How Did I Find That? Automatically Constructing Queries from Bookmarked Web Pages and Categories

Chris Staff
University of Malta,
Department of Computer Science and AI,
Malta
cstaff@cs.um.edu.mt

ABSTRACT

We present 'How Did I Find That?' (HDIFT), an algorithm to find web pages related to categories of bookmarks (bookmark folders) or individual bookmarks stored in a user's bookmark (or favorites) file. HDIFT automatically generates a query from the selected bookmarks and categories, submits the query to a third-party search engine, and presents the results to the user. HDIFT's approach is innovative in that we select keywords to generate the query from a bookmarked web page's parents (other web-based documents that contain a link to the bookmarked web page), rather than from the bookmarked web page itself. Our initial limited evaluation results are promising. Volunteers who participated in the evaluation considered 20% of all query results to be relevant and interesting enough to bookmark. Additionally, 56.9% of the queries generated yielded results sets (of at most 10 results) containing at least one interesting and bookmarkable web page.

1. INTRODUCTION

Users who browse the Web store references to pages that they would like to revisit in their Web browser's Bookmark (of Favorites) file [1][2]. Sometimes, some users organise their bookmark file to collect related bookmarked Web pages into the same category, [1][2]. Frequently, bookmark files become disorganised over time, with categories containing references to pages about unrelated topics, and stale references (e.g., to pages that no longer exist) [9]. Some research has been conducted into why users bookmark Web pages and how they use those bookmark files [1][9][2]. There seems to be little research into how a user's bookmarks can be used to automatically find related content on the Web [6].

Manually finding Web pages that are interesting, relevant, and useful enough to bookmark usually takes considerable effort. Users must think of a query that might satisfy a requirement. Search results must be looked at to see if the query has really been satsified. The query may undergo a number of reformulations until a satisfactory Web page is found by a search engine. Once the user has visited the page, the user may bookmark it, to support a re-visit some time in the future. Sometimes a user may decide to search for more information related to a bookmark or a collection of related bookmarks. But all the effort that was put into deriving queries in the first place has been thrown away. Can the user remember what query was used to find each bookmark in the category? [14][5][10][5] store queries submitted during a user session that eventually results in a bookmarked page.

Rather than relying on user generated queries, 'How Did I Find That?' uses a Web page's 'context' to automatically generate a query. The query is submitted to a third-party search engine[1], and a page of up to 10 results is shown to the user. Eventually, 'How Did I Find That?' will be incorporated into a Web browser, and will automatically and periodically search for relevant web pages. For the time being, users can select individual bookmarks or whole categories of bookmarks as the basis of their search. A query is composed from the top ranking terms that occur in the context of the selected bookmarked web pages, and at most ten results are shown to the user. In our evaluation (see section 5), 56.9% of the results sets for 58 automatically generated queries contained at least one relevant result that was also bookmarkable (on average there were 1.72 bookmarkable results per results set).

This paper is organised as follows: Section 2 discusses similar work. We describe our approach to automatically generating queries from Web browser bookmark files in section 3. The evaluation approach and results are described in sections 4 and 5 respectively. Finally, we discuss the results and our future work in section 6.

2. BACKGROUND AND SIMILAR SYSTEMS

There is little research about automatically generating queries from a collection of bookmarks maintained by a user. Web browsers, such as Microsoft Internet Explorer, Mozilla, Mozilla Firefox, and Safari, contain tools for managing interesting pages that a user intends to revisit as bookmarks, but the tools are mostly deficient, unhelpful at reminding users how and in what context the pages had been bookmarked [9]. The bookmark files themselves are not without problems.

[1]For this study, we used Google at `http://www.google.com`.

Although users may organise bookmarks into a hierarchy of folders (or categories, as we call them here), effort is required to maintain them, and they can quickly become out of date or disorganised [1][2]. However, from an adaptivity point of view, bookmarks are useful because they contain documents that a user has found relevant and useful enough at some point to actually keep a record of them [7].

Queries to automatically search the Web are generated for a number of reasons, including helping users find relevant information while browsing [4][3][9][15], acting as a search intermediary [13][14][10][5], or creating and sharing paths through related material [6].

Queries may be automatically generated by identifying terms that describe a user's interest. Systems that help users to find relevant information while browsing typically use terms extracted from the pages that the user has visited recently. HyperContext identifies a "context path" as a collection of pages visited during a "context session" [15]. Nakajima, *et. al.*, defines context as the collection of pages that have been visited between a search page and a bookmarked page [10]. El-Beltagy, *et. al.*, define context as the description extracted from the centroid of a document cluster [7]. Bugeja also automatically extracts terms from web pages that are bookmarked in the same category and uses them to construct a query to find relevant web pages [5]. [14] incorporate an explicit query submitted by the user which is modified following relevance feedback .

The most common approach to identifying significant terms is based on a modified TFIDF [7][10][14][15][5], though others use, for example, a Simple Bayesian Classifier [3] or Association Rules [4]. In TFIDF, terms occurring in documents are ranked according to a weight that takes into account the frequency of term occurrence within a document (Term Frequency) as well as the number of documents in which the term occurs (Document Frequency) [12]. In information retrieval, a term that has a high frequency of occurrence within a particular (relevant) document and that occurs in few documents in the collection is considered to be a good discriminator between relevant and non-relevant documents, so the Inverse Document Frequency (IDF) is used to calculate a term weight. On the other hand, we are more interested in identifying those terms that occur in documents seen recently by a user (or which occur in some cluster or co-occur in some bookmark category) and that are likely to enable us to find more relevant documents. In this case, terms which occur frequently in individual documents *and* in a large number of documents seen by the user are likely to be good descriptors of a user's interest. Like Nakajima, *et. al.* [10], HyperContext extracts terms from documents visited during a "context session", but unlike Nakajima, *et. al.*, it does not require that a 'context' begins with a search page (so it can operate with no user input), and it does not count all the terms that occur in the documents. Rather, it identifies segments called "context blocks" around the links that are followed by users. Terms that are used in the automatically generated query occur frequently in context blocks as well as in a large number of documents visited during the context session [15].

3. APPROACH

Let's say that we have a Web browser bookmark category containing a number of bookmarks that have been manually bookmarked and organised by a user. If we can find out what it is about these Web pages that makes them related, then we should be able to generate a query that would return other related documents in a query results set.

We could have identified frequently occurring terms that occur in most of the documents contained in a bookmark category (or, simply, the highest occurring terms if a single bookmarked document is selected by the user), or created a cluster centroid from the category members, but instead we have chosen to apply a modified HyperContext [15] approach. In HyperContext, an accessed document is *interpreted* in the context of the link, or parent, that was followed to access it. An interpretation is a collection of keywords describing the document that are relevant in that context [15]. To generate a query from a bookmark category, we can create a category centroid based on the bookmarks' interpretations. HyperContext does not generate queries from bookmark files, and in 'How Did We Find That?' we do not have access to a user's path of traversal that eventually resulted in a document being bookmarked.

We have modified the HyperContext approach for 'How Did I Find That?' to look at a number of Web pages that contain a link to the bookmarked document (parents), because we do not know which link was followed by the user before the page that was bookmarked was accessed. We extract the region in each parent for each parent of each bookmark that contains the source of the link and create a centroid representation. We rank the centroid's (stemmed) terms (after stop-words have been removed). In this way, the query is automatically created from a bookmarked page's parents, rather than from the bookmarked page itself.

3.1 Processing Steps

HDIFT is a Python 2.3.4 program that interfaces with the Extended Boolean document indexing and retrieval system SWISH-E[2] and the Google Web API[3].

As shown in Fig. 1, there are six processing steps involved. We first enable a user to select a bookmark or category about which she would like to find similar documents. We then find twenty of each of the selected documents' parents, and process them to find the 'context blocks': the regions that contain links to the selected bookmarked web page. We merge the context blocks to create a centroid, and construct a query from the 10 top-ranking terms in the centroid. The query is submitted to Google through the API, and the results are displayed to the user.

The first step assumes that a user has chosen a bookmark or category for which she would like to find relevant Web pages. HDIFT then interfaces with Google through the Google Web API to identify the parents of a Web page (using the 'link:' operator in the Google query). As the processing overhead to identify a context block in a parent can be quite high, we limit the number of parents per bookmarked page to 20. As

[2] `http://www.swish-e.org`

[3] The Google API is available from `http://www.google.com/apis`. There is a limit of 1000 queries and 1000 results per query per day using the API.

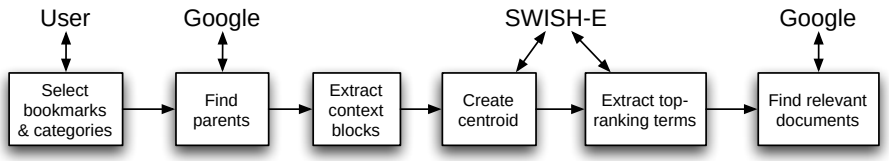

Figure 1: 'How Did I Find That?' processing steps.

these are ranked according to Google's PageRank algorithm [11], we have identified the 20 pages most likely to be used to access the bookmarked page.

The parents must be English-language documents. We define a document to be an English-language document if at least 5% of the terms in the document (ignoring HTML tags) are English language stopwords. If less than 5% of the terms in the document are English language stopwords, then we reject the document[4]. The next task is to identify the context blocks, the regions in each parent that contain a link to the bookmarked web page. We search through the text to identify the location of the link anchor, and once we find it, we scan the document to identify the context block. The context block is an HTML chunk between delimiters. Examples of delimiters are the heading tags (<H1>, <H2>, etc.), horizontal rule (<HR>), and table elements (<TR>, <TD>). In addition, we limit the context block to 100 words or less.

Once a context block is identified it is written to a text file for subsequent indexing. If a user selects a category of bookmarked pages, then rather than just one bookmarked page, each context block from each parent of each bookmark in the category is written to a separate text file for indexing. Once all the context blocks have been identified, SWISH-E is used to generate a central index. Stopwords are removed during the indexing process, and the terms are stemmed using SWISH-E's stemming option. Eventually, when the top-ranking terms are used in a Google query, the original terms, rather than their stems, must be submitted. Therefore, we also keep a map of the stems to their originals.

Once the central index has been created, we can process it to create a centroid representation of the cluster of context blocks to extract n terms with the highest rank to construct the Google query. SWISH-E provides a mechanism whereby all term information can be accessed. The frequency information is in the form of a set of document IDs and an indication of where in the document the term has occurred (e.g., in a title or heading, etc.) for each occurrence of a term. We convert the frequency information into a term weight, which takes into account each term's Document Frequency (expressed as a percentage of contexts of bookmarked web pages in the collection in which the term occurs), and derive an average weight for the term. We then extract the top 10 term stems with the highest average weights (after removing terms shorter than 2 characters in length), look up the original terms for each stem (ORing the terms if the stem has been mapped to more than one original), and concatenate the terms into a query string. A completed query string might look like 'firefox extension OR extensions download OR downloading OR downloads OR downloader tab OR tabbed OR tabs opera mozilla web windows OR window page OR pages links OR link google'.

Finally, we submit the query to the Google Web API using PyGoogle[5], and display the top 10 results to the user.

4. DATA COLLECTION

We evaluated 'How Did I Find That?' during September-October 2006 by asking volunteers to anonymously upload their Bookmark (or Favorites) files to the HDIFT Web server[6] and then to select one or more bookmarks or categories of bookmarks to run HDIFT against. Volunteers were recruited by e-mail from undergraduate and postgraduate students in the Department of Computer Science and AI at the University of Malta, the WebIR mailing list[7], and lecturers and students in all disciplines at the University of Malta. A total of 20 unique bookmark files were uploaded. Once a bookmark file was uploaded, each volunteer was given a unique HDIFT ID to enable them to return to view their own results and give feedback.

Once a selection was made, a server-side script extracted the URLs associated with the bookmark or category and prepared a file for input to the HDIFT algorithm. The HDIFT algorithm generated a query and submitted it to Google, using the method described in section 3.1, and stored the results of the query on the server. As HDIFT generates queries based on frequently occurring terms that appear in the context blocks of the bookmarked web pages' parents, if a page does not have parents (a request to Google with the 'link:' operator finds no documents) then it may not be possible to generate a query. In addition, a generated query may have no results returned by Google. Each query has a maximum of 10 results, equivalent to the first page of results that Google returns. It is possible that there are more than 10 results, but given Jansen's *et. al.* findings [8] we assume that for the majority of users, there should be relevant information in the first 10 results. There are five feedback levels: 'I would bookmark this', if the page was relevant and the user would bookmark it; 'Relevant, but I wouldn't bookmark it'; 'Not given yet', a default value to indicate that the user has not evaluated the result; 'Error opening page', because it is possible for a page to be off-line or removed; and 'Not relevant'. Evaluators were asked to try to evaluate as many results as they could. Once feedback had been submitted, it was not possible to modify it.

[4]Note that we do not perform a language test on bookmarked Web pages themselves.

[5]PyGoogle is a Google Web API wrapper for Python available from `http://pygoogle.sourceforge.net/`

[6]`http://poseidon.cs.um.edu.mt/~csta1/hdift/uploadbk.php`.

[7]`http://groups.yahoo.com/group/webir/`

Volunteers could select as many categories and bookmarks as they liked. In addition, volunteers could return on up to five separate occasions to make additional selections. Finally, there was no way to limit file uploads (unless we ran out of HDIFT IDs), so it is possible for the same volunteer to upload the same bookmarks file several times and make more selections. We could tell if the bookmark files were bit identical that they were probably submitted by the same person, but this happened only once, and selections were only made off one of the duplicate bookmark files anyway.

In all we collected 20 bookmark files, at least one selection was made off 16 of them, 267 valid queries were generated in all (a valid query has at least one Google result) and feedback was given on 58 of the valid queries (21.7%). In the next section, we evaluate the results of the valid queries.

5. RESULTS

Although only limited feedback was obtained (only 20 bookmarks files submitted, and feedback on results given on only 58 out of 267 valid queries), the results are promising. A detailed breakdown is given later in this section, but overall, Google returned 502 results in all for the 58 queries. Of these, 19.9% were considered good enough to bookmark, 23.9% were relevant but the user did not think they were worth bookmarking, and 38.5% were not relevant. Feedback was not given at all on 16.5% of the results, and there were HTTP problems with the remaining 1.2% (Google would have returned the page in the results set, but the page may have been non-responsive when the user tried to access it). Furthermore, the results sets for 56.9% of the queries on which feedback was given contained at least one result that the user considered worthy of bookmarking.

5.1 Analysis of the Bookmark Files and Overall Selections

In all, 20 bookmark files were uploaded, with users making selections of categories and individual bookmarks from 16, and giving feedback on the results of queries generated from selections made from 7 bookmark files. In all, 145 categories, containing an average of 16 bookmarks each, and 331 individual bookmarks were selected from 16 bookmark files, yielding a total of 476 attempts to generate a query. There were 267 valid queries generated (queries which when submitted to Google returned at least one result), meaning that there were 209 failed attempts. The failed attempts can be differentiated between failure to generate a query, and failure to obtain results following a query. We generated a query but failed to obtain results 11 times for category selections and 18 times for individual bookmark selection. We failed to generate a query 13 times for selected categories, and 167 times for selected individual bookmarks. We explain the high failure rate for individual bookmarks in subsection 5.2.

There were 121 valid queries (i.e., queries with results) from selected bookmark categories, and 146 valid queries from selected individual bookmarks. Feedback was given on the results of 58 queries in all, 31 queries generated from categories and 27 generated from individual bookmarks. In all, 502 results were provided for the 58 queries, and feedback was given on 413 of them.

As shown in Table 1, more results overall are considered bookmarkable when the query is generated from a category, rather than from an individual bookmark. The number of non-relevant results increases when the query is generated from individual bookmarks. On average, 1.72 results per query were considered bookmarkable, with 0 the lowest and 6 the greatest number of bookmarkable results per query. We also measured the degree of satisfaction with the results by the rank in which Google returned results (Fig. 2).

We can see that the number of non-relevant results (Fig. 2c) seems to be unaffected by rank. The number of bookmarkable results (Fig. 2a) is highest at P1 independently of whether the query is generated from a category or an individual bookmark.

5.2 Discussion of Results

Apart from the low number of responses to the request for volunteers to assist with the evaluation of HDIFT, and the large number of user selections from bookmark files made without feedback being given on the results, the most significant figure is the failure to generate a query for a high number of selections made from bookmark files. In all, out of 476 bookmark selections made, 209 (43.9%) failed to result in a generated query. This was a far greater problem for individual bookmarks than for categories of bookmarks, with only 13 failed queries out of 145 (9%) selected categories, but 167 failed queries out of 331 (50.5%) selected individual bookmarks. We analysed the reasons for the high percentage of failed queries for individual bookmarks and discovered that 47 (28.1%) HTTP requests for the Web page resulted in an error code; the servers hosting 6 (3.6%) pages were unresponsive at the time of the request, and for the remaining 114 (68.3%), a 'link:' request to Google for documents linking to the page yielded no results.

Feedback was given on the results of 58 queries. In all there were 413 Google results generated on which users gave feedback for the 58 queries (83 additional results were not given feedback on, and six results gave the user an HTTP error code on access). Of these 413 results, only 7 web pages in the results already existed in the users bookmark file (in the same category or individual bookmark that was used as a basis to generate the query). The feedback given on these results have not been separated out from the global results given above. Interestingly, and even though the users had already bookmarked these web pages for the selected category or individual bookmark, only two of the results were considered interesting and bookmarkable. Another two results were considered relevant but not bookmarkable, and another two were considered not relevant! User feedback was not given on the final, already bookmarked, result. In future, we will remove Google results that already exist in the same user selection.

6. FUTURE WORK AND CONCLUSIONS

We have described 'How Did I Find That?' (HDIFT), an algorithm to find Web-based material that is related to Web pages that a user has bookmarked in the past. A user can select a category of bookmarked web pages, or individual bookmarked web pages from their personal bookmark file and HDIFT will automatically generate a query based on the selection, submit the query to Google, and present the

Table 1: Overall Feedback Levels

	'I would bookmark this'	'Relevant, but I wouldn't bookmark it'	'Not relevant'
Total = 413	100	120	193
Total %	24.2	29.1	46.7
Queries from Categories only %	27.4	31.5	41.1
Queries from Individual Bookmarks only %	21.3	26.9	51.9

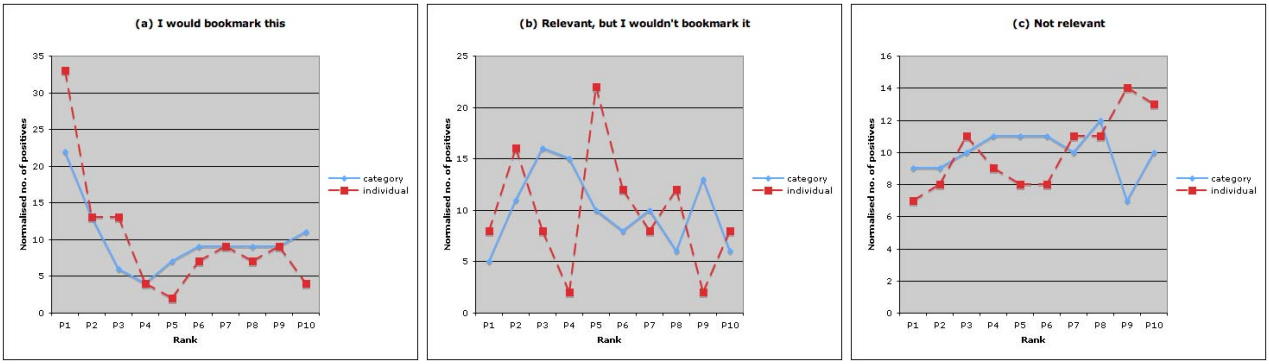

Figure 2: Feedback levels according to results rank.

results to the user. Rather than generating the query directly from the bookmarked web pages, we download up to 20 of the document's parents (found using Google's 'link:' modifier) and create a centroid representation of the context. We use the centroid representation to construct a query. Although the number of participants in the evaluation was low, results are promising and indicate that HDIFT is able to find relevant bookmarkable web pages. The results appear to be equally good for queries generated from categories of bookmarks and from individual bookmarks, although an issue still to be resolved is the inability to generate a query for an individual bookmark if Google cannot find any parents for it.

We intend to conduct studies with a smaller group of people to compare HDIFT with the results of extracting terms from the centroid of documents in category, and manually generated queries (by the participants) from the same category. In the same study, we will identify synonyms in the bag-of-words representation of the centroid and the generated query so they can be ORed in the query. We intend to analyse the bookmark files for information about the frequency with which bookmarks are added; the order in which they are added; average gaps between returning to a category to add new bookmarks; the number of stale links in bookmark files, etc. Finally, we intend to utilise the HDIFT algorithm to perform automatic bookmark classification to help users keep bookmark files automatically organised.

7. REFERENCES

[1] D. Abrams and R. Baecker. How people use WWW bookmarks. In *CHI '97: CHI '97 extended abstracts on Human factors in computing systems*, pages 341–342, New York, NY, USA, 1997. ACM Press.

[2] D. Abrams, R. Baecker, and M. Chignell. Information archiving with bookmarks: personal web space construction and organization. In *CHI '98: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 41–48, New York, NY, USA, 1998. ACM Press/Addison-Wesley Publishing Co.

[3] D. Billsus and M. Pazzani. Learning probabilistic user models. In *Proceedings of the Workshop on Machine Learning for User Models, International Conference on User Modeling*. Springer-Verlag, 1997.

[4] D. Boley, M. Gini, R. Gross, E.-H. Han, K. Hastings, G. Karypis, V. Kumar, B. Mobasher, and J. Moore. Document categorization and query generation on the World Wide Web using WebACE. *AI Review*, 13(5-6):365–391, 1999.

[5] I. Bugeja. Managing WWW browser's bookmarks and history (a Firefox extension). Final year project report, Department of Computer Science & AI, University of Malta, 2006.

[6] P. Dave, I. Paul Logasa Bogen, U. P. Karadkar, L. Francisco-Revilla, R. Furuta, and F. Shipman. Dynamically growing hypertext collections. In *HYPERTEXT '04: Proceedings of the fifteenth ACM conference on Hypertext and hypermedia*, pages 171–180, New York, NY, USA, 2004. ACM Press.

[7] S. R. El-Beltagy, W. Hall, D. D. Roure, and L. Carr. Linking in context. In *HYPERTEXT '01: Proceedings of the twelfth ACM conference on Hypertext and Hypermedia*, pages 151–160, New York, NY, USA, 2001. ACM Press.

[8] B. J. Jansen, A. Spink, J. Bateman, and T. Saracevic. Real life information retrieval: a study of user queries on the web. *SIGIR Forum*, 32(1):5–17, 1998.

[9] W. Jones, H. Bruce, and S. Dumais. Keeping and re-finding information on the web: What do people do and what do they need? In *ASIST 2004 Annual Meeting, Managing and Enhancing Information: Cultures and Conflicts*, November 2004.

[10] S. Nakajima, S. Kinoshita, and K. Tanaka. Context-dependent information exploration. In *Proceeding of the the 11th World Wide Web Conference (WWW2002)*, New York, NY, USA, 2002. ACM Press.

[11] L. Page, S. Brin, R. Motwani, and T. Winograd. The pagerank citation ranking: Bringing order to the web. Technical report, Stanford Digital Library Technologies Project, 1998.

[12] G. Salton and M. J. McGill. *Introduction to Modern Information Retrieval*. McGraw-Hill, New York, 1983.

[13] A. Sieg, B. Mobasher, S. Lytinen, and R. Burke. Concept based query enhancement in the ARCH search agent, 2003.

[14] G. Somlo and A. E. Howe. Querytracker: An agent for tracking persistent information needs. In *AAMAS '04: Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 488–495, Washington, DC, USA, 2004. IEEE Computer Society.

[15] C. Staff. *HyperContext: A Framework for Adaptive and Adaptable Hypertext*. PhD thesis, University of Sussex, 2001.