Constructing Web Views from Automated Navigation Sessions

Nadav Zin and Mark Levene
Department of Computer Science
University College London
Gower Street, London WC1E 6BT, U.K.
email: N.Zin@cs.ucl.ac.uk

ABSTRACT

Existing web search engines provide users with the ability to query an off-line database of indices in order to decide on an entry point for further manual navigation. Results are often presented as a list of URLs in descending order of relevance, with no information on the underlying topology of the result set. We believe that information on the topology is important for useful exploration and can also help to reduce the feeling of disorientation that users experience. We present an alternative to the result set of a conventional search engine, which we call a *web probabilistic view* – a weighted subgraph of the underlying document space which maximizes the overall expected relevance of trails. We model a web database as a probabilistic grammar and present several algorithms to calculate its weights, expressed as probabilities attached to transition rules. We provide results from a recent set of experiments, showing the effectiveness of our approach measured as improvement in the expected relevance of the grammar.

KEYWORDS: Hypertext, Trail, Web, View, Navigation, Probabilistic Grammar

1 Introduction

A hypertext database is a collection of interlinked pages of text or multimedia, which allows navigation in a document space by following links from one page to another [Nie95]. This non-sequential reading of the material (called *browsing*), can become quite frustrating when directed by a particular goal or interest. Some pages are, naturally, more relevant than others and the user has to decide which outlink to follow next. The task is essentially a multi-objective search in which users try to maximise their utility of visiting relevant pages while minimising the effort required to locate them, and at the same time enjoying the associative relationships between linked nodes.

Long manual navigation sessions, especially in large document spaces such as the WWW, can also cause a feeling of disorientation. To tackle the problem search engines help users select a ”good” starting page, thus restricting the search space for further exploration. In response to a particular query, search engines suggest a list of entry points, usually displayed in descending order of relevance, from which users can start a manual navigation session. However, returning a list of URLs in response to a query expression has a serious drawback – it looses an inherent property of the hypertext database; i.e. the topology in which those pages are embedded. In fact, selecting any of the links in the result set still leaves the user to speculate about the relevance of subsequent links.

Although the idea of filtering out relevant pages resembles a view in a relational database (where only those tuples satisfying a query expression are shown), a search engine’s result set can hardly be considered as a hypertext view. Based on a network of documents as source, we define a hypertext view as a sub-network comprising a set of trails. It should be possible, at least in theory, to enumerate all possible trails and select those that best suit the user’s criteria. However, in reality the size of the document space and potential cycles in the topology deem such an approach impractical. Some heuristics are therefore required for highlighting relevant trails that constitute a good hypertext view. It should be noted that the concept of a trail is not a new one and was first introduced by Bush in his visionary 1945 paper describing a hypothetical hypertext machine called *memex* [Bus45]. Bush envisages *trail blazing* – the task of constructing association sequences between pieces of information. Sequences could then be ”loaded” into memex, and so trails serve as a medium for information exchange between people sharing their knowledge and experience.

In this paper we introduce a generic algorithm for calculating web views and four reinforcement policy variants. These differ in the credit horizon taken at the trail level and the learning rate used between iterations. We distinguish between a fixed credit horizon - which credits all of the transition rules of a trail homogeneously, and a dynamic one - which looks only at the trail’s remainder. The learning rate taken between iterations can also be computed with two different policies: a fixed one with a constant factor, and dynamic one which self-adapts throughout execution.

2 The Web Probabilistic Grammar

Consider the hypertext database shown in Figure 1. Starting at the initial page **a**, this database can be traversed in many ways, each resulting in a different trail. Two such trails are highlighted in the figure. We assume that, given the context of a query, the utility "gained" by visiting a page (shown in brackets next to the label) depends on the content rather than the position within the trail. While viewing a page, the user is faced with the following decision problem: navigate to one of the outlinks or terminate the navigational sequence altogether. Choosing the first option increases the accumulated utility of the navigational sequence by the relevance of the respective destination page. It also costs another navigational step. The trade-off between increasing the accumulated relevance and decreasing the overall number of steps taken is tantamount to a global optimization of the trail's average relevance.

Our model is based on a finite automata representation of the hypertext database in which nodes are modelled as states and links as transition rules [LL99]. Changing states in the automaton closely resembles the user activity of navigation in the document space. A special reference is made to a starting node called the *database source* from which navigation starts.

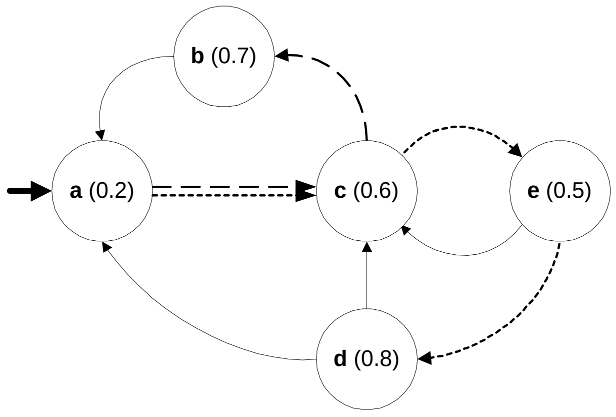

Figure 1: Two trails in a hypertext database

The hypertext database of Figure 1 can be defined as the regular grammar $G = \langle V_N, V_T, R, S\rangle$ where the set of terminal symbols $V_T = \{a, b, \ldots, e\}$, the set of non-terminal symbols $V_N = \{A, B, \ldots E\}$, the set of transition rules $R = \{A \rightarrow aC,\ A \rightarrow a,\ B \rightarrow bA,\ B \rightarrow b,\ C \rightarrow cB,\ C \rightarrow cE,\ C \rightarrow c,\ D \rightarrow dA,\ D \rightarrow dC, D \rightarrow d,\ E \rightarrow eC,\ E \rightarrow eD,\ E \rightarrow e\}$ and the start state $S = A$. Note that R contains two types of production rules: $X \rightarrow xY$ in which non-terminal symbols appear both in the premise and conclusion parts – representing actual links in the hypertext graph, and $X \rightarrow x$ in which a one non-terminal appears only in the premise – used to support termination of strings. We call the latter *stopping rules*.

The grammar model not only enables specification of the hypertext topology but also provides means of investigating trails [LL99] (c.f. [Par98]). Navigation sessions can be viewed as derivations of words in a language defined by the grammar. The n-step derivation $S \overset{n}{\Rightarrow} w$ where w is a sequence of terminal symbols called a *word*. A hypertext grammar is deterministic and unambiguous, and thus implies a bijection between words (the derived strings) and trails (the sequences of production rules used in their generation). From now on we will use those terms interchangeably. Complex document spaces allow many navigational sequences between pages, a collection of which is central to our model of a *web view*. The set $\{w|S \overset{*}{\Rightarrow} w\}$ of all words generated by a grammar G is called a *language*, denoted as $L(G)$. In the context of this work we call $L(G)$ the *total view* of the hypertext. A *web view* is then any subset of the total view.

Web navigation is not a random activity but is often motivated by the user's interest in a specific topic. Search engines allow users to specify topics as *ad-hoc* queries expressed as a set of keywords with boolean operators. Those serve as functions from nodes to a relevance range, typically between 0 and 1. The actual calculation of relevance scores is a classical information retrieval problem (see [RSJ76] [GF98]) and is outside the scope of this model. In this work we stay indifferent to the scoring method as long as it is applied consistently on all nodes. Extending relevance scoring we define a trail's utility $u(t)$ as the simple average of its page relevancies and a view's utility $u(v) = E[u(t)]$ as its expected trail relevance.

So far, our model implicitly assumed that, given a certain page, outlinks are equally likely to be followed. However, real web navigation is a more complex stochastic process, as the outlinks probability distribution at the node level is not necessarily uniform. Users tend to select outlinks that, they believe, would lead to relevant pages and consequently increase their overall utility in the session. We incorporate this belief into our model of a *web probabilistic grammar* $G = \langle V_N, V_T, R, P, S\rangle$ defined as a regular grammar in which every transition rule of R is assigned a corresponding weight $P_r \in (0, 1]$ such that $\forall n \in V_N : \sum_n P_r = 1$ where n is the non-terminal premise of r. Note that P defines probability distributions over non-terminals to support selection of rewrite rules (for a comprehensive survey of probabilistic grammars see [Wet80] and [Cha93]). A non-probabilistic grammar (such as the simple web grammar) can, therefore, be seen as a special case in which those distributions are uniform [LL99].

Rule weights are also useful to describe the probability of words. We assume that, given a page, the selection of an outlink follows a Markovian process which is not affected by previous nodes in the trail. The probability $P(S \overset{*}{\Rightarrow} w)$ of a word can, therefore, be taken as a product of the individual rules used in its derivation. More formally, let $t = [r_1, r_2, \ldots, r_n]$ be the trail corresponding to w. Then, $P(S \overset{n}{\Rightarrow} w) = \prod_{i=1}^{n} P(t[i])$. Observe that the range of P is $(0, 1]$ and therefore $0 < P(w) \leq 1$. Due to the duality of trails and words, we thus have $P(t) = P(w)$.

Probabilistic web views that share the same topology can be seen as *dialects* of the same grammar. Their syntax produces the same words but with different frequencies, depending on the probabilities attached to transition rules. Given a certain topology we look for the weight vector P^* for which the grammar $G = \langle V_N, V_T, R, P^*, S\rangle$ maximizes the expected trail relevance $E[u(t)]$.

3 The Reinforcement Algorithm

This section describes a reinforcement algorithm for calculating rule weights in web probabilistic grammars. The approach is based on a *sample-credit-update* loop, a common concept in reinforcement learning [SB98], in which exploration and credit are tied together. The general principle is as follows: 1) a sample of trails is taken from the language; 2) transition rules are credited according to the relevance of trails in which they appeared and a new set of probabilities is calculated; and 3) the automaton is updated according to the learning rate $0 \leq \alpha < 1$ and the new values are used to generate a new sample. The loop terminates when the $u(v)$ settles on a fixpoint (i.e. there is no significant change for a fixed number of iterations).

We distinguish between two credit and update policies resulting in four variants of our generic algorithm. The *Fixed Credit Horizon* (FHC) is one in which all the trail's links are credited equally with the same value. When the credit of a rule depends on its position in the trail the policy is *Dynamic Credit Horizon (DCH)*. Weights are updated between iterations as a linear combination of new and current values with α and $(1 - \alpha)$ respectively. When α is fixed throughout the run we say that the algorithm has a *Fixed Learning Rate* (FLR). However, when α changes between cycles we say that the algorithm has a *Dynamic Learning Rate (DLR)*.

The algorithm assumes the following primitive routines: i) $initialise : R \rightarrow (0, 1]$ which takes a set of rules and returns their corresponding weights set P such that $\forall n \in V_N : \sum_n P_r = 1$ where n is the non-terminal premise of r, ii) $normalise : C \rightarrow (0, 1]$ which takes a set of credits corresponding to the rules set R and returns a new weights set $P \propto C$ normalized at the premise level, and iii) $sample(G, n)$ which takes a probabilistic grammar G of the form $\langle V_N, V_T, R, P, S\rangle$ and returns a set of n words in $L(G)$ induced by P.

The generic algorithm, presented in Figure 2, should be read as follows. Initialise the probabilities vector P such that rule weights corresponding to outlinks of each node add up to one. Until convergence of the trail expected relevance perform the following loop. Let G be a probabilistic grammar with distribution P. Use G to generate a sample T of n trails. Calculate the vector C of rule credits according to the policy (FCH or DCH). Normalize C at the node level to become an alternative probability vector P'. Update the probabilities vector as a linear combination of the old P and the new alternative P' with the appropriate α. For the DLR, α is determined as a function of T (a reasonable criteria is to compute α according to the change in expected trail relevance). Then, use the new P to generate the next sample. Convergence is detected when the expected trail relevance has not significantly changed for a fixed number of iterations.

```
Algorithm 3.1 (web_view ($\langle V_N, V_T, R, S\rangle, n, Q, \alpha$))
1.  begin
2.    $P \leftarrow$ initialise $(R)$ ;
3.    repeat
4.      $G \leftarrow \langle V_N, V_T, R, P, S\rangle$;
5.      $T \leftarrow$ sample $(G, n)$;
6.      $C \leftarrow$ credit $(T, Q)$;
7.      $P' \leftarrow$ normalize $(C)$;
8.      $P \leftarrow$ update $(P, P', \alpha)$;
9.    until the expected trail relevance converges to a fixpoint;
10.   return $\langle V_N, V_T, R, P, S\rangle$;
11. end.
```

Figure 2: The generic web view algorithm

4 Empirical Validation

In order to evaluate our approach an extensive set of off-line experiments were conducted using synthetic data with similar characteristics to the web. This enabled us to maintain a controlled experimental environment, test extreme cases and reduce the overall time by cutting out the network traffic. We used Java 1.2 for implementation and Oracle8 as an off-line repository.

Our first step was to create a library of 1000 nodes from which many instances of *web cases* were created. We used $u(n) = e^{-10x}$ with x uniformly distributed in [0,1] to simulate the scoring function of a node n by the search engine, giving us an expected relevance of just above 0.1. Although node relevancies were only assigned once in the library, they could still be used throughout the whole experiment with the assumption that the distribution is fixed over queries.

Using the library we then generated 15 web cases in 3 different sizes of 100, 500 and 1000 nodes. Real data collected from the web enabled us to study the empirical distribution of node connectivity. We used this distribution to randomly assemble the nodes in each web case instance. Each of the four algorithms was ran against all web case instances with three different sample sizes of 250, 500 and 1000 trails. For the FLR policy α was set to 0.5, whereas for the FLR fluctuations around 0.5 were allowed. Overall 180 jobs were executed, repeating each configuration 10 times. We now provide a brief summary of the results (for full details see [ZL99]).

As can be seen in Table 1, all algorithms increase the expected trail relevance from an initial average of 0.075 to an average of 0.170. Note that only 20% of all nodes in the library had a greater relevance, which means that the we have managed to focus and construct trails from those nodes that had higher scores. It should also be noted that convergence was reached in less than 100 iterations (42.1 on average). We note that, as expected, smaller sample sizes took longer to converge, and in addition that the average trail length is shorter in larger networks. Moreover, the results suggest that performance is sublinear with the size of the network, rendering our approach scalable.

Figures 3-6 present typical behaviour of the four algorithms on 1000 nodes web cases with a sample size of 250 trails. Note the increase in expected trail relevance (Figure 3) along with the entropy of its distribution (Figure 4). This suggests that all four variants have managed to converge on web views consisting of ”good” trails with similar derivation probabilities.

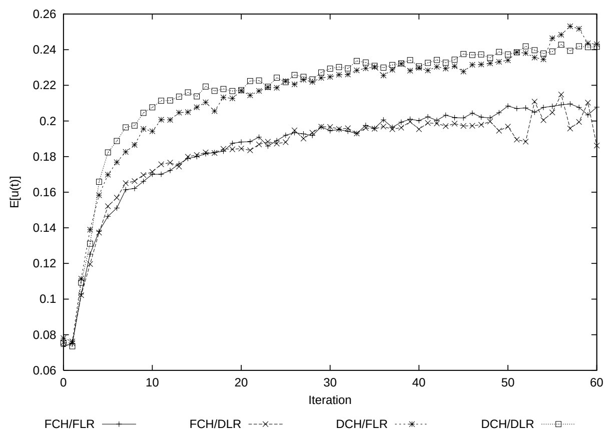

Figure 3: Expected trail relevance

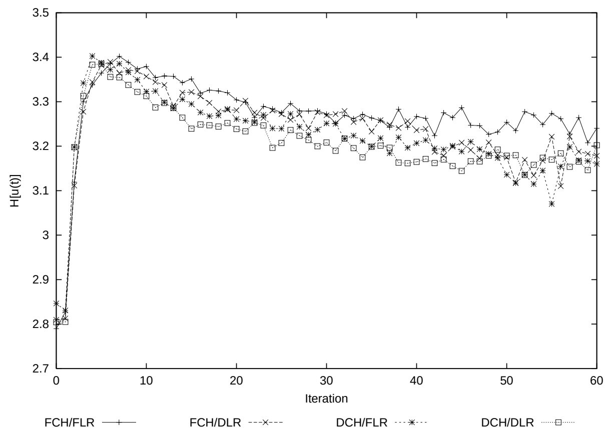

Figure 4: Entropy of trail relevance

Furthermore, the fixed credit horizon manages to keep a higher diversity (Figure 5) – over 75% distinct trails in the sample. Finally, the trails produced by the dynamic crediting algorithms, were not only more relevant (Figure 3) but also shorter (Figure 6).

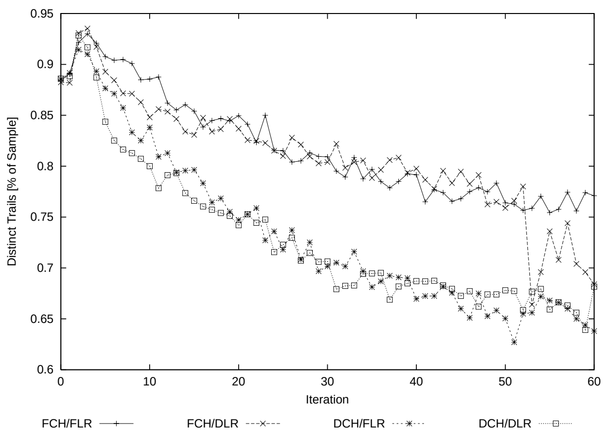

Figure 5: Sample diversity

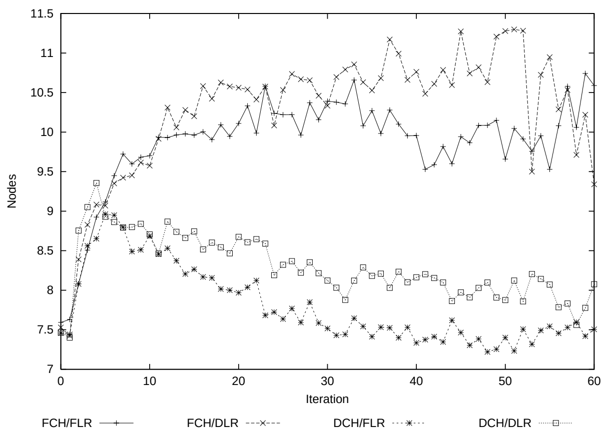

Figure 6: Average trail length

5 Concluding Remarks

Web views as a navigation support tool in large document spaces was suggested much before the phenomenal explosion of the World-Wide Web. [UY89] have suggested to combine a record of the user’s path in the Intermedia system with a map of available links, which they called a *web view*. In [MF95], the authors describe the *Navigational View Builder* – an interactive tool which allows users to create their own visualization of the WWW. Views as a conceptualizing mechanism were also discussed in [SNP97] in the context of virtual hierarchies and virtual networks. Moreover, techniques for web searching have now progressed beyond traditional information retrieval (see, for example, machine learning technique in [CS96] and augmenting engines with information about link structure in [Kle98]).

Automata theory provides good tools for modeling a hypertext database [LL99]. Its extension to probabilistic grammars enables analysis of user navigation in the document

		Expected Trail Relevance				Iterations for Convergence				Average Trail Length			
		FCH		DCH		FCH		DCH		FCH		DCH	
Nodes	Sample	FLR	DLR	FLR	DLR	FLR	DLR	FLR	DLR	FLR	DLR	FLR	DLR
100	250	.143	.143	.184	.185	35.0	34.1	34.5	27.5	14.0	14.2	16.8	16.7
	500	.133	.132	.172	.174	30.8	30.1	27.3	25.3	12.6	12.6	17.8	17.2
	1000	.124	.125	.167	.168	24.9	26.8	24.8	23.9	11.3	11.4	17.4	17.3
500	250	.165	.164	.209	.204	46.5	42.8	55.1	49.5	12.3	12.3	10.2	10.9
	500	.154	.150	.193	.186	46.7	39.8	58.3	44.8	12.7	12.4	12.1	12.0
	1000	.139	.139	.171	.168	36.2	36.6	44.2	35.0	11.9	11.8	13.5	13.5
1000	250	.182	.179	.215	.211	52.7	50.4	55.9	53.1	12.1	12.0	8.5	9.1
	500	.170	.167	.213	.207	51.6	48.0	63.6	53.5	13.0	12.8	9.5	10.0
	1000	.156	.154	.198	.194	46.9	44.0	60.4	56.1	13.3	13.0	11.2	11.3

Table 1: Summary of experimental results

space. The approach presented in this paper uses repeated sampling for learning of the optimal transition probabilities. We have shown that the generic algorithm for reinforcement of probabilities significantly increases the expected trail relevance. The final weights define an implicit structure, containing trails with high relevance with respect to a given query. The collection of high relevance trails is in fact a view of the hypertext structure which highlights recommended navigation sequences.

We believe that a web view and the algorithms used for its construction will be useful components in the next generation of web search tools. We have recently finished implementing a software robot to collect real data from the web into a local repository. We intend to use it to conduct another set of experiments to further validate our approach.

REFERENCES

Bus45. V. Bush. As we may think. *The Atlantic Monthly*, July 1945.

Cha93. E. Charniak. *Statistical Language Learning*. The MIT Press, Cambridge, Massachusetts, 1993.

CS96. W. W. Cohen and Y. Singer. Learning to query the web. In *AAAI Workshop on Internet-Based Information System*, 1996.

GF98. D. A. Grossman and O. Frieder. *Information Retrieval: Algorithms and Heuristics*. Kluwer Academic Publishers, Dordrecht, 1998.

Kle98. J. M. Kleinberg. Authoratitive sources in a hyperlinked environment. In *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms*, pages 668–677, San Francisco, 1998.

LL99. M. Levene and G. Loizou. A probabilistic approach to navigation in hypertext. *Information Sciences*, 114:165–185, 1999.

MF95. S. Mukherjea and J. D. Foley. Visualizing the world-wide web with the navigational view builder. *Computer Networks and ISDN System, Special Issue on the Third International Conference on the World-Wide Web '95, Darmstadt, Germany*, April 1995.

Nie95. J. Nielsen. *Multimedia and Hypertext*. Academic Press, 1995.

Par98. S. Park. Structural properties of hypertext. In *Proceedings of the 9th ACM Conference on Hypertext and hypermedia*, pages 180–187, 1998.

RSJ76. S. E. Robertson and K. Spark Jones. Relevance weighting of search terms. *Journal of the American Society for Information Science*, 27(3):129–146, May-June 1976.

SB98. R. S. Sutton and A. G. Barto. *Reinforcement Learning*. MIT Press, Cambridge, Massachusetts, 1998.

SNP97. P. A. Smith, I. A. Newman, and L. M. Parks. Virtual heirarchies and virtual networks: Some lessons from hypermedia usability research applied to the world wide web. *International Journal of Human-Computer Studies*, (47):67–95, 1997.

UY89. K. Utting and N. Yankelovich. Context and orientation in hypermedia networks. *ACM Transactions on Information Systems*, 7(1):58–84, January 1989.

Wet80. C. S. Wetherell. Probabilistic languages: A review and some open questions. *Computing Surveys*, 12:361–379, 1980.

ZL99. N. Zin and M. Levene. Constructing web views from automated navigation sessions. *Research Note RN/99/35, Department of Computer Science, University College London*, June 1999.