterion, the two variable have a relationship with a negative slope whose value decreases with the confidence. Therefore, the stopping criterion gives the analyst good control over the number of rules obtained. Moreover, Figure 11 shows that only for small values of the confidence it is possible to distinguish the number of rules returned by the greedy and best-trail versions.

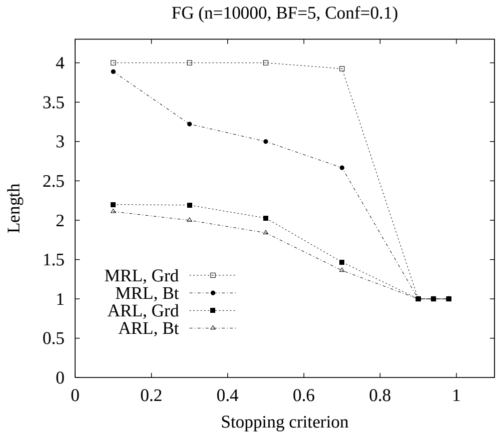


Figure 12: The variation of the length of the rules with the fine-grained stopping criterion when applied to the synthetic data.

Figure 12 shows the variation of the average rule length (ARL) and the maximal rule length (MRL) with the stopping criterion when the confidence threshold was set at 0.1. It can be seen that even for high values of the stopping criterion, which corresponds to small rule-sets, it is possible to obtain long rules especially with the greedy version. In fact, with the greedy version it is possible to obtain some rules close to the maximal length with the stopping criterion set with the value 0.70.

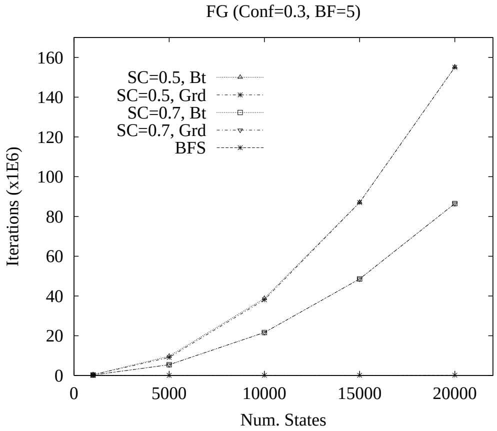


Figure 13: The performance of the fine-grained heuristic with the synthetic data.

Figure 13 gives the variation of the number of iterations with the size of the grammar for different values of the stopping criterion (SC); the performance of the BFS algorithm is also given. As expected, the results show that the FG heuristic is much more time consuming than the BFS. While the BFS algorithm exhibits a linear behaviour the results given herein suggest that the FG heuristic is polynomial. In fact, we were able to fit a quadratic function in all the given curves, always with a correlation coefficient greater than 0.999. However, as was stated in Section 3.3, the performance was not an issue when devising the heuristic and some directions for the improvement of its performance were also given. Finally, the two versions of the heuristic are indistinguishable in the number of iterations performed although when considering the running time the greedy version performs slightly better.

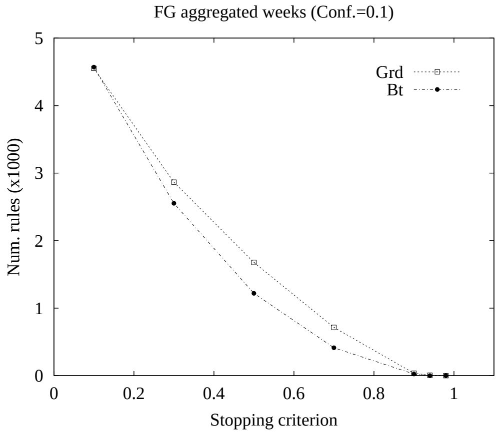


Figure 14: The variation of the number of rules with the fine-grained stopping criterion when applied to the real data.

Figure 14 shows the variation of the number of rules obtained with the stopping criterion for the real data. Each point in the plot corresponds to the average results of the seven data sets for a given configuration; each data set corresponds to a week of log data. The results confirm those obtained with synthetic data by showing that there is a proportionality between the number of rules and the value of the stopping criterion. In fact, a small variation in the value of the stopping criterion will lead to a similar variation in the number of rules obtained and that variation is close to uniform throughout the range of the stopping criterion.

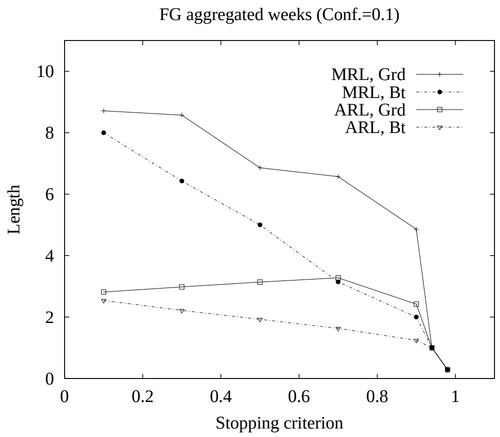


Figure 15: The variation of the size of the rules with the fine-grained stopping criterion when applied to the real data.

Figure 15 shows how both ARL and MRL vary with the stopping parameter for the real data sets. The results confirm those obtained with synthetic data. Moreover, it is interesting to note that with the greedy version the average rule length has its maximum value when the stopping criterion is set at 0.7, which is another indication that some long trails tend to be explored first. According to the definitions in Section 3.3 the rule-set inferred contains both the candidate and the maximal trails for a given value of the stopping parameter. Alternatively, the heuristic could be setup to give the analyst the maximal trails only.

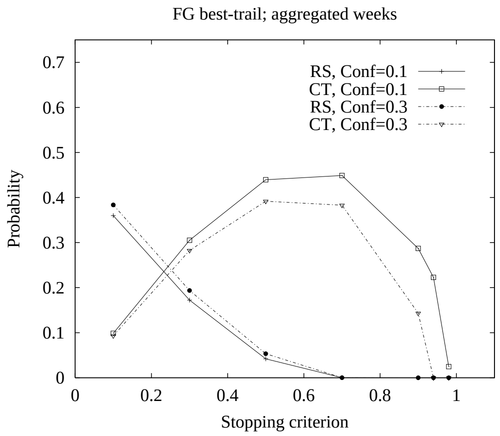


Figure 16: The evolution of both the candidate and the rule-set with the stopping criterion when the best-trail version is applied on the real data.

Figure 16 and Figure 17 provide a detailed analysis of the rule-set composition by giving the amount of probability in it that corresponds to maximal trails and the amount of

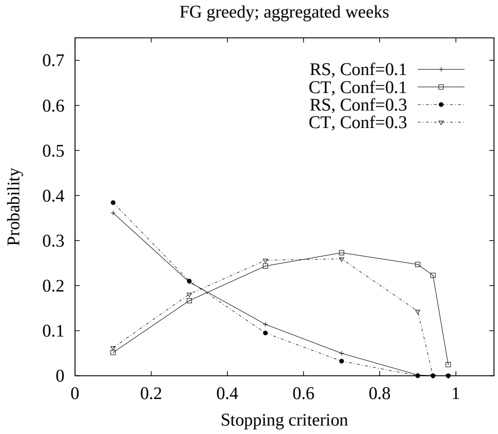


Figure 17: The evolution of both the candidate and the rule-set with the stopping criterion when the greedy version is applied on the real data.

probability that corresponds to candidate trails. In fact, these figures show the evolution of the sum of the probabilities of the maximal trails (RS) and the candidate trails (CT) with the value of the stopping criterion. While Figure 16 corresponds to the best-trail version Figure 17 corresponds to the greedy version. The results show that the best-trail version tends to keep a much larger set of candidate trails and consequently a smaller set of rules. This is explained by the ability of the greedy version to obtain some maximal trails at early stages in the exploration. This fact helps to explain why the running time of the greedy version is better since when having to locate the trail to expand next among all the candidate and admissible trails, it has to traverse a smaller set of candidates than the corresponding best-trail version.

4. CONCLUSIONS

In a previous paper we proposed to model a collection of user navigation patterns as a hypertext probabilistic grammar whose higher probability strings correspond to the user preferred trails [2]. Therein, an algorithm was proposed to find all the strings with probability above a specified cut-point.

In this work, we propose a new heuristic which aims at enhancing the HPG capabilities by enabling the analyst to have more control over the number, size, and probability of the rules induced. The heuristic implements an iterative deepening search that incrementally builds the rule-set. A stopping parameter is provided which measures the distance between the induced rule-set and the set of all strings with probability above the cut-point. Experiments with both synthetic and real data were conducted and the results show that the stopping parameter gives the analyst control over the number of rules mined and their probabilities.

As future work we plan to incorporate in the HPG model relevance measures of the web pages computed from a user query in order to assist the analyst in finding trails relevant to a given topic represented by a set of keywords.

5. REFERENCES

[1] J. Borges and M. Levene. Mining association rules in hypertext databases. In *Proc. of the fourth International Conference on Knowledge Discovery and Data Mining*, pages 149–153, New York, USA, August 1998.

[2] J. Borges and M. Levene. Data mining of user navigation patterns. In *Proc. of the Web Usage Analysis and User Profiling Workshop*, pages 31–36, San Diego, California, August 1999.

[3] J. Borges and M. Levene. A heuristic to capture longer user web navigation patterns. In *Proc. of the first International Conference on Electronic Commerce and Web Technologies*, Greenwich, U.K., September 2000. To appear.

[4] I. Cadez, S. Gaffney, and P. Smyth. A general probabilistic framework for clustering individuals. In *Proceedings of the sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Boston, Massachusetts, August 2000. To appear.

[5] I. Cadez, D. Heckerman, C. Meek, P. Smyth, and S. White. Visualization of navigation patterns on a web site using model based clustering. In *Proceedings of the sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Boston, Massachusetts, August 2000. To appear.

[6] L. D. Catledge and J. E. Pitkow. Characterizing browsing strategies in the world wide web. *Computer Networks and ISDN Systems*, 27(6):1065–1073, April 1995.

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

[8] C. Chatfield. Statistical inferences regarding markov chain models. *Applied Statistics*, 22:7–20, 1973.

[9] M.-S. Chen, J. S. Park, and P. S. Yu. Efficient data mining for traversal patterns. *IEEE Transactions on Knowledge and Data Engineering*, 10(2):209–221, March/April 1998.

[10] J. Conklin. Hypertext: An introduction and survey. *IEEE Computer*, 20(9):17–41, September 1987.

[11] 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.

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

[13] J. Nielsen. The art of navigating through hypertext. *Communications of the ACM*, 33(3):296–310, March 1990.

[14] M. Perkowitz and O. 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.

[15] M. Perkowitz and O. 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.

[16] R. R. Sarukkai. Link prediction and path analysis using Markov chains. In *Proceedings of the ninth International World Wide Web Conference*, Amsterdam, Holland, 2000.

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

[18] M. Spiliopoulou and L. 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.

[19] J. Srivastava, R. Cooley, M. Deshpande, and P.-N. Tan. Web usage mining: Discovery and applications of usage patterns from web data. *SIGKDD Explorations*, 1(2):12–23, 2000.

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

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

[22] T. W. Yan, M. Jacobsen, H. Garcia-Molina, and U. Dayal. From user access patterns to dynamic hypertext linking. In *Proc. of the fifth International World Wide Web Conference*, pages 1007–1014, Paris, France, May 1996.

[23] N. Zin and M. Levene. Constructing web-views from automated navigation sessions. In *Proc. of the ACM Digital Libraries Workshop on Organizing Web Space*, pages 54–58, Berkeley, California, August 1999.