Exploiting Hyperlinks for Automatic Information Discovery on the WWW

Chia-Hui Chang, Ching-Chi Hsu and Cheng-Lin Hou
Department of Computer Science and Information Engineering
National Taiwan University, Taipei, Taiwan 106
{chia, CCHsu, clhou} @ails1.csie.ntu.edu.tw

Abstract

The explosion of the World Wide Web as a global information network brings with it a number of related challenges for information retrieval and automation. The link structure, which is the main feature of the hypermedia environment, can be a rich source of information for exploration. This paper is centered around the exploiting of hyperlinks in the subject of automatic discovery. In this paper, we will discuss the design of site traversing algorithms and identify some important parameters for hyperlink-based information discovery on the Web. Meanwhile, we propose a new evaluation method, the search depth, *for comparing various hyperlink traversing algorithms.*

1 Introduction

The World Wide Web is a world of great richness and diversity. It contains a complete universe of online information. However, with its continuing growth, finding information on the Web has become a difficult and time-consuming task because of the Web's tremendous size. Hence, providing better information services becomes a challenge for many researchers.

Of the services provided by most information systems, there are two fundamental services that have been implemented on the internet: document search and document routing. Document search is the process of selecting and returning to the user desired documents from a large corpus based on users' queries. While document routing is the dissemination of incoming documents to appropriate users on the basis of user interest profiles. In the past few years, most of us have experienced the convenience of document search provided by index servers on the Web, for example, AltaVista, etc. However, we have also experienced the bombing of the abundant information provided by the index servers as the Web grows. As for document routing services, they are provided mainly in the USENET environment, where documents are filtered and disseminated to subscribed users [9, 11].

Another information service on the Web, which is the main subject of this paper, is automatic discovery that finds new information on behalf of users. Two approaches are adopted for this purpose. One is query based discovery, which finds new information by submitting queries to index servers to get results for further filtering [7]. The other is hyperlink-based discovery, which traverses the links from given starting pages provided by users, and hopes to reach some valuable pages that the user could not easily get to by manually navigation [6].

The subject of this paper is centered at the design and evaluation of hyperlink-based discovery tools on the Web. The idea comes from the observation that links are manually constructed pointers which reflect an author's attempts to relate his/her writings to others. Thus, given some related documents as starting points, search mechanisms can be designed for information discovery to decrease the amount of browsing efforts. In [6], Menczer first explores this approach and applies agent technology to choose promising hyperlinks from some starting points. The system, called ARACHNID, is tested in the environment of Encyclopedia Britannica, where relevant sets of articles associated with a large number of queries are readily available. The result is good comparing to breadth-first search. However, if such performance sustains on the Web is unknown. Since the links and contents in the corpus of the Encyclopedia Britannica are far more robust and richer than those links and contents on the Web, it is hard to say if such a algorithm applies on the Web.

Another question regarding hyperlink-based discovery is the way of performance evaluation. In ARACHNID, *precision* (the proportion of retrieved documents that are relevant) and *recall* (the proportion of relevant documents that are retrieved) are measured since the algorithm is applied in a closed environment. However, the measure of recall is not adequate on the Web, since the Web is an open environment. Meanwhile, there is an important issue that has been ignored. Recall that the purpose of automatic discovery is to find new information that the user could not easily reach by himself. Thus, the search depth from the starting point is also an important factor in performance evaluation. If a discovery agent retrieved documents that are located just around the starting pages, it is possibly that users could reach them directly by clicking the anchors in the starting pages.

These two questions are the motivation of our research on hyperlink-based discovery. In the following sections, we will focus on link exploration to design search algorithms for automatic discovery. We will also propose performance evaluation by way of precision and search depth. Based on these two evaluation methods, we compare the advantages of various mechanisms employed in a discovery.

1.1 Related Work

The link structure of a hypermedia environment can be a rich source of information about the contents of the Web. Exploration of hyperlinks is a hot topic in recent research, especially in the subject of document search. Given a query, several attempts [3, 4] are tried to rank better the search results. In [3], Chakrabarti et al. proposed a new ranking method by finding *authority pages* and *hub pages* from an expanded set of 200 hypertexts. They define a good *hub* as a page that points to many good authorities, and a good *authority* as a page that is pointed to by many good hubs. Based on these fundamental ideas, they constructed an automatic resource compiler that provides a similar effect just like other manually-complied taxonomy such as Yahoo.

Another application is the browsing assistants, which are essentially interface agents that look over a user's shoulder and supply suggestions in response to the user's query. For example, in Letizia [5] and WebWatcher [1], the agents learn a user's interest along the browsing process, evaluate each hyperlink by prefetching their contents, and suggest the most promising hyperlink to the user (according to some criteria specified by the user or the browsing history). The idea is intuitive from the observation that the process of browsing could in some sense be modeled by best-first search. Thus, the success of the browsing assistants relies on the estimation of hyperlink relevance.

The third kind of hyperlink applications is the Hyperlink Vector Voting$^{(sm)}$ introduced by Li [4]. The crucial difference of Hyperlink Vector Voting from other index servers is to collect anchor descriptions as the content of a hypertext. The point is what a site says about itself is not considered reliable. What really matters is what others say about that page. Through this idea, they hopes to provide a decentralized and democratic way to identify each Web page.

In [10], Spertus generalizes the varieties of link information on the Web and describes how the links can be exploited to find individuals' homepages and new locations of moved pages. The final application of the link structure is for automatic discovery. As we mentioned above, ARACHNID has a promising performance over the set of Encyclopedia Britannica, but the result over the Web is unknown. In addition, search goal can be another factor in the subject of automatic discovery. We will provide our answers to these two questions in the following context.

1.2 Exploiting Hyperlinks for Automatic Discovery

Our research goal is to explore hyperlinks for automatic discovery. By *automatic discovery*, we mean to find information that the user could not easily reach to by manually browsing or traversing through hyperlinks. The discovery process is initiated by given some hypertexts as starting points. There are various resources that the starting pages come from. For example, the bookmark file in a user's browser, or search results returned by consulting a search engine. For the query examples and starting pages we used here, they are obtained through the multi-engine search tool proposed in [2], where each query is specified by a description profile as the search goal. The top 10 relevant hypertexts then constitute the starting points for the discovery activity.

One commonly agreed fact of hyperlinks is that links are manually constructed pointers which reflect an author's attempts to relate his/her writings to others. This consensus is the orientation of the idea on applying link traversing algorithm for discovery. Nonetheless, a generally overlooked fact is that relevance between two linked documents decreases as the distance grows. That is, hypertexts in close vicinity are more similar to each other than hypertexts in far distance. The reason is that, as one traverses away from starting points, there are more spurious reasons why two hypertexts are linked together. Hence, the most relevant hypertexts are indeed residing just around the starting pages.

Under this circumstance, a breadth-first search algorithm is enough to traverse all hypertexts in the neighborhood of the starting pages, and the central problem becomes the selection of the most relevant pages from the traversed ones according to some measurement of text resemblance to the search goal. How-

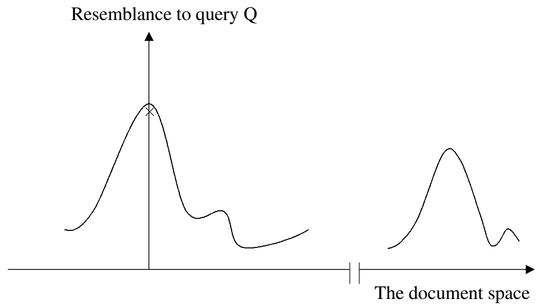

Figure 1: **The resemblance to a query Q decreases as distance from starting point (marked as x) grows.**

ever, this approach is contradictory to our discovery goal, since a discovery process that returns nearby Web pages from the starting pages is less significant than one that returns pages father from the starting pages. Our point is that even though hypertexts near the starting pages are usually more similar to our query, it is possible that another set of relevant hypertexts exists after crossing several traversing steps (see Figure 1).

On the other hand, even though recall is a more adequate evaluation than precision for hyperlink-based discovery, it is infeasible to measure unless we can traverse through all of the links. To remedy the impossibility of recall, new performance evaluation is needed. Hence, we propose the *search depth* (defined as the average depth of hypertexts) of the retrieved Web pages that are relevant as our evaluation method for hyperlink-based discovery instead of recall. Here, the depth of a document is the number of links chained from some starting point to that document. Based on these understanding of hyperlinks and automatic discovery, we then go on to design site traversing algorithms for this purpose.

2 The Basic Site Traversing Algorithm

Figure 2 outlines the pseudo code for the basic Site Traversing Algorithm (STA). The search algorithm is basically a multi-agent search, where each agent has a set of candidate hyperlinks and the search goal. The queries and starting points used in the following experiments are obtained as discussed in Section 1.2. At initialization, each agent a is assigned a starting page to be traversed and a search goal $v(q)$, which is the expanded description of a query q. It can also be obtained from the starting hypertexts such that the discovery can be interpreted as "finding more information similar to the starting hypertexts".

When exploring a hyperlink, an agent examines each outgoing link and compares the link's exploring potential to a preset threshold ϵ. Note that when the threshold is set to zero, the basic STA can be viewed as a breadth-first discovery method. In real implementaion, an extra variable is used to control the spawn of new agents to avoid the exhaustion of all resources at a time. This approach also ensures that hyperlinks that are nearer to starting points are explored earlier than those that are farther from starting points.

There are several features in the basic STA. First, each agent makes its own decision about what hyperlinks to be explored without additional communication to or from other agents. This feature is implemented by a preset threshold ϵ such that whenever the measure of a hyperlink h has a exploring potential, λ_{h,E_a}, greater than ϵ, this hyperlink gets a chance to be traversed by a new agent a'. The new agent a' inherits the search goal E_a and is assigned a candidate set $C_{a'}$ containing hyperlinks anchored from h. The advantage of such a design is the minimum communications in the multi-agent search.

The second feature is the control of the termination condition. Generally, if a high threshold is set, the search process will terminate at some point since no more hyperlink has exploring potential greater than ϵ. But if a low threshold is set, the search process may continue such that it never ends. Thus, a second parameter is used to control the number of hyperlinks to be visited such that the process terminates when an upper bound (200) is reached.

The visited hyperlinks of the basic STA is then ranked by their similarities to the search goal. After ranking, the top 30 hypertexts with highest similarities are reported to the user for relevance evaluation. Note that the stage of final ranking is necessary, because returning hypertexts in the visited order will result in all hypertexts that are nearest to starting points to be returned, which is definitely not what we expect.

As a whole, the hyperlink traversing follows a best-first search principle to some extent. With different methods used for evaluating a hyperlink's exploring potential, the basic STA could perform differently. In addition, evaluating the relevance of hyperlinks to the query is also a main factor affecting precision in the final stage. A commonly used evaluation method in many information systems is the text-based evaluation, where only the text information of a hypertext is used in measuring similarity to a query. In the next section, we will show the results of various threshold settings using text-based evaluation and then compare

Given a query q with description profile $v(q)$ and m document instances
For each starting page h, initialize an agent a, where
    the search goal E_a is assigned with $v(q)$,
    the candidate set C_a is initialized with the outgoing links, O_h
Repeat
    For each agent a:
        For each hyperlink $h \in C_a$
            evaluate the exploring potential of the hyperlink λ_{h,E_a}
            If $\lambda_{h,E_a} > \epsilon$ then reproduce a new agent a' with
              search goal $E_{a'} \leftarrow E_a$ and
              candidate set $C_{a'} \leftarrow O_h$
Until No more hyperlink is left to be explored or N hyperlinks are visited

Figure 2: **The Basic Site Traversing Algorithm**

them to breadth-first search (with threshold $\epsilon = 0$) by measuring their precisions and search depths.

2.1 Text-based Evaluation

The text-based evaluation we adopt here uses the consine measure of two documents in the vector space model. In this model, the association measure between a document d and a query q is computed by the inner product of the document vector $v(d)$ and the query vector $v(q)$ as follows:

$$S_{d,q} = v(d) \cdot v(q) = \sum_{t \in d \cap q} w_{d,t} w_{q,t} \tag{1}$$

where $w_{d,t}$ is the weight of term t in document d and is computed by multiplying the augmented normalized term frequency with the inverse document frequency (TFxIDF)[8].

Here we use S_{h,E_a} as the exploring potential of a hyperlink λ_{h,E_a}. In fact, when measuring the text-similarity of a hyperlink h to the search goal, not only the text content of h itself can be used, but also the text contents of the hypertexts that are pointed by h. However, it is expensive to evaluate such a value by pre-fetching all contents of outgoing links.

Table 1 shows the performance comparison of four experiment sets (with threshold ϵ set to 0, 0.1, 0.2 and 0.3) over thirteen queries. The performance comparison include several metrics in three parts, including the total number of traversed pages (defined as V), the top 30 pages (defined as T) ranked by text-similarities to the search goal, and the relevant pages in the top 30 pages (defined as R). In each of the three parts, the longest (LSD) and the average (ASD) search depth are measured. For the top 30 pages, we also examine their average text-similarities and precisions to show the final ranking results.

	ϵ=0	ϵ=0.1	ϵ=0.2	ϵ=0.3
FQ for 200	0/13	2/13	8/13	9/13
No. of Pages	200	178	90	78
LSD in V	1.92	3.23	2.92	2.23
ASD in V	1.55	2.24	1.87	1.36
FQ for 30	0/13	1/13	6/13	8/13
Avg. S_{h,E_a} in T	0.25	0.32	0.35	0.31
LSD in T	1.85	3.15	2.92	1.85
ASD in T	1.44	2.16	1.76	1.22
Precision(T)	0.55	0.63	0.56	0.62
LSD in R	1.77	3.00	3.14	3.00
ASD in R	1.44	2.06	1.94	1.77

Table 1: **Performance comparison of text-based evaluation method (from 13 queries).**

From the traversed page set V, we found that for higher thresholds, the discovery process terminates earlier before 200 pages are traversed. Meanwhile, the search depths are shorter in average. However, when we examine the performance of individual query, the search depths of high thresholds are actually much longer than those of low thresholds for some queries. Such effects are shown in Table 2, where the results of 5 queries that have the largest numbers of traversed pages among the 13 queries are averaged. The decreasing search depth effect vanished when averaged among 13 queries. This is because most queries terminate prematurely when high threshold is set such that the average depth is low. Indeed, half of the 13 queries have traversed pages less than 30 hyperlinks when the threshold is set to 0.3 as shown in the failed queries (FQ) in Table 1. One may notice that the average search depth of the 5 queries is shorter than

	ϵ=0	ϵ=0.1	ϵ=0.2	ϵ=0.3
No. of Pages	200	200	200	196
LSD in V	1.8	2.0	2.6	4.4
ASD Length in V	1.5	1.7	2.0	2.4
Avg. S_{h,E_a} in T	0.32	0.40	0.41	0.41
LSD in T	1.6	2.0	2.6	3.4
ASD in T	1.2	1.6	1.7	2.1
Precision(T)	0.53	0.62	0.64	0.62
LSD in R	1.6	1.8	2.2	3.0
ASD in R	1.4	1.4	1.5	1.8

Table 2: **Performance comparison of text-based evaluation method (from 5 queries that have the largest numbers of traversed pages).**

that of total 13 queries when the threshold is set to 0.1. This is because 0.1 is too low for these 5 queries such that the bound (200) is reached early. In general, the breadth-first search method ($\epsilon = 0$) always explores documents that are adjacent to the starting points. Thus, the search depth is the lowest among the four experiment sets.

Next, comparing the average search depth in the top 30 pages (T) with that in the traversed pages (V), the comparable values tell us the existence of high similarity pages at remote distance. Table 1 also shows the user assessment of the top 30 pages' relevance. The result shows that precision of low thresholds (0.1) is no less than that of high threshold even though hyperlinks with lower text-similarities are traversed. Meanwhile, the precision of high thresholds (0.3) is also no less than that of low thresholds even though the discovery process terminates earlier for higher thresholds. A similar result can also be seen in Table 2, where the precision of the 5 queries is about the same for various thresholds. This implies that the dominating factor of precision relies on the final ranking function.

2.2 Popularity and Richness Evaluation

The second method we propose for hyperlink exploring potential is the content popularity and richness (CPR) evaluation for hyperlinks. In Section 2.1, the selection of a hyperlink to be explored is based on the text relevance, i.e. the similarity measure between the hypertext and the search goal. Nonetheless, we can take advantage of the link information provided in a hypertext. Hence, we propose the content popularity and richness (CPR) method to estimate the potential of a hyperlink. The CPR method is based on three tips of a hypertext, including text content, popularity and richness. First, the text content of a hypertext h can be used to compute its text relevance by S_{h,E_a}, as defined in Eq. (1). Second, the popularity of the hyperlink, pop_h, is defined as a function of the hypertext's incoming links, I_h, proportional to the number of incoming links.

$$pop_h = \begin{cases} \alpha_l + (1-\alpha_l) \cdot \frac{log|I_h|+1}{log(M+1)} & \text{if} \quad |I_h| < M \\ 1 & \text{if} \quad |I_h| \geq M \end{cases}$$

Likewise, the richness of the hypertext, $rich_h$, is defined as a function of the hypertext's outgoing links, O_h, proportional to the number of outgoing links.

$$rich_h = \begin{cases} \beta_l + (1-\beta_l) \cdot \frac{log|O_h|+1}{log(L+1)} & \text{if} \quad |O_h| < L \\ 1 & \text{if} \quad |O_h| \geq L \end{cases}$$

Thus, the evaluation of the exploring potential of hypertext h for a given evaluation criterion E_q, EP_{h,E_q}, is computed by:

$$EP_{h,E_q} = S_{h,E_q} \cdot pop_h \cdot rich_h \qquad (2)$$

Note that pop_h ranges from α_l to 1 with α_l varying decreasingly according to the search depth. Similarly, the richness $rich_h$ is normalized in a range of β_l to 1 with β_l varying increasingly to the search depth. At start, α_l is assigned to 0.7 and β_l is assigned to 0.3 in order to emphasize the importance of richness (such that we can traverse further). As the search process progresses, α_l is decreased such that pop_h varies in a larger interval to emphasize the effect of popularity (to seek for some authority pages). Meanwhile, β_l is increased such that the effect of $rich_h$ lessens. In other words, the exploration focuses on *collection sites* (pages with a lot of outgoing links) at initial and gradually changes the focus to authority pages (pages with a lot of incoming links) as search progresses. The consideration here is that since the search goal itself may not be 100% precise, we can not simply rely on the evaluation of text similarity. Thus, popularity and richness of a hypertext is introduced as assistances in the evaluation of exploring potential.

Table 3 and 4 shows the performance of the CPR-based evaluation at various thresholds (0.05, 0.1, 0.2) for 13 and 5 queries as discussed above. The performance of search depths for various queries is similar to that of text-based evaluation. as well as precision performance. No obvious relationship between precisions and thresholds exists. On average, the number of queries that have traversed pages less than 200 are increased such that the average traversed pages of 13 queries are decreased accordingly. Comparing two cases from CPR-based evaluation $\epsilon = 0.05$ and text-based evaluation $\epsilon = 0.1$, an overall resemblance of the results is shown with favorable search depths for the CPR-based evaluation. Thus, CPR-based evaluation can traverse deeper the search space.

Given a query q with description profile $v(q)$ and m document instances
    (1) decide the threshold ϵ
For each starting page h, initialize an agent a, where
    the evaluation criteria E_a is assigned with $v(q)$,
    the candidate set C_a is initialized with the outgoing links O_h
Repeat
    For each agent a:
        For each hyperlink $h \in C_a$
            evaluate the exploring potential of the hyperlink λ_{h,E_a}
            (2)(3) decide whether h is to be explored or not
        For each hyperlink h to be explored
            reproduce a new agent a' with
            and candidate set $C_{a'} \leftarrow O_h$
Until No more hyperlink is left to be explored or N documents are visited

Figure 3: **The Enhanced Site Traversing Algorithm**

	ϵ=0	ϵ=0.05	ϵ=0.1	ϵ=0.2
FQ for 200	0/13	2/13	8/13	11/13
No. of Pages	200	173	96	42
LSD in V	1.92	3.54	2.92	3.00
ASD in V	1.55	2.43	1.90	1.82
FQ for 30	0/13	2/13	4/13	8/13
Avg. S_{h,E_a} in T	0.25	0.32	0.32	0.29
LSD in T	1.85	3.46	2.69	2.77
ASD in T	1.44	2.27	1.88	2.05
Precision(T)	0.55	0.61	0.49	0.63
LSD in R	1.77	2.75	2.67	6.2
ASD in R	1.44	1.93	2.03	4.44

Table 3: **Performance comparison of CPR-based evaluation method (from 13 queries).**

	ϵ=0	ϵ=0.05	ϵ=0.1	ϵ=0.2
No. of Pages	200	200	200	117
LSD in V	1.8	2.2	2.8	6.8
ASD in V	1.5	1.8	2.2	3.7
Avg. S_{h,E_a} in T	0.32	0.41	0.42	0.43
LSD in T	1.6	2.0	2.6	5.3
ASD in T	1.2	1.7	2.1	3.1
Precision(T)	0.53	0.69	0.64	0.76
LSD in R	1.6	2.0	2.7	5.3
ASD in R	1.4	1.7	2.1	3.1

Table 4: **Performance comparison of CPR-based evaluation method (from 5 queries).**

3 The Enhanced Site Traversing Algorithms

From earlier discussions and experiments, precision is limited by the method we use for final ranking. That is, we have to resort to better description of the search goal $v(q)$ in order to improve precision. However, in this paper we focus on improving search depth by employing an enhanced site traversing algorithm. Three features, printed in bold face in Figure 3, are embedded in this algorithm, including threshold prediction, probabilistic selection and forced probabilistic selection. The effects of these three factors are shown in Table 5, where the result of CPR-based evaluation with threshold 0.2 is re-displayed for comparison (column "Fixed ϵ").

The main problem of the basic site traversing algorithm in Section 2 is the premature end such that the discovery process terminates at a search level near the starting points. Both high thresholds and low thresholds can result in this situation. For a high threshold, the traversing stops progress because no hyperlink has exploring potential greater than such a high threshold. For a low threshold, the discovery terminates because too many hyperlinks are selected for traversing at lower search levels near the starting pages, just like breadth-first search ($\epsilon = 0$). In fact, there is no single threshold that can fit all queries.

Indeed, different queries require different thresholds. Thus, choosing an appropriate threshold for each query is the first priority in the enhanced STA. We have tried several ways to estimate a good threshold from the CPR values of known hyperlinkss. The first

	Fixed ϵ	Predict ϵ	Prob	Forced
No. of Pages	77.4	179.0	127.0	200.0
LSD in V	5.0	4.8	4.0	12.0
ASD in V	2.48	2.47	2.42	5.44
Avg. S_{h,E_a} in T	0.45	0.39	0.37	0.37
LSD in T	4.80	4.80	3.80	7.60
ASD in T	3.09	3.05	2.56	5.39
Precision(T)	0.73	0.73	0.77	0.68
LSD in R	5.75	4.80	3.80	7.60
ASD in R	3.62	3.13	2.54	5.17

Table 5: **Performance comparison of three features in the enhanced STA. From left to right, they are fixed threshold, threshold prediction, probabilistic selection, and forced probabilistic selection.**

trial uses the CPR values of the starting points. This idea comes up from the observation that the medium CPR value of the starting hyperlinks when used as the threshold retrieve a reasonable number of hyperlinks in previous experiments. However, the result is not as good as we expect. For some queries, the search depth is increased; for others, it is decreased. An acceptable value we use here is based on the nonzero CPR values of the hyperlinks in the first levels. We use the kth CPR value from these nonzero hyperlinks (sorted by CPR) as the threshold for that query (k is decide by $Max\left\{\frac{L_1}{2}, 40\right\}$, where L_1 is the number of hyperlinks with nonzero CPR at level one). The maximum value of k is limited to 40 to prevent under-estimated thresholds such that it results in low search depth. Note that we can try other predictions using the CPR values of hyperlinks in the second or other levels, but it will be much complex and lose the independence of agents.

Comparing the result of the predicted threshold (column *"Predict ϵ"* in Table 5) to that of CPR-based evaluation with fixed threshold $\epsilon = 0.2$ for all queries (column *"Fixed ϵ"*), the main effect we noticed is the increased hyperlinks that are explored, especially for many queries with less than 200 traversed hyperlinks in the fixed threshold using CPR evaluation. In fact, in addition to the comparatively equal search depths shown in these two approach, the variance of the search depths for all the queries is also much lower when using the predicted threshold. However, the predicted threshold is not as good for some queries that traverse far for fixed threshold $\epsilon = 0.2$. Thus, the second feature, probabilistic selection, is introduced for the second trial.

Using the predicted thresholds, we bravely introduce probabilistic selection into our traversing algorithms. In deterministic environment with a preset threshold (even if it is predicted from above), the longest depth of the explored hyperlinks is restricted. In fact, various traversing paths are just like different ways to get down the hill (Figure 1), but whether or not there exists another similar group of hypertexts is unknown. Therefore, nondeterministic is used to give chances to hyperlinks that have exploring potential less than the threshold. Indeed, probabilistic selection serves as an alternative to jump to another hill by trial and error. To apply probabilistic selection, each agent a first decide the number of hyperlinks to be explored. This value, called n_a, is counted by the number of links that have CPR greater than ϵ. Next, each outgoing link h is assigned a probability, $P[h|a]$:

$$P[h|a] = \frac{e^{\mu\lambda_h}}{\sum_{h \in C_a} e^{\mu\lambda_h}} \tag{3}$$

such that each agent uses a stochastic selector to pick n_a outgoing links according to hyperlinks' probabilities. The result of this probabilistic selection is shown in the second column to the last (Table 5). Comparing to deterministic method, the performance is less stable. Premature termination occurs for some queries such that the total number of explored is decreased. The reason is that the exploring potential decreases for most queries as the search depth grows, thus n_a becomes zero at some level.

To remedy this shortfall, *forced probabilistic selection* is used instead. In this approach, the number of selected hyperlinks to be explored for an agent a is forced to $Min\{lb_a, 10, |C_a|\}$, where lb_a is the number of hyperlinks with CPR greater than the threshold. With this forced probabilistic selection, the search depth is increased significantly, not only in the traversed pages V but also in the top 30 pages T. Due to the forced selection of hyperlinks, the termination condition is bound at the time when 200 pages are traversed. Thus, the search depth is much longer comparing to others. Meanwhile, the most important result from this experiment is that valuable pages also exist in remote sites, which proves our statement that another hill of relevant documents are linked to the starting pages via some paths (Figure 1).

4 Conclusions and Future Work

Finding information on the World Wide Web is a challenge. Users usually have to navigate through Web pages to find new resources by following hyperlinks. As the Web continues to grow, users must traverse more links to find what they are looking for, making it impractical to just wander the Web searching for information. Therefore, automatic traversing tools that explore hyperlinks for users have practical use in life. In this paper, we design hyperlink-based site traversing algorithms to explore the Web for users.

The basic STA described in Section 2 chooses proper hyperlinks for traversing based on the content popularity and richness evaluation. Section 3 introduces threshold prediction and probabilistic selection in the enhanced STA. The traversed hyperlinks are finally ranked by their text similarities to the query.

Performance evaluation for information discovery tools on the Web is a tough work. Precision evaluation often requires users to give relevance feedback while recall is impossible to evaluated on the Web. The measure of search depth of the traversed hyperlinks can be an alternative for hyperlinked based discovery. In Section 2, we show the potential of CPR-based evaluation over simple text-based evaluation measuring in search depth. In fact, due to the uncertain location of relevant pages, we would like set the main search goal of a hyperlink-based discovery tool to fully explore the neighborhood of starting pages. Then, from the traversed hyperlinks, the most promising ones are filtered for users. Therefore, in Section 3, we focus on how to improve the traversing depth by predicting a good threshold for each query, and by introducing nondeterminism into the site traversing algorithm.

The ongoing work which is not reported in this paper, is the adaptive search goal that is employed in the enhanced STA. This approach is adopted because of the phenomenon that CPR values of hyperlinks decrease as we traverse far from the starting pages. Thus, less and less hyperlinks are selected as traversing goes because of the fixed threshold in a query. In order to solve this problem, each agent tries to merge the search goal with its contents and propagate the new search goal to its descendants. Thus, the search goal adapts across agents. The other reason for the adoption of the adaptive search goal is the old consideration that the search goal itself is not 100 percents exactly what the user wants. Hence, slightly modification of the search goal is a chance to get to the search goal.

There are other trials can be done to exploit hyperlinks for automatic discovery. First, expanding neighborhood of the starting pages is one idea that is worth trying. Since not only the hyerlinks existed in the documents are useful, the hypertexts that contains links to the documents are also worth exploring. Thus, including the incoming links or sibling pages can also lead us to some useful pages. Also, if one likes to explore new web servers instead of pages, we can limit the search steps from general page links to site links (anchors that point to pages outside the server we are visiting).

Finally, we would like to conclude that the utilization of a hyperlink-based discovery tool is mainly to explore the possibility of useful Web pages from neighborhood of starting pages. Due to the weak linkage between Web pages, there is no guarantee that traversing can reach to another relevant information cluster. Thus, the design goal of our traversing algorithms is to explore the search space as far as possible. Of course, if our search goal is to discovery new information resources, other approaches such as query-based discovery mechanism may be another alternative. In that case, information filtering becomes the main factor dominating precision.

References

[1] R. Armstrong, D. Freitag, T. Joachims, and T. Mitchell. Webwatcher: A learning apprentice for the world wide web. In *AAAI Spring Symposium*, 1996.

[2] C.H. Chang and C.C. Hsu. Customizable multi-engine search tool with clustering. *Computer Network and ISDN Systems*, 29(8-13):1217–1224, Aug 1997.

[3] S. Chakrabarti et al. Automatic resource compilation by analyzing hyperlink structure and associated text. In *Proc. of the Seventh International WWW Conference*, Brisbane, Australia, Apr 1998.

[4] Y. Li. Towards a qualitative search engine: Hyperlink vector voting. *IEEE Internet Computing*, 2(4):24–29, 1998.

[5] H. Lieberman. Letizia: An agent that assists web browsing. In *Proc. of the International Joint Conference on Artificial Intelligence*, Montreal, Canada, August 1995.

[6] F. Menczer. Arachnid: Adaptive retrieval agents choosing heuristic neighborhoods for inforamtion discovery. In *Proc. 14th Intl. Conf. on Machine Learning*, 1997.

[7] A. Moukas. Amalthaea: Information discovery and filtering using a multiagent evolving ecosystem. In *Proc. of the Conference on Practical Application of Agents and Multiagent Technology*, London, UK, Apr 1996.

[8] G. Salton. *Automatic Text Processing: the transformation, analysis, and retrieval of information by computer*. Addison-Wesley, 1989.

[9] B. Sheth and P. Maes. Evolving agents for personalized information filtering. In *Proc. of the 9th IEEE Conf. on AI for Applications*, pages 345–352, 1993.

[10] Ellen Spertus. Parasite: Mining structural information on the web. In *Proc. of the Sixth International WWW Conference*, Santa Clara, CA, USA, Apr 1997.

[11] T. Yan and H. Garcia-Molina. Sift - a tool for wide-area information dissemination. In *Proc. of the 1995 USENIX Technical Conference*, pages 177–186, 1995.