Automatic Classification of Web Pages into Bookmark Categories[*]

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

ABSTRACT

We describe a technique to automatically classify a web page into an existing bookmark category whenever a user decides to bookmark a page. HyperBK compares a bag-of-words representation of the page to descriptions of categories in the user's bookmark file. Unlike default web browser dialogs in which the user may be presented with the category into which he or she saved the last bookmarked file, HyperBK also offers the category most similar to the page being bookmarked. The user can opt to save the page to the last category used; create a new category; or save the page elsewhere. In an evaluation, the user's preferred category was offered on average 67% of the time

1. INTRODUCTION

Bookmark management systems that can help classify bookmarked web pages, track web pages that have moved since they were bookmarked, help a user to find web pages similar to pages that were bookmarked, and that generally assist with their own organisation are becoming increasingly important. Recent surveys indicate that a user's bookmark file contains on average 184 entries [4], and that approximately 73.7% of pages visited are page revisits [8], with interaction through either a bookmark file, or the history list of recently visited sites, or the browser's back button being the most common ways of revisiting a page. Modern Web browsing software, such as Mozilla, Microsoft Internet Explorer, and Safari, provide only limited support for automatic management of bookmarked web pages (see section 2). Even less support is provided for navigating through the list of recently visited web pages to enable a user to return to a recently visited page. Bugeja's HyperBK [3] addresses some of the issues. A moved bookmarked Web page can be tracked via a third-party search engine. A user can be reminded of the query that had been used to find a web page before it was bookmarked, or HyperBK can suggest a query to use to find web pages similar to a category of bookmarked web pages. HyperBK provides a variety of perspectives or views to potentially make it easier for a user to find an entry in the list of recently visited web pages. Finally, HyperBK automatically suggests a bookmark category into which to store a web page whenever the user requests to bookmark a web page. This last feature is the subject of this paper.

In section 2 we discuss general bookmark management issues. We describe HyperBK in section 3. The web page classification algorithm is described in section 4, and results of the evaluation are presented in section 5. Similar systems are reported in section 6. We give our future work and conclusion in section 7.

2. BOOKMARK MANAGEMENT ISSUES

Bookmark files tend to be partially organised, with some web pages carefully clustered into a single category or a hierarchy of categories [2]. Many other web pages are not assigned to any category in particular, and are either lumped together in the top-most category or into a generically named category (e.g., 'My Favourite Pages'). The complete or partial URL of frequently used bookmarks may be accurately remembered so that as it is being keyed into the browser's address bar, the browser will assist with a simple URL completion task. Infrequently visited, but bookmarked, web pages may be easy to find, if they have been stored in the category the user is looking through, but frequently, the page will not have been placed into its most relevant category, and may prove difficult to find again. If pages are (almost) always saved to the most relevant bookmark category, then they may be easier to find again in the future. As bookmark files tend to be poorly organised, the user could probably do with some assistance. Safari, Mozilla FireFox, and now Microsoft Internet Explorer show the last category saved to, instead of the root category, so the user must either knowingly save a bookmark to the wrong category, or must otherwise locate or create a more appropriate category.

Every once in a while, a user may wish to update a category by looking for more related information. Here, a frequently or infrequently visited category or page may be selected, and the user will try to remember the strategy that was used to find this page in the first place. If the user had foresight,

[*]This paper is based on the work of Ian Bugeja, a BSc IT (Hons) student at the University of Malta, who built the system described in this paper for a Final Year Project under my supervision.

he or she may have also bookmarked the results page of the search engine query that had been used, so that rather than trying to remember the query, the URL containing the query can simply be re-requested. This makes a number of assumptions: that the user remembered to bookmark the query; that the page bookmarked was actually related to the query; and that the user wants to find pages that are relevant to a single page (many search engines allow a straightforward search for pages similar to a given URL). Sometimes, however, a user may wish to find pages that are similar to a category or cluster of related bookmarks.

Other bookmark management related issues are concerned with the freshness or currency of the bookmarked URLs. Web servers do go down or are renamed. Pages may exist for a short period of time. The page contents may change, even though the URL continues to exist. Hard disk space is becoming incredibly cheap. Given the relatively small number of bookmarks on average, and the small average size of bookmarked files, it should not be too expensive to allocate a permanent local cache for web pages that have been bookmarked. If the address of the original of the bookmarked page is later changed, then rather than having only a vague memory of what important thing the page contained, the page can be reloaded from the local cache. In addition, assistance can be given to automatically relocate the page on the Web (if it still exists).

Web browsers and hypertext systems in general have always had problems finding a suitable interface to help users navigate their way around recently visited nodes. Historically, this was identified as one of the issues leading to the 'Lost in Hyperspace' syndrome [5], in particular because it is difficult to build a representation of the user's recent browsing activity that accurately matches his or her mental model of the recent past. Web browsers have tended to adopt a flat, linear representation of what may be a complex user session including cycles, backtracking, and branching. Current Web browsers have increased the complexity of the problem by retaining the flat, linear path representation, even though users can now have multiple tabbed windows and multiple windows, potentially containing concurrent tasks that may or may not be related to each other. The problem of searching for information in history is likely to increase in the future, as systems such as MyLifeBits [7] contemplate digitally storing *everything* seen by a user throughout his or her lifetime.

3. BACKGROUND TO HYPERBK

HyperBK was developed as a Mozilla Firefox extension[1] to provide simple mechanisms to address some of the issues referred to in section 2, but the primary motivation was to automatically classify a web page that a user in the process of bookmarking, according to the bookmark classifications or categories that the user has already created. We describe the automatic classification task more fully in section 4, but here we give a brief overview of the other tasks.

3.1 Views of the History List

Bugeja provides multiple representations of the history list. On visiting a page, the page is classified to find a suitable category in the bookmark file, just in case it will be bookmarked. Apart from the ability to see the recently visited pages in the order of most recently visited, the history list resembles the bookmark file, with visited pages allocated to bookmark categories (see Fig. 1). This enables users to locate visited pages according to their topic. This does require a high degree of correctness in the classification of web pages, otherwise pages will be stored in the wrong category and users will have difficultly using the topic-related view to locate previously visited web pages. Other filters available to find visited pages are by frequency count, to quickly identify frequently visited pages; by keyword, a representation of the contents of the page are kept so keyword search is possible on the content of visited pages, rather than just their title; and, finally, by co-incidence. In this last case, we assume that the user cannot remember any distinguishing characteristics of the visited web page, but remembers another page visited in roughly the same period. This other page can be located using the tools described above, and then a time filter can be used to pull out the other pages visited just before or after the remembered page. One possible weakness of Bugeja's approach, which is inherited from the way Web browsers, and Mozilla in particular, implement their history lists, is that only the last visit to a web page is recorded. If the user is looking for web page, P_1, which has been visited at time T_1 and again at the later time T_2, and the user remembers visiting another page, P_2 close to time T_1, and T_2 - T_1 is larger than the time filter, P_1's only recorded visit will be at time T_2, so looking up P_2 will not help the user to find page P_1. However, if P_2 is a very frequently revisited page, then remembering it is also of little use, because the user will have far too many instances to choose from. The best page to remember is an infrequently visited page (preferably just once), so that remembering it will lead to the page P_1 that the user seeks.

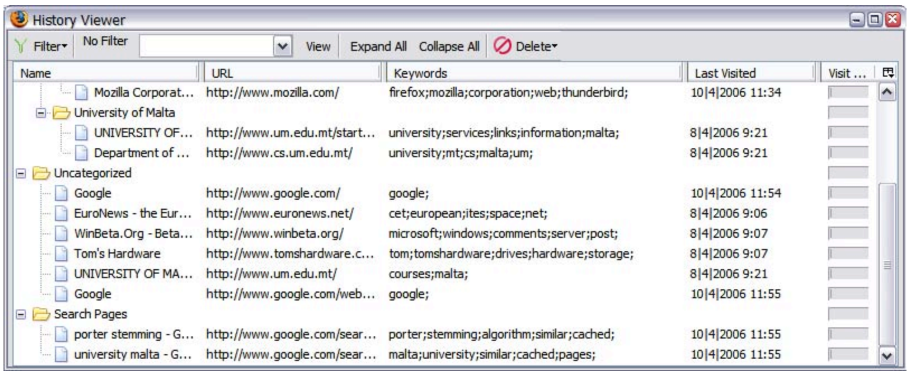
3.2 Web Page Tracker

A Web page is bookmarked because the user intends to revisit the page [1] and so saves the page's address in a bookmark or favorites file. However, Web pages may be moved to a new address by the maintainers of the web page, in which case the stored address is out of date, and the user may be given a server response that the page cannot be found when the user next tries to revisit it. HyperBK stores bookmarked web pages in a local cache, to enable the user to see the contents of the page even if the page is no longer available online, and also to assist with the automatic rediscovery of the page on the internet (if it still exists and is indexed in its new location by the search engine). We do not distinguish yet between a dynamic page and a static page. It may be useful to do so, because if a dynamic page changes frequently, then it may be likely that the page has been bookmarked for reasons other than the content, and attempting to rediscover a dynamic page according to its content is likely to fail.

3.3 Search for Related Pages

Each time a web page is bookmarked two things happen. First, we track back through the pages that the user has visited that act as 'Search Engine (SE) Referrers' (that is, a visited page that contains a link to the next visited page)

[1]HyperBK is available at https://addons.mozilla.org/firefox/2539/


History Viewer

Filter | No Filter | View | Expand All | Collapse All | Delete

Name	URL	Keywords	Last Visited	Visit ...
Mozilla Corporat...	http://www.mozilla.com/	firefox;mozilla;corporation;web;thunderbird;	10|4|2006 11:34	
University of Malta				
UNIVERSITY OF...	http://www.um.edu.mt/start...	university;services;links;information;malta;	8|4|2006 9:21	
Department of ...	http://www.cs.um.edu.mt/	university;mt;cs;malta;um;	8|4|2006 9:21	
Uncategorized				
Google	http://www.google.com/	google;	10|4|2006 11:54	
EuroNews - the Eur...	http://www.euronews.net/	cet;european;ites;space;net;	8|4|2006 9:06	
WinBeta.Org - Beta...	http://www.winbeta.org/	microsoft;windows;comments;server;post;	8|4|2006 9:07	
Tom's Hardware	http://www.tomshardware.c...	tom;tomshardware;drives;hardware;storage;	8|4|2006 9:07	
UNIVERSITY OF MA...	http://www.um.edu.mt/	courses;malta;	8|4|2006 9:21	
Google	http://www.google.com/web...	google;	10|4|2006 11:55	
Search Pages				
porter stemming - G...	http://www.google.com/sear...	porter;stemming;algorithm;similar;cached;	10|4|2006 11:55	
university malta - G...	http://www.google.com/sear...	malta;university;similar;cached;pages;	10|4|2006 11:55	

Figure 1: History viewer (from [3]).

until we have a chain of visited pages with either a search engine results page as the root or else the first page visited in the chain. It is possible to correctly identify the SE Referrer, even if a session is split across multiple tabs and windows, and even if a session is started in one tab or window and continues in another tab or window, as long as a link is actually followed (rather than, for instance, copying a URL and pasting it into an address bar). If a search engine results page is the root of the chain, then the keywords that were used to generate the results page are extracted and stored. The second thing that happens is that keywords from the bookmarked page are extracted and stored. These keywords are used to perform a keyword search on the history list (while the page is still in the history); to rediscover the address of a bookmarked web page if the web page has changed address; to generate a query to search for similar pages on the web; and to contribute to the description of the category to which the bookmark belongs (if any) to i) generate a query to search for web pages that could belong to the category on the web, and ii) to decide if a web page in the process of being bookmarked could belong to this category.

A user can invoke the contextual menu from a category name in the bookmark file and select the "See Also" option to search the Web for web pages similar to the bookmarks in that category (see Fig. 2). HyperBK can use either the query that the user had originally used to find web pages that were eventually bookmarked into the category (the Search Engine Referers described earlier in this section), or else HyperBK can construct a query from the keyword representations of web pages in that category. In a limited evaluation involving 7 users, users claimed that they were "Very Satisfied" or "Satisfied" with 82% of overall results of the queries generated automatically from categories in their bookmark files (the use of previous user created queries has not yet been evaluated, as it requires reasonably long term use of HyperBK).

HyperBK currently submits the automatically generated query to Google's Web Directory, rather than to the Google Search Engine *per se*, because in initial personal trials Bugeja obtained better results from the Web Directory ([3], pg. 71). The algorithm is being improved to take context into account, and to obtain useful results even when the query is submitted to the Google Search Engine [13].

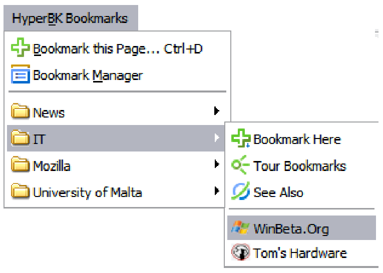


Figure 2: HyperBK Bookmark menu (from [3]).

4. CLASSIFYING WEB PAGES TO BE BOOKMARKED

Each web page that is accessed is parsed using the Document Object Model (DOM) to extract the text components. Stop words, HTML and JavaScript tags are removed, and the remaining words are stemmed using Porter's Stemming Algorithm [11]. According to Bugeja, the five stems with the highest frequency are selected to be representative of the page, but only if they have a frequency of at least two, otherwise only three stems are selected. This helps to keep down computational costs. If a web page author has used META keywords in his or her document, then the five META keywords that most frequently occur in the document are used as the document representation instead.

In a future experiment, the keyword selection process will be modified. First, we will segment a document into its component topics, and extract keywords from the topic most likely to be relevant to the user. Although the results obtained by evaluation are good (see section 5), we think we can improve on them. Picking the top five ranking terms will not necessarily accurately capture the user's interest in the page. A web page is likely to consist of more than one topic, and it is unlikely that a page has been bookmarked because the user is interested in each topic. If only high ranking terms are selected, then potentially, the terms individually represent different topics that occur on the page, so some of the selected terms may, in fact, be irrelevant to the user. We will build upon HyperBK's ideas related to the SE Referrer to find out which keywords were used on a search engine query page which ultimately led to the current page. We can then use these keywords to represent the user's interest in the page. Alternatively, we can examine the parent that was used to access the current page, extract the region surrounding the link and assume that terms occurring in that region accurately describe the potential interest in the to-be-bookmarked page. Finally, we can segment an accessed page into its component topics, merge similar topics, and generate a representation for each topic.

Ultimately, we want to recommend that the user saves the bookmark entry into a particular category. Fig. 3 shows the 'Add Bookmark' dialog box. The algorithm described below is used to select a candidate category in which to store the bookmark. A thumbnail of the page is shown in the top-left hand corner of the dialog box, and the candidate category is highlighted in the 'Bookmark Location'. If the user presses the 'OK' button, the bookmark will be stored in this location. Alternatively, the user can either search for a more appropriate location to store the bookmark; create a new category; or, save the bookmark to the last used location by pressing the 'Bookmark in...' button (in this case, the last location was 'University of Malta').

The algorithm used to select the candidate category is based primarily on simple keyword matches. As was described above, each time a web page is accessed, representative keywords are extracted and stored. If the page is bookmarked into a category, these terms are added to the set of terms that represent the category. The recommended category is the category that has the greatest number of keyword matches for the incoming web page. If no category scores highly enough (matches a high number of page terms), then the recommended category will be the category that contains another page from the same domain, as long as the category and incoming page share at least one keyword in common. Finally, if even this fails, then the title of page is compared to the category descriptions. The category with the highest number of matches is recommended.

The HyperBK approach does not automatically create categories and pick reasonable category names, meaning the user must be involved in creating categories and keeping the bookmark file organised by allocating pages to the correct bookmark category until there are a sufficient number of categories and bookmarked pages to make the recommendations accurate. A future experiment is planned to identify the smallest number of categories and category members to make recommendation feasible.

HyperBK includes a wizard to assist with the importation of bookmark files from other web browsers, and to assist with the automatic segmentation of categories if they become too large. Bugeja recommends that a category is split into subcategories once it reaches a membership of 20 pages, but he doesn't give reasons why 20 is a good number to split on. This setting is user changeable.

5. EVALUATION

We decided that the most appropriate way to evaluate the classification algorithm, apart from a longitudinal study in which HyperBK users would evaluate the system *in situ* over a period of time, would be to collect real user's bookmark files to see if HyperBK could assign bookmarked pages to categories in the same way that the users did. This means that we require bookmark files to be organised (and to contain some categories), and we assume that the user has assigned each bookmarked page to the correct category. Of course, this is a weak assumption, but we had insufficient time to conduct a longitudinal study. Bookmark file submission was conducted anonymously through a specially created portal.

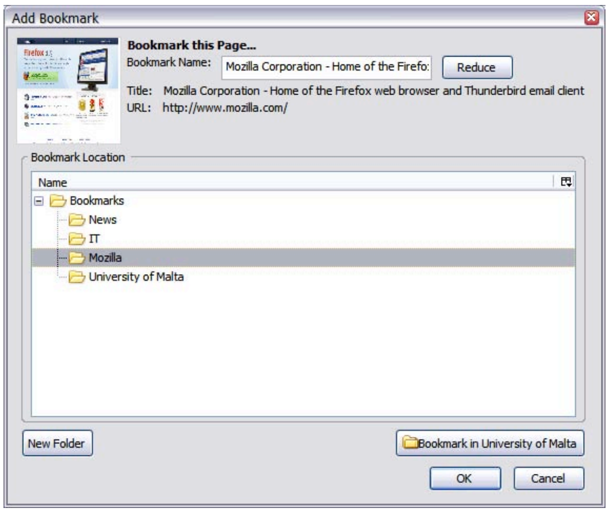
Students following the BSc IT (Hons) degree programme at the University of Malta were invited by e-mail to submit their bookmark files (regardless of which Web Browsing software they used). Of approximately 200 students contacted, 30 submitted their bookmark files (a return of about 15%). Of these, 22 files were considered inappropriate for use because they did not contain more than one or two categories (the files were too loosely structured) and we felt that including them in the evaluation could unfairly bias the results in HyperBK's favour.

We randomly removed 10 URLs from categories of 5 of the remaining 8 bookmark files. We removed less than 10 URLs from the other three: in two cases because there were too few categories overall and in the third case (79231 in Table 5) because although there were many more categories, most of them contained bookmarks that appeared to be mostly unrelated. The challenge was to place the randomly selected bookmarks into the same categories that the users had placed them. The results are given in Table 5.

Figure 5 plots the precision against categories. We would hope for a generally high precision, perhaps dropping slightly as the number of categories grows, especially if categories become less distinguishable from each other (because only 5 terms are selected to describe a page). For two bookmark files containing 38 and 45 categories, precision drops to below 0.7, which is probably unacceptably low. However, one of the two bookmark files contains many similar categories, and the other had many categories each containing unrelated bookmarks.

6. SIMILAR SYSTEMS

Bugeja [3] has given a recent, short survey on bookmark management systems. Bugeja notes that bookmark management systems are usually offered as stand-alone systems - unlike HyperBK, none is integrated into a browser, and as discussed in [3], web browsers offer minimal bookmark management facilities. Most of the systems Bugeja looks at do not offer automatic web page classification features, although Abrams, Baecker, and Chignell [2] list some requirements for bookmark management systems. Among their requirements are improving the organisation of bookmarks on behalf of the user, possibly by automatically "filing" new bookmark entries, and integrating the bookmark management system with a web browser. Feng and Brner [6] use "semantic treemaps" to categorise bookmark entries. Nakajima *et. al.* use keywords that appear in a document's "context" to to automatically construct queries, rather than to classify a document, where a context is composed of the pages visited between a search engine's query page and a page that is bookmarked [10]. Li and Yamanishi [9] use a "finite mixture model" to classify documents, but as Bugeja points out, this requires the prior existence and standard description of categories in which to place documents. On the


Add Bookmark

Bookmark this Page...

Bookmark Name: Mozilla Corporation - Home of the Firefo:

Title: Mozilla Corporation - Home of the Firefox web browser and Thunderbird email client

URL: http://www.mozilla.com/

Bookmark Location

Figure 3: 'Add bookmark' dialog (from [3]).

Table 1: Classification Evaluation Results (from [3]) (Legend: 'BKs' = Total Bookmarks; 'Cat.' = Total Categories; 'Hits' = bookmarks allocated into correct category; 'Misses' = bookmarks allocated wrongly; 'Near Hits' = bookmark allocated to a parent category (excluding the Bookmarks Root); 'Approx Precision' = Near Hits+Hits/total; 'Precision' = Hits/total.)

ID	BKs	Cat.	Root BKs	Root Cat.	Hits	Misses	Near Hits	Approx. Precision	Precision
23740	425	105	49	33	7	2	1	0.80	0.70
24166	330	64	2	26	8	0	2	1.00	0.80
88014	240	53	21	12	7	2	1	0.80	0.70
23248	197	45	6	9	5	3	2	0.70	0.50
79231	158	38	18	18	4	3	0	0.57	0.57
58917	139	29	21	11	8	2	0	0.80	0.80
76243	38	11	3	11	5	0	0	1.00	1.00
80999	22	7	1	5	4	0	0	1.00	1.00
				Totals	48	12	6	6.67	6.07
				Averages	4.8	1.2	0.6	0.67	0.61

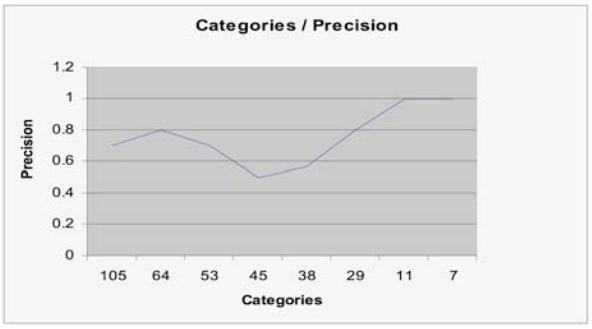


Figure 4: Categories v. Precision (from [3]).

other hand, Shen, *et. al.* [12] use a page summary on which to base a classification. Google also offers bookmark hosting[2], but whether they plan to offer automatic bookmark management remains to be seen.

7. CONCLUSION AND FUTURE WORK

Automatic bookmark file classification could be a useful extension to web browsers. Instead of just offering the last category used to store a bookmark, or dumping the newly created bookmark into a default location, HyperBK presents the user with a candidate category based on a simple Boolean matching algorithm, which has been extended to also consider the domain names of previously bookmarked pages and keyword extraction from titles. This simple algorithm gives reasonable accuracy. In an experiment, 67% of bookmarks were classified correctly. There is room for improvement. We intend to perform topic segmentation on web pages to extract keywords from the topic most likely to be of interest to the user, and we intend to carry out a more extensive evaluation. Instead of classifying a random selection of URLs from a user's bookmark file, we will attempt to allocate all the bookmarked pages to the user selected category in the order that they were really allocated. This will help us determine the average minimum category membership size required to consistently place a page in the correct category.

8. ACKNOWLEDGEMENTS

This paper is based on 'Managing WWW Browser's Bookmarks and History: A Firefox Extension', Ian Bugeja's Final Year project report [3]. Ian was a BSc IT (Hons) student under my supervision in 2005-06.

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

[4] A. Cockburn and B. McKenzie. What do web users do? an empirical analysis of web use. *Int. J. Hum.-Comput. Stud.*, 54(6):903–922, 2001.

[5] J. Conklin. A survey of hypertext. Technical Report 2, Austin, Texas, 3 1987.

[6] Y. Feng and K. Brner. Using semantic treemaps to categorize and visualize bookmark files. In *Proceedings of SPIE - Visualization and Data Analysis*, volume 4665, pages 218–227, January 2002.

[2]http://www.google.com/bookmarks

[7] J. Gemmell, G. Bell, R. Lueder, S. Drucker, and C. Wong. MyLifeBits: fulfilling the Memex vision. In *MULTIMEDIA '02: Proceedings of the tenth ACM international conference on Multimedia*, pages 235–238, New York, NY, USA, 2002. ACM Press.

[8] E. Herder. *Forward, Back, and Home Again - Analysing User Behavior on the Web.* PhD thesis, University of Twente, 2005.

[9] H. Li and K. Yamanishi. Document classification using a finite mixture model. In *Proceedings of the 35th annual meeting on Association for Computational Linguistics*, pages 39–47, Morristown, NJ, USA, 1997. Association for Computational Linguistics.

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

[11] M. F. Porter. An algorithm for suffix stripping. pages 313–316, 1997.

[12] D. Shen, Z. Chen, Q. Yang, H.-J. Zeng, B. Zhang, Y. Lu, and W.-Y. Ma. Web-page classification through summarization. In *SIGIR '04: Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 242–249, New York, NY, USA, 2004. ACM Press.

[13] C. Staff. How did I find that? Automatically constructing queries from bookmarked web pages and categories. In *CSAW'06: Proceedings of the third Computer Science Annual Workshop.*, to appear.