International Journal on Artificial Intelligence Tools
Vol. 13, No. 1 (2004) 27–44

World Scientific
www.worldscientific.com

STRUCTURAL WEB SEARCH ENGINE

ARASH RAKHSHAN, LAWRENCE B. HOLDER, and DIANE J. COOK

Department of Computer Science and Engineering
University of Texas at Arlington
Box 19015, Arlington, Texas 76019-0015, USA
{*rakhshan,holder,cook*}*@cse.uta.edu*

Received 18 July 2003
Revised 19 October 2003
Accepted 19 October 2003

We present a new approach in web search engines. The web creates new challenges for information retrieval. The vast improvement in information access is not the only advantage resulting from the keyword search. Additionally, much potential exists for analyzing interests and relationships within the structure of the web. The creation of a hyperlink by the author of a web page explicitly represents a relationship between the source and destination pages which demonstrates the hyperlink structure between web pages. Our web search engine searches not only for the keywords in the web pages, but also for the hyperlink structure between them. Comparing the results of structural web search versus keyword-based search indicates an improved ability to access desired information. We also discuss steps toward mining the queries input to the structural web search engine.

Keywords: Web search; hyperlink structure; graph.

1. Introduction

Structural web search is the process of searching the web for a specific hyperlink structure combined with textual content. Sometimes, it is not sufficient to apply purely text-based methods to find a large number of potentially relevant pages. People are likely to surf the web using its graph structure. The current web search engines can be used in order to search for some keywords or some combination of them without forcing any hyperlink structure between web pages. In other words the result of a particular search engine would be a number of hits each containing one web page.

In contrast the result of a structural web search engine (SWSE) is a number of hyperlink graphs, where each node represents a web page containing certain keywords and edges represent hyperlinks between web pages. The engine ensures that these structural hits match the user's structural query. For example, Fig. 1 shows a simple structural query in which the user is looking for a web page on

"UTA" that has a hyperlink to a web page on "UTA Library" and that is linked to from a web page containing "UTA Graduate". The result of structural search for such a query would be the same graph or hyperlink structure that the user has input except that it already has the web pages which satisfy not only the pure keyword search in a text-based search engine but also the hyperlink structure between them. Fig. 2 shows a sample hit returned by SWSE.

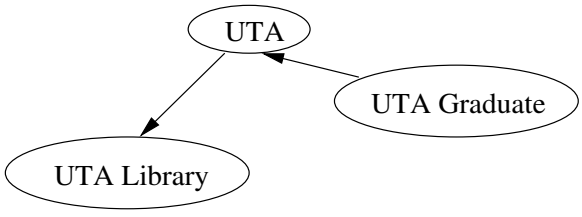

Fig. 1. A sample structural query.

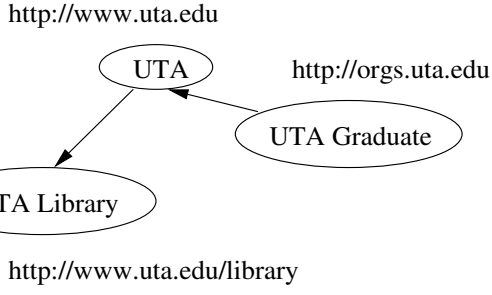

Fig. 2. A sample hit returned by SWSE.

In the next section we describe work related to structural web search. We then discuss the components of our structural web search engine. Next, we present the results of experiments comparing the results of the structural search to keyword-based search engines. We then discuss steps toward mining the queries input to the structural web search engine. We conclude with a discussion of the benefits of structural web search and directions for future research.

2. Related Work

Some existing keyword-based search engines make use of hyperlink structure to improve their performance. However, they do not support queries against the hyperlink structure itself. Some research has investigated querying a graph representation of the web, but has been limited to subsets of the web. We review these approaches here and point out the advantages of our approach.

2.1. *Current use of hyperlink structure*

Much research has been done on approaches to keyword-based web search, but the hyperlink structure has received relatively little attention [3]. While many search engines utilize structure to rank pages, the structure itself is not searched. In the Google search engine, a significant number of maps have been created of these hyperlinks to allow rapid calculation of a web pages' "Page Rank" [1]. PageRank is an excellent way to prioritize the results of web keyword searches. Aside from PageRank and the use of anchor text, Google has several other features. First, it has location information for all hits, and so it makes extensive use of proximity in search. Second, Google keeps track of some visual presentation details such as font size of words. Third, full raw HTML of pages is available in a repository.

Another approach to using hyperlink structure for ranking pages is to identify authoritative pages. The goal is to compile a list of web resources considered the most authoritative for a broad and well-represented topic on the web. At first these lists were constructed either manually or through a combination of human and automated effort. The ARC system [2] for automatically compiling a list of authoritative web resources on any (sufficiently broad) topic operates fully automatically. This technique is embodied in the Clever search engine for finding hub and authority pages. A good hub has many hyperlinks to good authority pages, while a good authority has many hyperlinks from good hub pages.

2.2. *WebSUBDUE*

Viewing the hyperlink structure of the web as a graph has been a strong motivation to improve the result of the search engines. Combination of the keyword extraction and using hyperlink structure has been the main idea of the recent research on the web searches like WebSUBDUE [8].

WebSUBDUE is a tool which retrieves sites corresponding to structures formed by graph-based user queries. WebSUBDUE is enhanced with a knowledge discovery system called SUBDUE [4], that discovers patterns in structural data and performs various types of data mining on the graph. SUBDUE discovers repetitive patterns, or subgraphs, in the graph. Since SUBDUE accepts data in the form of a labeled graph; in WebSUBDUE, the search query and the WWW are represented as labeled graphs, and discovered instances are reported as the results of the query.

For example, Figure 3 shows the SUBDUE representation of the simple keyword-based query "Structural Search Engines." A web page is represented by a vertex labeled as "_page_", and the keywords in the web page are represented as separate vertices labeled with the keyword. The relationships between the keywords and the web page are shown as directed edges starting from the main vertex to the keyword vertex labeled as the attribute, in this case "_word_".

Data collection in WebSUBDUE is performed using a web robot which follows links to pages residing on specified servers. As it traverses a web site, the robot generates a graph file representing the specified site. Once the URLs have been

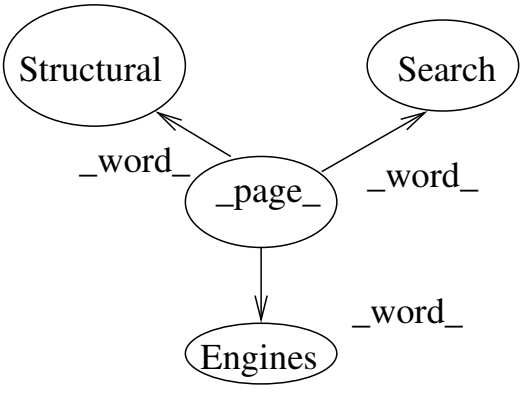

Fig. 3. SUBDUE format of a simple query.

crawled, a labeled graph is generated representing the website. The search engine allows the user to create a graph for a new domain or search an existing graph. New web sites can also be incrementally added to an existing graph. WebSUBDUE invokes SUBDUE to find instances of the graphical query in the graph of the web site. WebSUBDUE reports the graph vertices, edges and corresponding URLs for each discovered instance. SUBDUE's inexact graph match algorithm can be used by WebSUBDUE to find web sites that closely, but not exactly, match the user query.

In WebSUBDUE a sample WWW domain was tested to apply SUBDUE as a structural search engine. A graph format of the sample domain was created and the results were compared to the popular search engine, AltaVista. The power of WebSUBDUE is revealed even in text searching. As it is obvious from the query shown in Figure 3, we can use WebSUBDUE as a regular text-based search engine. The experiments showed WebSUBDUE outperformed text-based search engines like AltaVista. The main advantage of WebSUBDUE compared to other search engines is the ability to search not only for the keywords in a web page, but also for a specific structure between web pages imposed by the user query. Figure 4 shows a possible query where the user is looking for a web page containing "Structural Search Engines" which has a link to another web page containing "WebSUBDUE".

Integrating WordNet, the electronic lexicon database[5], to WebSUBDUE increased the search quality by searching through web pages for not only the exact keywords but also for synonyms of those presented in the user query. The experimental results showed the capability of this search engine compared to popular text-based search engines. These successful experiments led to extend the concepts of structural search and develop the SWSE.

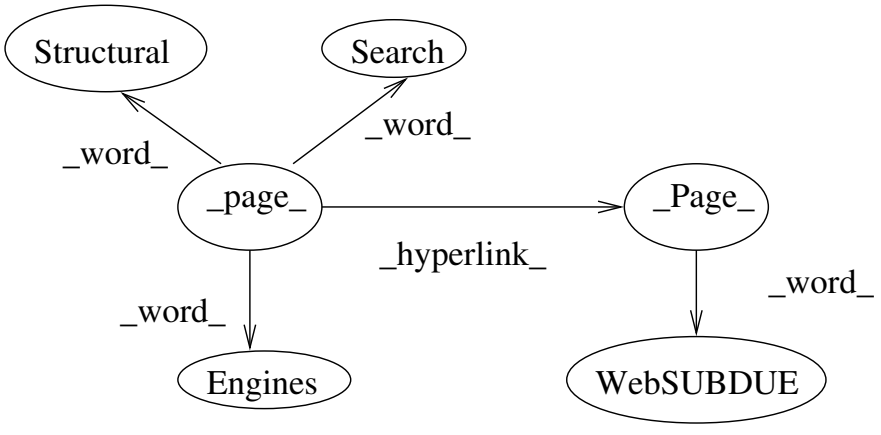

Fig. 4. SUBDUE format of a structured query.

2.3. *SWSE*

In our Structural Web Search Engine (SWSE) in order to search and find the hierarchical structure presented by the user, no data mining tool is required. So there is no need for data preparation. SWSE benefits from the fact that the web has been crawled with the most powerful and trusted web crawler by keyword based search engines like Google. SWSE retrieves only the web pages it needs to crawl, that is the web pages which already have keywords of interest to the user. It searches for the hyperlink structure posed by the user between the web pages it retrieves. The web pages do not have to be in any specific domain. SWSE uses a client-server approach, and it can be used online like any other keyword-based search engine. The SWSE's web-based user-friendly interface allows the user to draw and edit their query in graph form.

3. Structural Web Search

Fig. 5 shows a high-level overview of the Structural Web Search Engine (SWSE). SWSE is implemented in Java and executed as a client-server application on the web. SWSE is available at the following URL: http://swse.uta.edu.

The graph editor is a Java applet running on the client machine. Fig. 6 shows the SWSE interface. The whole query represented as a graph in the client is sent to the server as an object. A Java program listens for the users' query on the server and responds back to the client with the search result. The search result has the same hyperlink structure as the query presented by the user except it has the web pages satisfying not only the keyword search on each node but also the hyperlink structure between those web pages.

Once the server obtains the graph, for each node in the graph, it sends an appropriate query string, including the keywords in the node, to a search engine. In

Given: $G_q = (V, E)$, the query represented as a graph, where
vertices represent pages having one or more keywords,
edges represent hyperlinks.
Return: S = set of matches to G_q
begin
$G_q \leftarrow$ graph obtained from the graph editor
for each vertex $v \in V[G_q]$
$Hits(v) \leftarrow Google\_Hits(keywords(v))$
for each link $e \in E[G_q], e : v_i \rightarrow v_j$
for each $hit(v_i) \in Hits(v_i)$
for each $hit(v_j) \in Hits(v_j)$
if there is a hyperlink such that $hit(v_i) \rightarrow hit(v_j)$
Add $(hit(v_i), hit(v_j))$ to linked list located at $Matrix(i, j)$
else
Remove $hit(v_i)$ from $Hits(v_i)$
$G_i \leftarrow$ a combination of URLs from $Matrix$
while not all possible combinations of URLs have been checked
if G_i is isomorphic to G_q
Add G_i to S
$G_i \leftarrow$ another URL combination
return S
end

Fig. 5. SWSE server algorithm.

response, it receives the HTML pages, including the hits returned by the keyword-based search engine, e.g., Google. The program parses the HTML pages and extracts all of the hits.

Now for each node we have a number of URLs, each of which includes the keywords specified in the node. The server tries to find those URLs that satisfy the structure imposed by the query. In the following sections we describe in detail the components of the SWSE algorithm.

3.1. *Query presentation*

In order to design a structural web search engine we need to have the query presented as a graph, so the desired hyperlink structure can be imposed by the user. The nodes in the graph indicate web pages and the links between the nodes are the hyperlinks between them. We have developed a prototype interface where the users can draw their desired graph-based structure in a user-friendly graph editor and specify the keywords for each web page by inserting the keywords into the graph nodes. We assume the links are between two different nodes (i.e., no self links). This

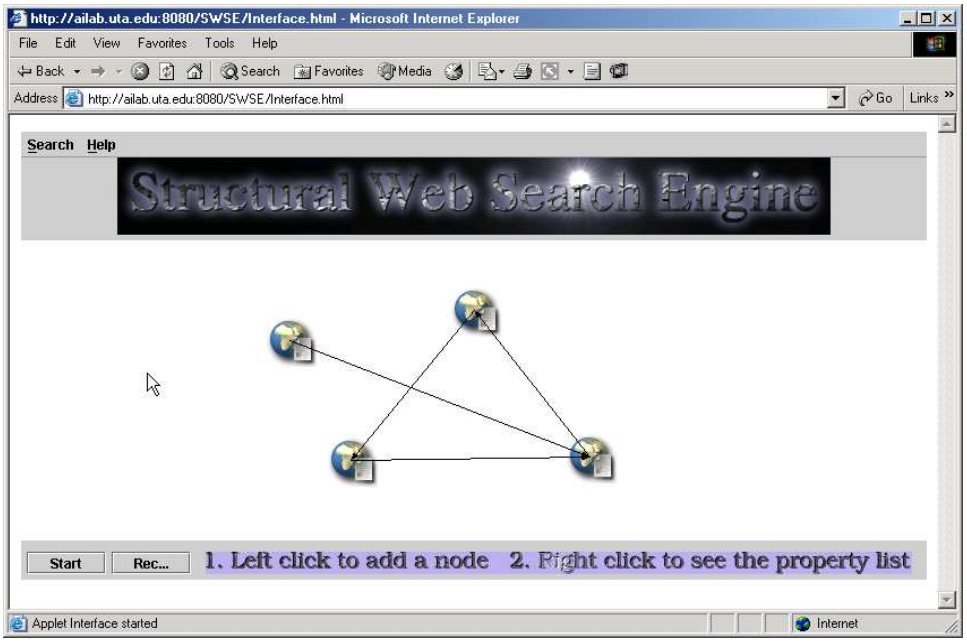

Fig. 6. Screen shot of the graph drawing tool.

is a reasonable assumption that reduces the complexity of the algorithm, because if all nodes in the graph are in the same domain, this means that the links are for navigational purposes, not for inferring useful information [7]. Fig. 6 shows a screen shot of a query in the graph drawing tool presented by a user when accessing the SWSE.

3.2. *Keyword hits extraction*

The SWSE needs to find the web pages which satisfy the keywords provided in each node of the graph query. Any text-based web search engine can be utilized to crawl the web for the specified keywords and pull out the keyword matches. We use Google because of the accuracy of its results. In order to extract the web pages containing the keywords, parsing the pages returned by Google was required. The customized Google search engine was helpful to ease the parse phase, but there is no way to directly send the query string to the server and get the result back. We needed a way to automatically extract the hits. Yahoo's version of the Google search engine provides such a facility. The keywords embedded in nodes of the query are automatically sent to the text-based Google search engine via the proper query string, and the results page is parsed to extract the web pages containing the keywords. Based on this approach the user can put any keyword acceptable by the Google search engine.

This approach limits the result to the number of hits we can get from Yahoo's version of the Google search engine, which is usually less than all the hits the Google search engine can find. We are contented with this number of hits, because the users usually surf the very first hits returned by a keyword-based search engine, but we do parse and extract as many hits as the Google search will provide us. With this approach we can assume that if a user cannot find any result returned by SWSE, he most likely cannot find any hit or any relevant hit by using a keyword-based search engine like Google, even if they surf through all of the hits returned by the search engine.

3.3. *Search for hyperlink structure*

Next, we need to find the specific structure imposed by the user between the web pages extracted from the keyword-based search. For example, if there is a link in the query from a node having "UTA" as its keyword to another node having "UTA Library" (see Fig. 1), we need to pull out all of the links in the hits retrieved from the first node, to see if there is such a hyperlink to a hit retrieved from the second node.

Referring to the algorithm in Fig. 5, for each link $e : v_i \rightarrow v_j$ in the query graph, the server picks the URLs retrieved for node v_i one at a time and extracts the links inside that web page to see if there is a link between that URL and a URL retrieved for node v_j. If there is no such link, it removes the URL from node v_i, because it does not satisfy the structural component of the query.

$$\begin{pmatrix} \text{Null} & \ldots & (URL_{1a}, URL_{na'}) \\ \vdots & (URL_{ik}, URL_{jk'}) & \vdots \\ (URL_{nv}, URL_{1v'}) & \ldots & \text{Null} \end{pmatrix}$$

Fig. 7. Two dimensional array including linked lists of URL pairs.

These URL pairs are stored in a two dimensional array of linked lists (see Fig. 7). We assign a unique number to each node in the graph before sending it to the server. Based on these unique numbers we can put each URL pair in the right place. If the two dimensional array is considered as an adjacency matrix of the graph ($Matrix$), then the $Matrix(i, j)$ element would be a linked list containing pairs of URLs (e.g., $(URL_{ik}, URL_{jk'})$). The first element of the pair is a web page from the node identified uniquely by the row number (e.g., kth hit of node i in our example). This page has a link to the web page located at the second element of the pair, which belongs to the node identified uniquely by the column number (e.g., k'th hit of node j in the example).

After we fill this matrix with information obtained by examining the URLs, we check all possible combinations of URLs to see if they satisfy all structural con-

straints of the query. This step of the algorithm is very efficient, because we do not have to use a full graph isomorphism test. The nodes for a particular combination of URLs are already mapped to the corresponding nodes of the query graph. Therefore, checking for a match requires only checking that the edges are consistent.

3.4. *Ranking the hits*

In the current version of the application, all of the results are being treated the same, that is, there is no ranking involved. The reason is that the web pages returned by the Google search engine have the keywords the user is looking for, and the structure should match the query structure exactly; otherwise, it is not a proper hit for our search. Another reason is that the user has already fixed the nodes by inserting some keywords in them, so no graph isomorphism is involved, and we cannot assume any sort criteria over equally-valid isomorphic hits. One extension would be to allow inexact matches to the query graph, which would involve the use of an inexact graph isomorphism algorithm. The degree of match can be used to rank the results. Another extension of this application would be exploiting WordNet[5], the electronic lexicon database. We can then rank the results based on a similarity criteria for the keywords matching the query.

3.5. *Saving the results*

Another goal of this research is to mine the queries submitted to the SWSE to see if there are prevalent patterns (sub-queries) in the user's queries. Also, we may be able to identify the type of user from sub-patterns in their query, information that may be useful in ranking the hits. To achieve this goal we need to store the queries posted by users including some additional information. The query is stored in two different phases. The first phase occurs exactly after the server gets the original query and before any other processes start, and it stores the query posted by the user without any other information. The second phase occurs after the search process is done and before the hits are sent to the client. In this phase we have more information to store, and for mining purposes we add information about the client URL so we can recognize identifying information about the user (e.g., whether they are coming from a .com or a .edu domain). The number of hits returned by the engine and timing information is included in the second phase. Figure 8 shows the SWSE log for the query presented in figure 1.

3.6. *Converting the data to SUBDUE format*

As we discussed earlier, we use SUBDUE [4] in order to mine the queries stored in the engine. The queries in both phases are stored in text format. We developed a stand alone application to convert this text file to the suitable format for SUBDUE. Fig. 9 shows the graph format accepted by SUBDUE for the query in Figure 1.

```
date = Wed Apr 02 10:55:44 CST 2003
query {
node 1 "uta"
node 2 "uta library"
node 3 "uta graduate"
link 1 2 "null"
link 3 1 "null"
}
hits_per_node {
node 1 40
node 2 40
node 3 40
}
max_hits_per_node = 40
query_hits = 23
Pulling hits out for all nodes takes: 443576 MilliSeconds
Extracting all possible pair of links between two web pages takes: 179 MilliSeconds
request_ip: 192.168.1.1
request_host: outside.ailab.uta.edu
request_Scheme: http
time_ms = 444813
}
```

Fig. 8. SWSE stored query format.

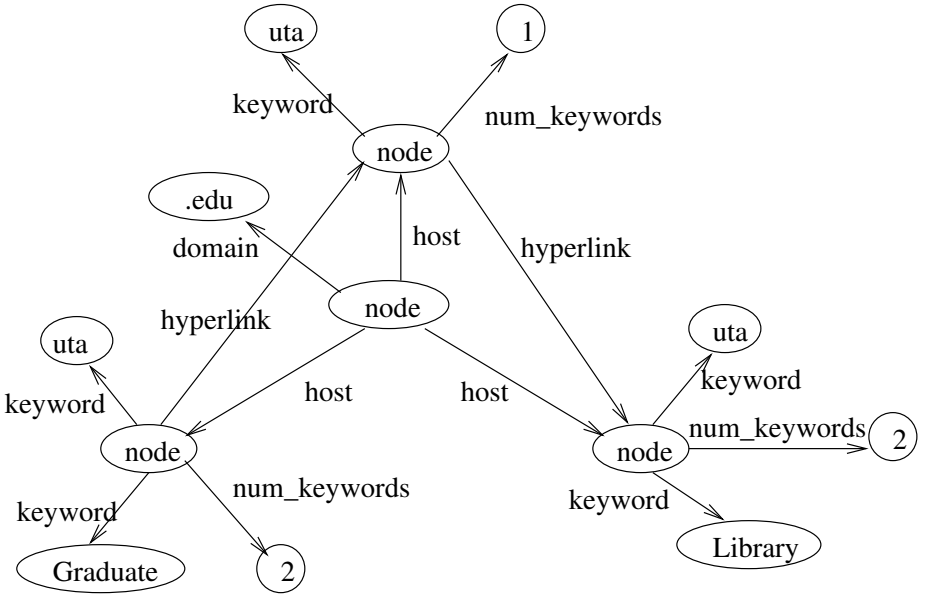

Fig. 9. Graph format of the query after adding more information.

4. Experimental Results

Since this is the first time a structural search is provided; we cannot compare the SWSE results with a similar search engine. To evaluate the capabilities of the

SWSE, we compare query results of SWSE with search results generated using three popular keyword-based search engines: Google, Altavista and Infoseek. Google's advanced search features include the use of link structure of the web to calculate a quality ranking for each web page and utilize them to improve search results. This provides a valuable point of comparison for the results discovered by SWSE.

4.1. *Test query*

As a test query, suppose we are looking for specific information about "Alfred Nobel". He focused on the development of chemical inventions, including such materials as synthetic rubber, synthetic leather, and artificial silk. He became wealthy and purchased an elegant mansion at Avenue Malakoff. He established close contact with Victor Hugo and other writers. Meanwhile his brothers joined in exploiting the oil wells in the Caspian Sea area.

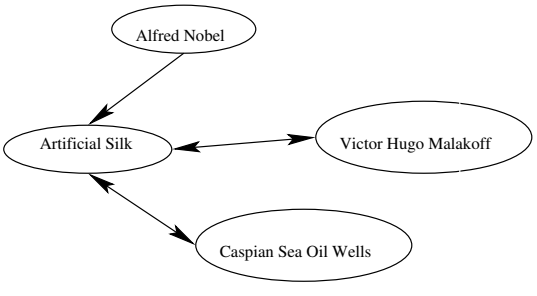

Fig. 10. Experimental query 1.

With the goal of finding a set of URLs relating the people and places described above, we gave SWSE the query shown in Fig. 10. The bidirectional arrow indicates the constraint that links exist in both directions.

Table 1. Experimental Results.

Keyword	**Satisfying URL**	**Google**	**Altavista**	**Infoseek**
Alfred Nobel	http://nobel.se/nobel/alfred-nobel/biographical/index.html	3	11	15
Artificial Silk	http://nobel.se/nobel/alfred-nobel/biographical/sanremo/index.html	493	N/A	N/A
Victor Hugo Malakoff	http://nobel.se/nobel/alfred-nobel/biographical/malakoff/index.html	4	15	32
Caspian Sea Oil Wells	http://nobel.se/nobel/alfred-nobel/biographical/life-work/russia.html	393	482	N/A

The results are summarized in Table 1. The first column of the table shows the keywords inserted in each node. After running the SWSE on the provided query,

the web pages' URLs which satisfy the keywords and hyperlink structure of the presented query are provided in the second column of the table. All of the search hits were retrieved at the time of writing the paper. There may be slight changes if they are tried at a different time. We provided the keywords in the query ("Alfred Nobel Artificial Silk Victor Hugo Caspian Sea Oil Wells") to all of the three keyword-based search engines and no results were found. In some cases even if we removed some of the keywords (e.g., "Oil" or "Wells"), still we did not get any result back or we got some results which were irrelevant to the topics of interest. For example, when we provided another query (the same keywords without "Malakoff" and "Oil Wells") to the Google search engine, we got 35 hits, but none of them included any of the URLs satisfying the structural query. Another experiment we conducted was trying to retrieve the results of keyword-based search engines for the keywords in each node individually. The results of this experiment are provided in the table under "Google", "Altavista" and "Infoseek" columns. The number represents the rank of the URL from the second column in the hits returned by the search engine (N/A means the URL was not in the returned list).

4.2. *Common queries*

We next considered more common examples demonstrating the inability of purely keyword-based approaches to find desire information. Fig. 10 shows a possible query from a user looking for a book by "Dietel" entitled "How to Program" that is pointed to by a "chapter" on "JMX" at "Sun", i.e., the user wants to be sure the book is recommended by the Sun company. In our search engine this query results in a hit relating a Sun Java web page that has a link to a Java book published by Prentice Hall, which is linked to from the web page for Dietel. The Google search returns no matches for the query "sun JMX chapter Deitel How to program".

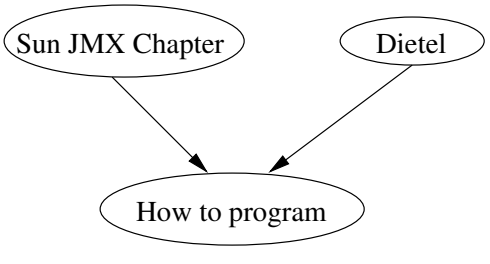

Fig. 11. Experimental query 2.

Query 3 shown in Fig. 12 may represent a student looking for UT campuses and information about visitors and the library at UTA (UT Arlington). The SWSE returned two hits to this query, but these hits were not found by Google, Altavista, or Infoseek using the keyword query "UT UTA campuses visitors library". Finally,

query 4 in Fig. 13 may represent a student looking at UT campuses and information for prospective students in three Texas universities. SWSE returns one hit, which was not found by Google, Altavista, or Infoseek using the keyword query "UT campuses Austin Arlington Dallas Prospective students".

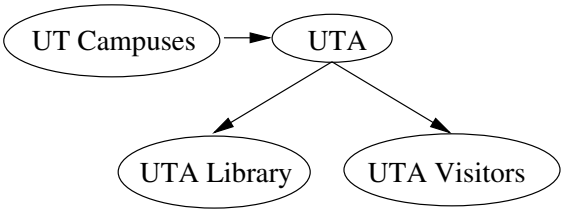

Fig. 12. Experimental query 3.

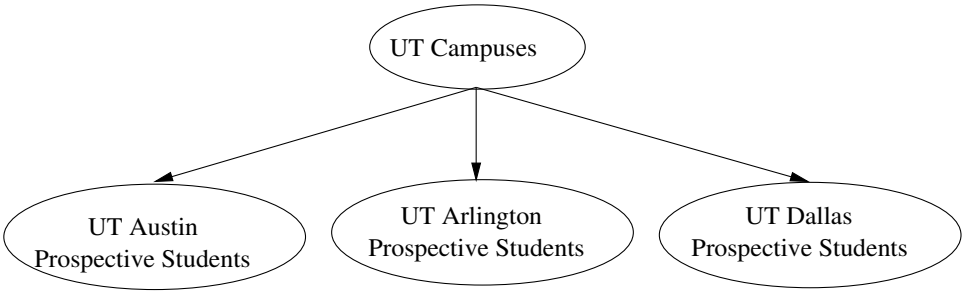

Fig. 13. Experimental query 4.

The results of these experiments indicate the ability of the structural web search engine to more quickly find hits possessing desired relationships among the topics of interest.

4.3. *User experience*

While users found the SWSE interface easy to use, there was some learning necessary on the part of the user in order to retrieve query matches. The SWSE works by performing individual keyword searches for each node in the query. So, if the nodes of a query all have general keywords, then there is a low probability that the retrieved pages would have hyperlinks between them. For example, if a user is looking for information about our graduate school and libraries at our university (UTA), they might enter a query like that in Fig. 14a. However, SWSE would not return any results for this query, because the first 40, or even first 100, hits on "Graduate" and "Library" would most likely not be those associated with "UTA".

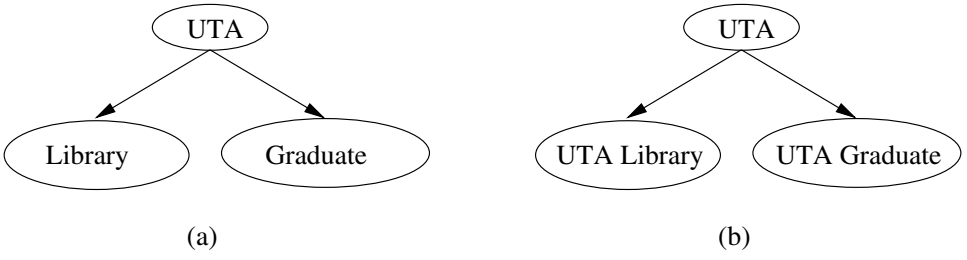

Fig. 14. Two alternative queries for information about graduate school and libraries at UTA.

One approach to this issue is for the user to include the keywords of root nodes to the keywords of the other nodes. For example, the query in Fig. 14b will allow the SWSE to find some hits to the query within the first 40 hits of the individual node keyword searches. A better approach might be to automatically propagate root node keywords to child nodes before executing the query. That is, automatically transform a query like Fig. 14a into that of Fig. 14b. This would be a good direction for future improvements to the search engine.

5. Preliminary Query Mining Experiments

As the number of queries stored by the SWSE increases, we intend to mine the queries for prevalent patterns (sub-queries). To test the potential of this approach, we have applied the graph-based data mining system SUBDUE[4] to a set of artificially-generated queries with known common patterns.

5.1. *Experiment 1*

In this experiment we constructed 100 queries containing the pattern shown in Fig. 15. We also added some random extra structure and keywords to each query. We believe this structure represents a typical level of query complexity based on preliminary data.

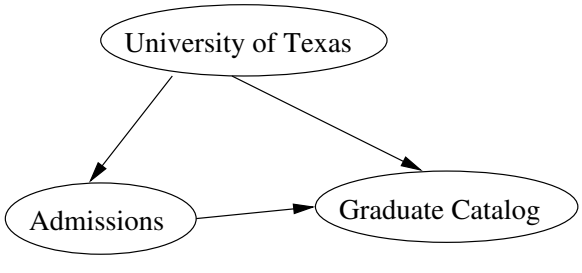

Fig. 15. Common pattern in queries for experiment 1.

As a reminder, the SUBDUE format of the query shown in Fig. 15 would be much more complex than what the subgraph itself looks like. As we discussed earlier each attribute is presented as a single node attached to a main node with a labeled edge (see Fig. 16).

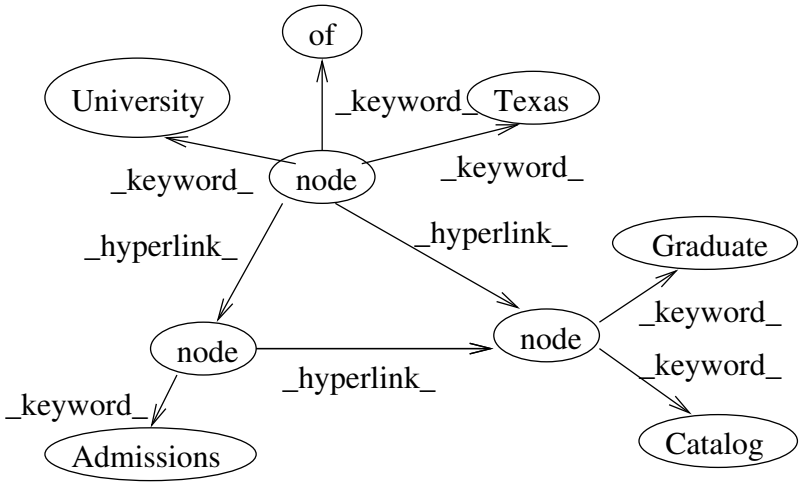

Fig. 16. SUBDUE format of the common pattern for experiment 1.

We ran SUBDUE on this set of 100 graphs and were able to discover the exact pattern from Fig. 15. We also experimented with adding noise (changing keywords and adding/deleting edges) to corrupt the patterns. SUDBUE was able to correctly identify the pattern up to 40% noise levels.

5.2. *Experiment 2*

Experiment 2 tests whether or not we can find a query pattern that can distinguish two types of users. We construct 100 queries in which 50 queries are considered to be from ".com" sites and 50 from ".edu" sites. In order to show that a query is posted by ".edu" or ".com" domain, we add another node to the graph having ".edu" or ".com" as the keyword and a edge with "domain" as the label. This edge connects to a central "node" vertex that is connected to the web page nodes of the query using a "host" edge. We connect this node to all of the nodes in the graph to indicate this property is an entire graph property. See Figures 17 and 18 for examples. We use a simple pattern to distinguish the two types of queries. Queries from ".edu" domains contain the keyword "Admissions", and queries from ".com" domains contain the keyword "Company".

For this task we execute SUBDUE in concept learning mode[6] and input a set of positive examples (".edu" query graphs) and negative examples (".com" query

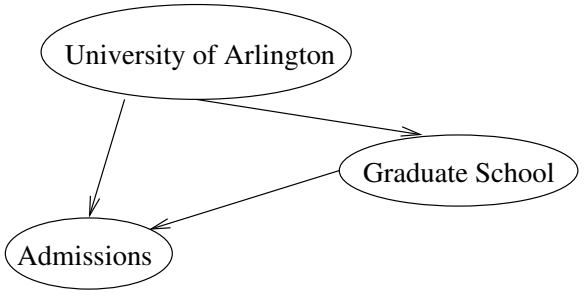

Fig. 17. A sample query for the ".edu" domain.

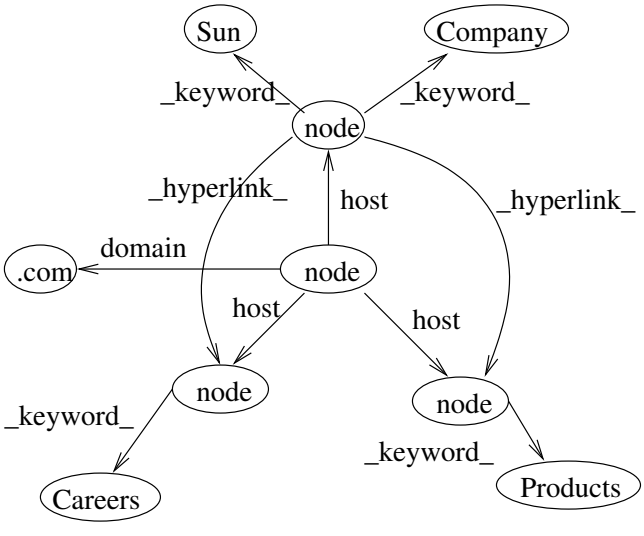

Fig. 18. A sample query for ".com" domain represented in SUBDUE format.

graphs). In this experiment SUBDUE correctly identified the distinguishing feature of a web page containing the keyword "Admissions". We also experimented with varying the number of positive example versus negative examples. Subdue was able to identify the correct distinguishing feature with as few as 20% positive examples, i.e., 20 positive and 80 negative examples.

The results in this section indicate the potential of learning valuable patterns from users' queries posted to the SWSE. As the engine gains in popularity, we hope to compile a large number of real queries on which we can perform similar processing with SUBDUE to discover real patterns.

6. Conclusions and Future Work

People are likely to surf the web using its link graph [1]. Visitors to a web site often "get lost in cyberspace" when they lose the context in which they are browsing and are unsure how to proceed in terms of satisfying their original goal [9]. Structural web search addresses this problem. We developed a search engine which the user can use to obtain information trails (or navigation paths) in response to a single query. The SWSE increases productivity while surfing the web, being more precise than keyword-based search engines and manually navigating the web pages.

Much research has focused on using hyperlink information in some way to enhance web search [2]. Although these systems use hyperlink structure to rank retrieved web pages, they do not perform structural search. In contrast, SWSE performs search to find a structural query combined with textual content. The experimental results reveal the advantage of this approach over a traditional keyword-based search engine when the user is interested in both the hyperlink structure of the web pages and the keywords embedded in those web pages. We intend to further improve the approach by allowing the user to add keywords to the graph edges in order to constrain the anchor text on hyperlinks, and by using inexact graph matching to find close matches and rank the matches by their degree of closeness.

The results described in this paper suggest a number of research directions impacting the areas of machine learning and data mining from graph structure. The mining of the web link structure has intellectual antecedents in the context of graph-based knowledge discovery and data mining systems (e.g., SUBDUE[4]). We intend to collect the structural queries entered by users and apply graph-based data mining to these queries to find common patterns and clusters. SUBDUE is a data mining tool that discovers repetitive substructures in graph-based data. We intend to input our graphical representation of the queries into SUBDUE and discover common patterns between the queries, classify them and find clusters to better understand, predict and optimize typical users' queries. Preliminary results indicate this approach has promise.

Acknowledgments

This work was supported by the National Science Foundation grant 0097517.

References

1. S. Brin and L. Page, The anatomy of a large-scale hypertextual Web search engine, *Computer Networks and ISDN Systems*, 30(1–7):107–117, 1998.
2. S. Chakrabarti, B. Dom, D. Gibson, J. Kleinberg, P. Raghavan, and S. Rajagopalan, Automatic resource list compilation by analyzing hyperlink structure and associated text, *Proceedings of the 7th International World Wide Web Conference*, 1998.
3. S. Chakrabarti, B. Dom, S. Kumar, P. Raghavan, S. Rajagopalan, A. Tomkins, D. Gibson, and J. Kleinberg, Mining the Web's link structure, *Computer*, 32(8):60–67, 1999.
4. D. J. Cook and L. B. Holder, Graph-based data mining, *IEEE Intelligent Systems*, 15(2):32–41, 2000.

5. C. Fellbaum (editor), *WordNet: An Electronic Lexical Database*, MIT Press, 1998.
6. J. Gonzalez, L. Holder, and D. Cook, Graph-based relational concept learning, *Proceedings of the Nineteenth International Conference on Machine Learning*, 2002.
7. J. Kleinberg, Authoritative sources in a hyperlinked environment, *Proceedings of the Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 668–677, January 1998.
8. N. Manocha, D. Cook, and L. Holder, Structural web search using a graph-based discovery system, *Intelligence*, 12(1):20–29, 2001.
9. J. Nielsen, *Designing Web Usability*, New Riders Publishing, 2000.