A Heuristic to Capture Longer User 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 data mining model to capture user web navigation patterns, which models the navigation sessions as a hypertext probabilistic grammar. The grammar's higher probability strings correspond to the user preferred trails and an algorithm was given to find all strings with probability above a threshold. Herein, we propose a heuristic aimed at finding longer trails composed of links whose average probability is above the threshold. A dynamic threshold is provided whose value is at all times proportional to the length of the trail being evaluated. We report on experiments with both real and synthetic data which were conducted to assess the heuristic's utility.

1 Introduction

Web usage mining [4] is a recent research field which studies techniques that use log data to find patterns in user web navigation sessions. These *sessions* take the form of sequences of links followed by the user, which we call *trails*. 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 by providing adaptive web pages [9]. 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; however, these tools have limited analysis capabilities producing 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 log files. There have so far been two main approaches for mining user navigation patterns from log data. One approach is to map log data onto relational tables and an adapted version of standard data mining techniques, such as mining association rules, is invoked, see for example [3, 14]. In the other approach techniques are developed which are invoked directly on the log data, see for example [2, 11, 12].

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, 2, 8, 15].

In [2] we presented a new model for handling the problem of mining log data which models the web as a regular grammar; the grammar states correspond to web pages and the production rules to hyperlinks. The navigation sessions are then incorporated into the model in order to build a *hypertext probabilistic grammar* (or simply a HPG). Data mining techniques are used to find the higher probability strings which correspond to the user's preferred navigation trails. 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, a HPG can be used as an off-line tool to analyse server logs. The web site designer can benefit from having a better understanding of the users' browsing behaviour, characterised by the set of most popular trails. By understanding the users preferences the designer can provide adaptive web pages or 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 spends 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 which includes related products. Knowing the popular trails also allows to identify the pages where users frequently terminate their sessions so that the contents of such pages 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. In addition the user would be able to compare and/or exchange his HPG model with those of his peers and, for example, identify the preferred trails which are unknown to him but are among the preferences of his peers.

Herein we present a new heuristic for HPGs which aims at finding longer strings composed of links with average probability above a given threshold. In fact, although the algorithm proposed in [2] to find all the strings with probability above a cut-point is 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 (for the definition of confidence see Section 2). 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. These drawbacks led us to investigate a heuristic to compute a relatively small set of long rules. The method is called *inverse fisheye* and makes use of a dynamic threshold which imposes a very strict criterion for small rules and becomes more permissible as the trails get longer.

Section 2 presents an overview of HPGs, while Section 3 presents the proposed heuristic. Section 4 details the results of experiments we performed with both heuristics. Finally, in Section 5 we give our concluding remarks.

2 The Hypertext Probabilistic Grammar Model

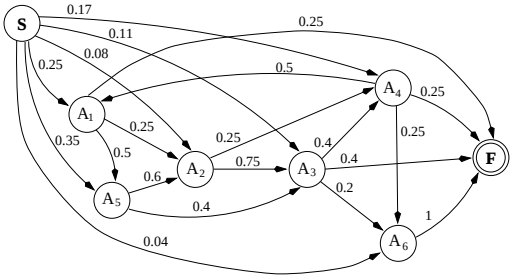
A log file consists in a per-user ordered set of web page requests from which are inferred the user navigation sessions. A *navigation session* is a sequence of page requests made by a user; techniques to infer the sessions from log data are given in [5]. We model a collection of user sessions as a *hypertext probabilistic language* [8] generated by a *hypertext probabilistic grammar* (or simply HPG) [2] which is a proper subclass of probabilistic regular grammars [13]. In a HPG, a non-terminal symbol corresponds to a web page and a production rule corresponds to a hypertext link. Two additional artificial states, S and F, represent the start and finish states of the navigation sessions. The probability of a 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 navigation sessions we obtain the number of times a page was requested, the number of times it was the start of a session, and the number of times it terminated a session. The number of times a sequence of two pages appears in the sessions gives the number of times the corresponding link was traversed. The probabilities of the start productions are weighted by a parameter α. If $\alpha = 0$ only states that were the first in a session have a start production with probability greater than zero, if $\alpha = 1$ the probability of a start production is proportional to the number of times the corresponding state was visited; α can take any value between 0 and 1. In the example of Figure 1 state A_1 was visited 4 times, 2 of which as the first state in a session and for $\alpha = 0.5$ we have that the probability of its start production is $p(A_1) = (0.5 \cdot 4)/24 + (0.5 \cdot 2)/6 = 0.25$. 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, $p(A_4A_1) = 2/4$, $p(A_4A_6) = 1/4$ and $p(A_4F) = 1/4$.

The grammar strings correspond to the user navigation trails. (We use the terms trail and string interchangeably.) A trail is in the language if its derivation probability is above a *cut-point*, λ. The set of trails with probability above the cut-point is the *rule-set*. The cut-point is composed of two thresholds $\lambda = \theta \cdot \delta$; $\theta \in (0, 1)$ is the *support* threshold and $\delta \in (0, 1)$ the *confidence* threshold. The support is the factor of the cut-point responsible for pruning out the strings whose first derivation step has low probability; the confidence is responsible for pruning out strings whose derivation contains transitive productions with small probabilities. The values of the support and confidence thresholds give the user control over the number and quality of trails to be included in the rule-set.

ID	User session
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)

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

The use of a Markovian model is sometimes criticised by arguing that it does not represent exactly the actual user navigation sessions. However, we view that as an advantage of the model since the probability of a long trail being often followed in exactly the same manner is low. Moreover, the contents of a page viewed recently should have more influence in choosing the next link to follow than a page viewed in the early steps of the session. On the other hand, the probability of choosing a link is not completely independent of the browsing history. Thus, we make use of the Ngram concept 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 next link choice, see [2]. In an Ngram each state corresponds to a sequence of N pages visited leading to a trade-off between the model accuracy and its complexity, since when the order of the model increases so does its size, measured in the number of states.

In [2] we reported the results of experiments with a modified Breadth-First Search (BFS) algorithm that induces all HPG's strings with probability above a cut-point. Although the BFS is very efficient it has the drawback of potentially returning a very large rule-set for small values of the cut-point and the rules are too short when the cut-point is close to one. Figure 2 (a) shows the variation of the number of rules with the confidence threshold and (b) the variation of the average (ARL) and maximum (MRL) rule length with the same threshold. It 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. These problems led us to the study of the inverse fisheye heuristic.

3 The Inverse Fisheye Heuristic

Herein, we propose a method to find a relatively small set of long rules composed of links with high probability on average. To that effect, we make use of

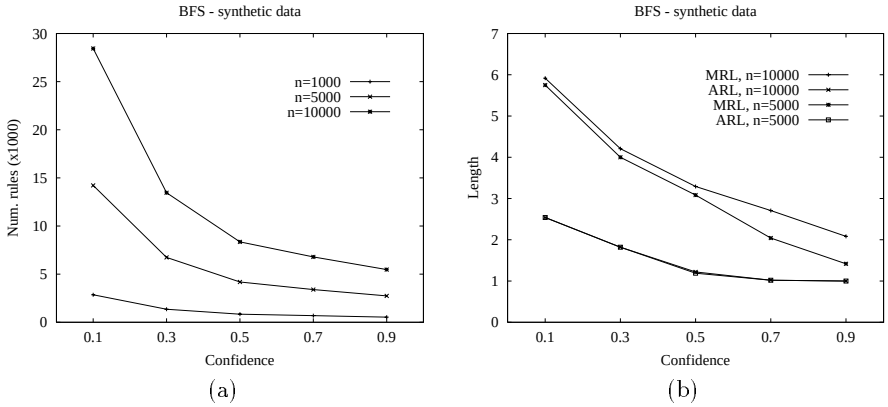


Fig. 2. Characteristics of the rule-set induced with the BFS algorithm.

a dynamic cut-point which imposes a very strict criterion for short trails and becomes more permissible as the trails get longer. This idea was motivated by the fisheye-view concept of Furnas [6], which is a method to visualise large information data structures on small computer displays. In a fisheye view the detail with which a document is shown is proportional to the global importance of the document and inversely proportional to its distance from the current document. We call our heuristic the *inverse fisheye* (IFE), since, as opposed to the Furnas concept, our method benefits those pages that are further away from the start of the trail being evaluated. With the IFE heuristic the cut-point becomes more permissible as the trails get longer. We propose two different ways of setting the dynamic cut-point. In the first the cut-point keeps its value proportional to the depth of exploration, and in the second the cut-point is devalued by a factor proportional to the expected decrease in the trail probability. We call the first method the *geometric cut-point* version. The initial value for cut-point is set by means of its two components, the support θ and the confidence δ thresholds, and an exploration tree is incrementally built from the start state while the value of the cut-point is updated as a function of the depth of the exploration. The geometric cut-point is defined to be

$$\lambda_G = \theta\delta^d \ ,$$

where d is the depth of the exploration tree measured by the number of links. The geometric cut-point is devalued geometrically in a way that keeps its value proportional to the trail's length. When the depth of the tree is 0, and the start productions are evaluated, the cut-point corresponds to the support threshold value. In the subsequent stages of the exploration the cut-point incorporates the confidence threshold a number of times corresponding to the number of transitive productions which derive the trails being evaluated.

We call the second method for setting the dynamic cut-point the *branching factor* version. An exploration tree is built and the cut-point is devalued as a

function of the exploration depth. In this case the devaluation takes into account the average branching factor, $BF = l/n$, where n denotes the number of grammar states and l the number of links. The branching factor corresponds to the expected number of out-links in a state and $1/BF$ to the average probability of an out-link. We define the branching factor threshold as

$$\lambda_B = \theta \text{ if } d = 0 \text{ and } \lambda_B = \theta \frac{\delta}{BF^{(d-1)}} \text{ if } d \geq 1 \ .$$

With both versions of the IFE heuristic an additional parameter has to be specified which sets an upper bound for the exploration depth, $\bar{d}$. This parameter is necessary because, for small values of the cut-point, there is no guarantee that the exploration will terminate. In fact, there can be a cycle of links generating a trail whose probability decreases at a slower rate than the dynamic cut-point, in such situations a trail would be explored indefinitely. We now present some interesting properties of the inverse fisheye heuristic.

Proposition 1. If $\delta = \frac{1}{BF}$ then $\lambda_G = \lambda_B$.

Proof. If $d = 0$ we have $\lambda_G = \theta\delta^0 = \theta = \lambda_B, \ \forall\delta$. If $d = 1$, $\lambda_G = \theta\delta^1 = \theta\delta\frac{1}{BF^0} = \lambda_B$. If $d > 1$ we have $\lambda_G = \lambda_B \ \equiv \ \theta\delta^d = \theta\delta\frac{1}{BF^{d-1}} \ \equiv \ \delta = \sqrt[d-1]{\frac{1}{BF^{d-1}}} = \frac{1}{BF}$. $\square$

Proposition 2. If $\delta > \frac{1}{BF}$ then $\lambda_B < \lambda_G$.

Proof. $\lambda_B < \lambda_G \ \equiv \ \theta\delta\frac{1}{BF^{d-1}} < \theta\delta^d \ \equiv \ \delta > \frac{1}{BF}$. $\square$

Proposition 3. For λ_G and a trail $t = ba_1 \ldots a_l$, where b and $a_i, 1 \leq i \leq l$ are the links composing the trail, we have: $\frac{p(a_1)+p(a_2)+\ldots+p(a_l)}{l} > \delta \left(\frac{\theta}{p(b)}\right)^{\frac{1}{l}}$.

Proof. $p(b)p(a_1)p(a_2)\ldots p(a_l) > \theta\delta^l \ \equiv \ \left(\frac{p(b)p(a_1)p(a_2)\ldots p(a_l)}{\theta}\right)^{\frac{1}{l}} > \delta \ \equiv$
$\equiv \ \left(\frac{p(b)}{\theta}\right)^{\frac{1}{l}} (p(a_1)p(a_2)\ldots p(a_l))^{\frac{1}{l}} > \delta \ \equiv \ (p(a_1)p(a_2)\ldots p(a_l))^{\frac{1}{l}} > \delta \left(\frac{\theta}{p(b)}\right)^{\frac{1}{l}}$
and by the theorem of the geometric means, [7] it follows that:

$\frac{p(a_1)+p(a_2)+\ldots+p(a_d)}{l} > \delta \left(\frac{\theta}{p(b)}\right)^{\frac{1}{l}}$. $\square$

The first property shows that the two methods for setting the dynamic cut-point are equivalent when $\delta = 1/BF$. The second property shows that when $\delta > 1/BF$ the branching factor version is more permissible than the geometric version since its value decreases faster. The third property implies that when $p(b)$ is close to θ the average link probability of a rule induced by the geometric version is greater than or equal to the confidence threshold δ, that is, the rule-set is composed of rules whose average link probability is greater than the confidence threshold. Note, that the property is not complete since not all trails with average link probability greater than δ are induced by the inverse fisheye property.

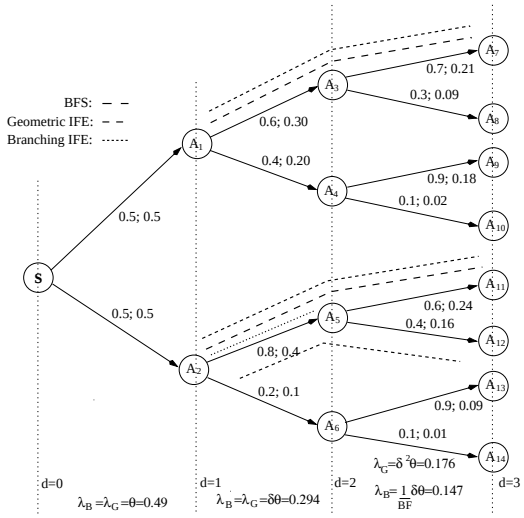


Fig. 3. A exploration tree for the inverse fisheye heuristic with $\theta = 0.49$ and $\delta = 0.6$.

Figure 3 shows an example of an exploration tree where $BF = 2$, $\theta = 0.49$, and $\delta = 0.6$. The cut-point values for the two versions are given in the figure, as well as the rules induced by the heuristic and by the BFS algorithm. A pair of numbers next to a link represents the probability of the link and the probability of the trail beginning in state S. When $d = 0$ the cut-point corresponds to the support value, when $d = 1$ it corresponds to the product of the confidence and the support $\delta\theta$; the inverse fisheye property only starts having effect for $d \geq 2$. With the geometric version only trails whose average link probability is above $\delta = 0.6$ are induced; trail $A_2A_5A_{12}$ is not a rule since its average link probability is just 0.6. Moreover, trail $A_1A_4A_9$ meets the acceptance criterion for $d = 2$ since its probability of 0.18 is above the cut-points, however, the trail is not a rule since it is rejected at an early stage when $p(A_1A_4) = 0.2 < 0.3$. Finally, the example also shows that in this case the BFS is ineffective since it induces a single short rule, i.e., A_2A_5. On the other hand, if the BFS algorithm is set up to run with the cut-point $\lambda = 0.176$ (the final value for the geometric version) the gain relative to the geometric version would only be the short rule A_1A_4.

4 Experimental Results

To assess the effectiveness of the IFE heuristic, experiments were conducted with both synthetic data and real log files. 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 thresh-

old between 0.1 and 0.9, and the support threshold was fixed to $1/n$ for each grammar size. For each configuration 30 runs were performed. The real log files were obtained from the authors of [10] and correspond to 2 months of log data which was divided into weeks of usage data. As stated in Section 3 the goal of the IFE heuristic is to induce a relatively small set of long trails. In the experiments we have set the maximum depth of exploration to vary between 3 and 6.

Figure 4 (a) shows the variation in the number of rules with the confidence threshold when the depth of exploration is set to 6. The IFE heuristic induces smaller rule-sets unless the confidence has a very low value. Note that for the IFE heuristic the confidence should be set with a value higher than $1/BF$, otherwise every trail has a high probability of being a rule. Figure 4 (b) shows the variation of both the average (ARL) and maximum (MRL) rule length with the confidence threshold. The results show that both versions of the heuristic induce a smaller rule-set with longer trails than the BFS algorithm.

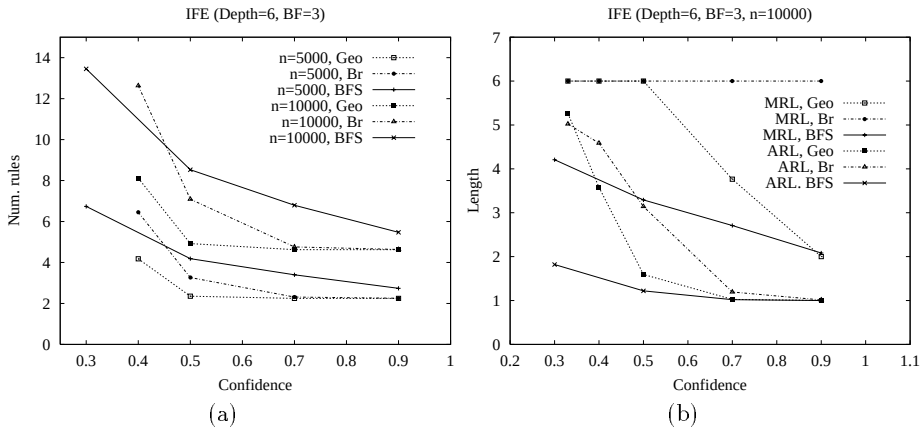


Fig. 4. Inverse fisheye rule-set characteristics with synthetic data.

Figure 5 (a) shows the variation in the number of iterations with the number of states where it can be seen that both versions present linear behaviour. Figure 5 (b) shows the variation of the number of rules obtained with the depth of exploration for the geometric version of the IFE heuristic. This results show that it is possible to have some control over the number of rules obtained by setting the both the dynamic cut-point and the exploration depth.

Figure 6 shows the distribution of the rules' length for the real data; each result corresponds to the average for the weekly data sets. The results show that with this real data set the branching version gives rule-sets that are too large, even for very high values of the initial cut-point. The geometric version of the heuristics achieves better results, especially for higher values of the support

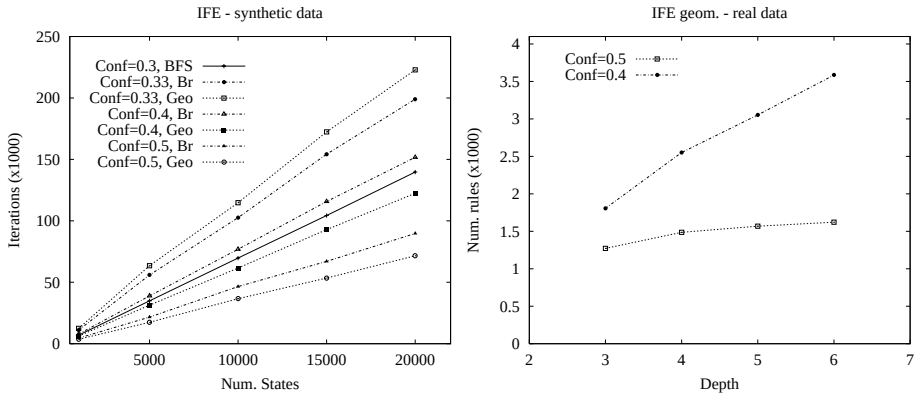


Fig. 5. Performance of the heuristics with synthetic data.

threshold where smaller rule-sets are obtained which contain longer rules. The analysis of the structure of the real data revealed that in spite of the real data sets having an average branching factor similar to the specified for the synthetic data, its standard deviation is very high (9 on average). This explains why in this case the BFS algorithm induces long rules for high values of the cut-point, since there is a considerable number of states with just one out-link.

Real data, aggregated weeks with $\theta = 1/n$									
				Rule length					
δ	Algorithm	Tot.	Avg.	1	2	3	4	5	6
0.5	BFS	1289	1.9	414	567	277	31		
0.5	Geometric	1623	3.4	131	343	460	323	175	191
0.7	BFS	933	1.8	335	423	162	13		
0.7	Geometric	871	2.3	188	349	251	62	18	3
0.9	BFS	714	1.78	276	325	107	6		
0.9	Geometric	681	1.87	234	312	124	11		
0.9	Branching	14749	5.8	234	77	175	371	836	13056

Fig. 6. Distribution of the rules' length with real data having $\theta = 1/n$.

5 conclusions

We propose a new heuristic which aims at providing the analyst with more control over the number, length, and probability of the rules induced by a hypertext probabilistic grammar. The heuristic makes use of a dynamic threshold which is very strict when evaluating short trails and becomes more permissible as the

trails get longer. Two different ways of varying the dynamic threshold are devised, one that takes into account the average branching factor of the underlying hypertext system (the branching version) and another that keeps the threshold proportional to the length of the trails being explored (the geometric version). Experiments with both synthetic and real data were conducted and the results suggest that the heuristic provides enhanced control over the size of the rule-set and the length of its rules. Moreover, the real data experiments shown that the branching version does not perform well in web sites where the branching factor has a high variance; in these situations the geometric version achieves better results. We plan to conduct experiments with other real data sets in order to get a better picture of the performance of the heuristics . As future work we plan to incorporate in the HPG model relevance measures of the web pages relatively to a user query in order to assist the analyst in finding trails that are also relevant to a given set of keywords.

References

1. J. Borges and M. Levene. Mining association rules in hypertext databases. In *Proc. of the 4th Int. Conf. on Knowledge Discovery and Data Mining*, pages 149–153, New York, 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, 1999.
3. M. Chen, J. Park, and P. Yu. Efficient data mining for traversal patterns. *IEEE Transactions on Knowledge and Data Engineering*, 10(2):209–221, 1998.
4. R. Cooley, B. Mobasher, and J. Srivastava. Web mining: Information and patterns discovery on the world wide web. In *Proc. of the 9th IEEE Int. Conf. on Tools with Artificial Intelligence*, pages 558–567, 1997.
5. R. Cooley, B. Mobasher, and J. Srivastava. Data preparation for mining world wide web browsing patterns. *Knowledge and Information Systems*, 1(1):5–32, 1999.
6. G. Furnas. Generalized fisheye views. In *Conf. proc. on Human factors in computing systems*, pages 16–23, 1986.
7. N. Kazarinoff. *Geometric Inequalities*. Random House, 1961.
8. M. Levene and G. Loizou. A probabilistic approach to navigation in hypertext. *Information Sciences*, 114:165–186, 1999.
9. M. Perkowitz and O. Etzioni. Adaptive web sites: an AI challenge. In *Proc. of 15th Int. Joint Conf. on Artificial Intelligence*, pages 16–21, Nagoya, 1997.
10. M. Perkowitz and O. Etzioni. Adaptive sites: Automatically synthesizing web pages. In *Proc. 15th Nat. Conf. on Artificial Intelligence*, pages 727–732, 1998.
11. S. Schechter, M. Krishnan, and M. D. Smith. Using path profiles to predict http requests. *Computer Networks and ISDN Systems*, 30:457–467, 1998.
12. M. Spiliopoulou and L. Faulstich. WUM: a tool for web utilization analysis. In *Proc. Int. Workshop on the Web and Databases*, pages 184–203, Valencia, 1998.
13. C. Wetherell. Probabilistic languages: A review and some open questions. *Computing Surveys*, 12(4):361–379, 1980.
14. T. Yan, M. Jacobsen, H. Garcia-Molina, and U. Dayal. From user access patterns to dynamic hypertext linking. In *Proc. of the fifth Int. World Wide Web Conference*, pages 1007–1014, Paris, 1996.
15. 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, 1999.