An Adaptive Search Framework for Supporting Cooperative Work in Organizations

Papon Yongpisanpop, Passakorn Phanachitta,
Masao Ohira and Kenichi Matsumoto
Software Engineering Laboratory
Graduate School of Information Science
Nara Institute of Science and Technology
Ikoma, NARA 630-0192, JAPAN
[papon-y, passakorn-p, masao and
matumoto]@is.nasit.jp
+81-743-72-5202

ABSTRACT
Recently people have come to rely on the use of web search engines to learn how to accomplish tasks, solve problems and gain information. However, search results are sometimes too varied and lack relevance to topics related with the organization when using the major search engines such as Google, Yahoo or Bing in an organization. To help with this, in this research we propose a framework called the Adaptive Search Framework. It can learn from the information provided by the users and adapt itself to choose more relevant and important web pages that are related to important topics in the organization. We also propose a re-ranking algorithm for search results. The algorithm gives a score based on the importance and popularity inside the organizations. Our preliminary results show that the Adaptive Search Framework can learn and return results more relevant to topics of interest to the organization at the top ranks. This helps users save time in searching for desired information on the web.

Author Keywords
search engine, web mining, web search interface, collaborative search

ACM Classification Keywords
H.5.m. Information Interfaces and Presentation (e.g. HCI): Miscellaneous

INTRODUCTION

In many organizations, employees use search engines to learn how to accomplish tasks, solve problems and gain information. There are many conventional search engines available, such as Google, Bing, or Yahoo. These major search engines return search results based on relevance scores reflecting the popularity of the results with the majority of people in the world. However, these results are too varied and often irrelevant to the important topics inside an organization. This led us to consider what if there was an additional search engine layer that could learn from what users have searched before and adapt itself to return the top-ranked results that are more relevant to topics in the organization without modifying the conventional search engine itself. In this paper, we propose the Adaptive Search framework, which is a framework designed to be implemented on top of normal search engines. It allows the search engine to adapt itself to the organization. It collects and learns from user-provided information to return results that are more relevant to the topics of interest to the organization. It also includes a collaborative function for users to help each other search, and the framework itself can classify who has expertise in which field inside an organization. Three major problems with using search engines in an organization are (1) most search engines return results that are too varied and not related to the topics of interest in organization because the search engines are based on the majority of people in the world, (2) users need to collaborate when finding the information to solve a problem, but most search engines do not provide a collaborative function for users help each other search, and (3) users need to be able to identify experts in the organization, but most search engines do not provide support for this. To help address these problems, search engines could use information collected when users perform searches to choose and return results related to topics within a user's organization, which should allow users to spend less time searching. To do this, we decided to develop a framework implemented on top of normal search engines.

The rest of this paper describes the work in progress on the Adaptive Search Framework. The next section, Related Works, provides a brief description of search engines and methods of processing search results as a background. It does not provide an in-depth survey of the literature. After that, the section Adaptive Search Framework describes the prototype system that has been implemented. The next section, Experimental Verification, describes a small-scale experiment using the Adaptive Search Framework. The last two sections, Discussion and Conclusion, explore the results of the experiment and describe future work on the Adaptive Search Framework.

APCHI'12, August 28–31, 2012, Matsue-city, Shimane, Japan.

RELATED WORKS

Web Search Engine

A web search engine searches for a specified keyword and returns a list of the web pages relevant to the keyword. Typically, a web search engine operates as follows: 1. Web crawling, 2. Indexing, and 3. Searching. Web crawling and indexing are performed alternately in a cycle. At the beginning of a cycle, a Web crawler retrieves all the web pages contents and stores them as files in a proper format (i.e. Stanford WebBase format). Next, each web page is parsed into a plain text format and sent to an indexer to be analyzed. The web indexer then extracts each term in the page and adds the information to an index database. For example, the indexer extracts terms from the titles, headings, or special fields called meta tags. The purpose of indexing is to allow information to be looked up as quickly as possible. The cycle ends here, with the index database serving as a snapshot of the whole web page set for the users queries. The web crawler then starts the operation again for the next cycle, and the index database will be updated again at the end of the cycle.

Google Custom Search

Google custom search [16] allows users to create a search engine that searches only the contents of a specific website or that focuses on a particular topic. With Google custom search, users can select, prioritize, or ignore specific websites. This allows the user to tailor the search engine to the interests of specific users, taking into account the context and purpose of the search.

For example, when a car salesman searches for lotus on Google search, there are many results about lotus flowers and IBM lotus software. The generic Google search does not limit the context to that of the lotus which is a brand of car. A Google custom search, on the other hand, could search only preselected websites about cars, providing more relevant results to the car salesman. However, the Google custom search engine does not provide any way for users to collaborate to perform searches, nor are the results adapted to the interests of specific users in an organization. The results are still based on popularity measures produced by the majority of users in the world.

Search results clustering

Annotating and clustering search results are key parts of the solution proposed in this paper. Clustering search results classifies web pages from the search results into categories. Some keywords return highly varied results. For example, the keyword "Apache" can return a set of links to the Apache tribe, Apache helicopter, Apache software foundation, and other types of Apache. Grouping these results into categories makes it easier for users to find the web pages they desire.

There are several search result clustering tools, such as Apache Carrot^2 [5] , Vivisimo [20] , and IBM Mapuccino [12]. In this study, we use Apache Carrot^2 since it is an open source library augmented with a set of supporting applications. This allowed us to build a search results clustering engine simply, without any limitation on the number of uses. Such clustering engines can automatically organize a set of search results into topics without external information such as taxonomies or pre-classified contents. Since Apache Carrot^2 is a clustering engine designed for online use, only URLs, titles, and snippet fields are required clustering search results. However, this same simplicity may indicate a lack of in-depth contents, which may not achieve outstanding accuracy in the clustering result.

Digg.com

Digg.com [6] is a website that allows people to discover and share contents from anywhere, with members of the community "voting" for materials. The website provides tools for the members of the community to discover contents, discuss topics, and connect with people with similar interests. Digg.com builds lists of popular contents from across the web. However, as with Google custom search, Digg.com uses the score from majority votes of people in the world. The returned results are based on a majority score.

Digg.com

Digg.com [6] is a website that allows people to discover and share contents from anywhere, with members of the community "voting" for materials. The website provides tools for the members of the community to discover contents, discuss topics, and connect with people with similar interests. Digg.com builds lists of popular contents from across the web. However, as with Google custom search, Digg.com uses the score from majority votes of people in the world. The returned results are based on a majority score

Adaptive Search Engine

Most of the conventional search engines today give back search results based on the popularity of the web pages, however the concept of an Adaptive Search Engine is that of a search engine that will be able to learn from the data collected from each individual user, predict the interests of each user from that information, and return search results related to the topics of interest to that user. In other words, a user is more likely to be interested in search results related to things that the user usually searches for. An adaptive search engine puts today's search in the context of the user history of searches.

Bing: Adaptive Search

Bing [3] is a web search engine developed by Microsoft. In September 2011, Bing announced its newest feature which was called "Adaptive Search". As explained by Adrian Cook [15], the concept of Adaptive Search is that "*Every time you search on Bing, the information provided helps Bing understand what you are trying to do. The more you search, the more Bing can learn and use that information to adapt the experience so that you can spend less time searching and accomplish what you set out to do.*" With Bing, search results for each individual user are personalized based on data collected during previous uses of the search engine. This data is used to determine the individual context of search queries and deliver more personalized results.

But Bing still serve on individual purpose. It learns from the data collected from user, predict the interests of each user from that information, and return search results related to the topics of interest to that user.

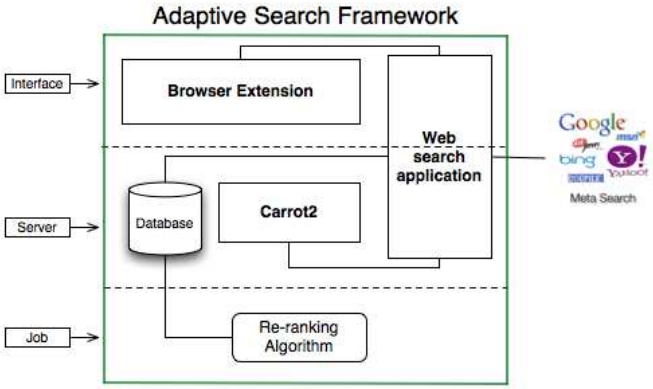


Figure 1. Adaptive Search Framework Design Architecture

ADAPTIVE SEARCH FRAMEWORK

The Adaptive Search Framework developed in this research was implemented on top of the search engines. It collects data (bookmarks, clicks, links, categories) when users search and interact with a web browser. Every time users search on the Adaptive Search Framework, the information provided as they search helps the framework understand what the users are trying to do. The more users search, the more the framework learns. It uses that information to adapt the experience so users spend less time searching and accomplish more easily what users want to do. This section describes the Adaptive Search Framework in terms of its architecture, the data flows involved in using it, and the re-ranking algorithm it uses.

Architecture

Figure 1 presents the general architecture we are using for the development of Adaptive Search Framework. The framework is separated into 3 layers. The top layer is the interface layer, where users perform searches and obtain results just as with the normal search engines. To organize the search results, we developed browser extensions that support the users. The middle layer is the server layer, which returns relevant results related to the keyword and to topics within an organization. This layer communicates with outside search engines by sending query that the users input and obtaining the results. It uses Carrot^2 to cluster the results into groups, which are defined as tags. The web pages and tags are bound with categories, and a search of the Adaptive Search Framework database is performed to determine whether there are any previous results related to the categories. The Adaptive Search Framework database contains information from previous searches within the organization. Then the server layer returns the results to the users through the interface layer, containing both results from the current query and related results from previous searches. After the user interacts with these results, the system stores information about the query, web pages, and tags that the user interacts with. The bottom layer, the job layer, is scheduled on a regular basis to calculate scores for users and web pages using data stored in the database.

Data Flow

In the Adaptive Search Framework, users perform two main activities, searching and bookmarking. In this section, we use these two use cases to explain our data flow implementation. Figure 2 illustrates these two use cases.

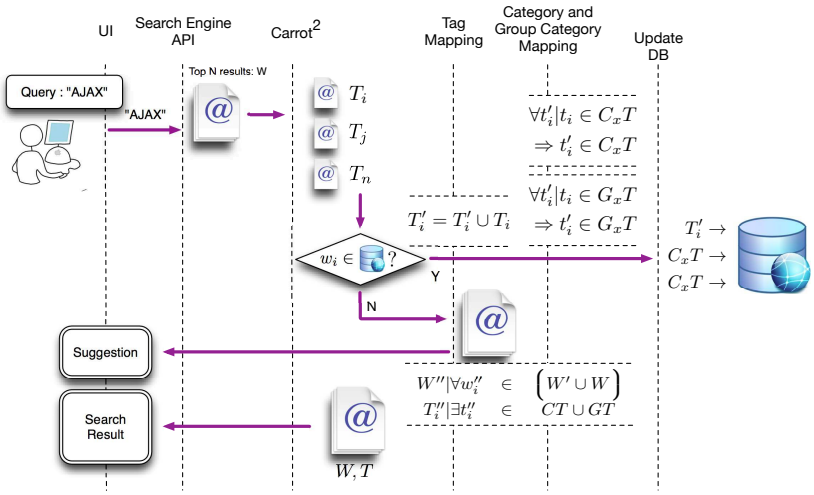
Begin with searching, as shown in figure 2(a) after query q has been passed from a user, it will be sent to conventional search engine API. Top n ranked web pages will be return as a search result set. Let set W denotes the returned top n webs consisted of $w_1, w_2, .., w_n$. Each w_i also contains 3 components, those are"url", "title", and "snippet". For a further process, we send the whole set W to Carrot^2 API where each component of w_i's will be treated as search result clustering source that return clustered label of each w_i. We call these clustered label as tags (t), and we denote a set T_i as a set of w_i's corresponding tags. At this step, we pocess two sets, W and T. Next, we check each member of W if it has already been stored in our framework database. Note that we will explain the store's condition soon later in this section.

In case w_i has already been in the database, we will update T_i to the stored w_i record called w_i'. That is the replacement of w_i's tags will be $T_i' \cup T_i$. For each tag t_i, it will later be assigned to several categories C_x or group categories G_X. We also have to update the linkage between new tags and the stored categories and group categories (C_xT and G_xT). That is $\forall t | t \in T_i$ will be update to each category C_x in C_xT linkage where C_x is its corresponded category to t_i.

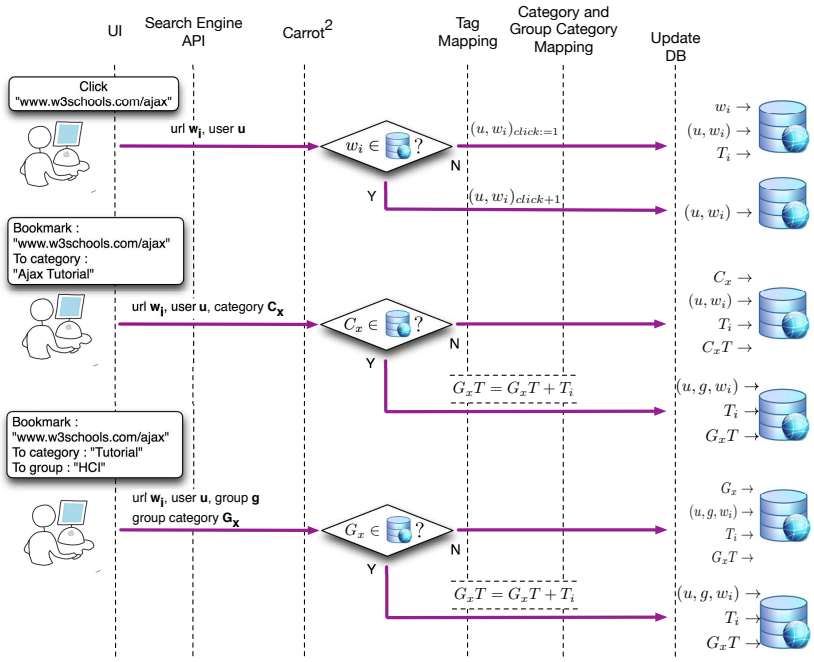
On the other hand, in case that w_i is nonexistence in the database, it will only be processed for our suggestion feature. The web pages suggestion list consists of web pages the database, which has at least one category corresponded to at least one tag of a web page returned from the top n ranking search result set. To achieve that, first of all we merge all T_i into T ($T = \bigcup_1^n T_i$) and list all C_x that have a link to t_i into CT ($CT = \bigcup_1^n C_xT | \exists t_i \in C_xT$) and GT ($GT = \bigcup_1^n G_xT | \exists t_i \in G_xT$). We then do a reverse mapping from CT and GT to obtain T' that consist of all t_i' in $CT \cup GT$, and do a reverse mapping again from T' and get a set W' that satisfied the aforementioned condition. We also need to combine some records from the current search result set W in to the suggested web page set W'. Since we have precessed T', we can map if a t' has a link with a w_i in W and obtain the suggestible web page list from W. At last web pages suggestion list W'' come from W' which are web pages those are existed in the database, merging with W that satisfied the tag condition. So that it will become $W'' | \forall w_i'' \in (W' \cup W) \rightarrow w_i'' \in W''$ pairs with $T_i'' | \exists t_i'' \in (CT \cup GT) \rightarrow t_i'' \in T_i''$ To show the original result set form the conventional search engine, we render only W in that area.

Clicking a Result Link

After a search result has been clicked in the user interface, the url of the clicked link w_i and the user's identity u will be passed to the framework. In case w_i has never been clicked by any user, its identity will not be existed in the database. So that we need to create it with its initialized click counter as 1. We then store user-click with w_i record as a tuple (u, w_i), and



(a) Searching data flow



(b) Clicking through the search result, individual bookmarking and group bookmarking data flow

Figure 2. Proposed Framework's Data Flow

thew_i's corresponded tags T_i in the database. Another case that w_i is already existed, we just then update tuple (u, w_i) by increased the click counter by one.

Bookmarking a Web Page

We have two bookmarking types in our framework, bookmarking for oneself and for group. The different between both of them is the feedback scope of suggesting a new web page to a user. Individual bookmarking only influences to the user who bookmarked it. Group bookmarking; on the other hand, influences altogether group of the user.

In case that a bookmarked web page has been clicked through from the user interface, the identity of that web page w_i must be existed in the database. Then, the required parameters in this bookmark case are w_i, u and the bookmark category C_x. At first we bind u with w_i and store (w_i,u) as a bookmark record and store it in the database as a bookmark identity. If C_x has just been created right before a user bookmarked it, we have to store it as a record in the database at first. We then map all T_i, which is corresponded to the bookmark page w_i to C_x as $C_x T$, and store all of them in the database.

However, if a user choose to bookmark any web page without searching from the framework, we need to process its corresponding tags at first. We choose to pass that web page's basic components such as title and url through the search in-

$$
\begin{aligned}
a(p) &= \omega_1 \sum_{q \to p} h(q) + (1-\omega_1)\left(\omega_2 \sum_{r \to p} u(r) + (1-\omega_2)\left((\omega_3 \sum_{s \to p} u(s) + (1-\omega_3)\sum_{t \to p} u(t))\right)\right) \\
h(p) &= \omega_1 \sum_{p \to q} a(q) + (1-\omega_1)\left(\omega_2 \sum_{r \to p} u(r) + (1-\omega_2)\left((\omega_3 \sum_{s \to p} u(s) + (1-\omega_3)\sum_{t \to p} u(t))\right)\right) \\
u(r) &= \omega_2(\sum_{r \to i} a(i) + \sum_{r \to j} h(j)) + (1-\omega_2)\left(\omega_3(\sum_{r \to k} a(k) + \sum_{r \to l} h(l)) + (1-\omega_3)\left(\omega_4(\sum_{r \to m} a(m) + \sum_{r \to n} h(n))\right.\right. \\
&\quad \left.\left. + (1-\omega_4)(\sum_{r \to o} a(o) + \sum_{r \to p} h(p))\right)\right) \qquad (1)
\end{aligned}
$$

terface as query that allows tags to be processed as well as the ordinary routine.

Bookmarking a Web Page into a group

In group bookmarking, tags and categories will be similarly processed to individual bookmarking. A bookmark page w_i and its corresponding tags T_i are bound with a group category G_x as G_xT and all of them will be stored in the database. The different is the bookmarked record are bound from user, group, and web together as a tuple (u,g,w_i) instead of (u,w_i) in an individual bookmarking. We also do the same if a bookmarked page did not come from the search result by passing a query of the web page's basic components to the framework.

Re-ranking Algorithm

The Adaptive Searching Framework uses an iterative re-ranking algorithm derived from Kleinberg's HITS algorithm [10]. As shown in equation 1, in this algorithm, the authority weighting of a web page *p* calculated by combining the sum of the hub values of all pages *q* pointing to *q* and the sum of the weights of all users *r* visited (weight by $omega_2$), individual bookmarks (weight by $omega_3$), and group bookmarks p. This combination forms the final authority weight of *p*. The hub weight is similarly calculated. The weight of a user *r* is calculated by summing up the authority and hub weights of all pages he has visited (weight by $omega_2$), or bookmarked by himself (weight by $omega_3$) or group (weight by $omega_4$). Another term is indirectly influenced by the user *r*'s weight which comes from his group participation (weight by 1-$omega_4$) The score in this term comes from web pages that all users in a group have bookmarked as a group.

EXPERIMENTAL VERIFICATION

We conducted a small-scale experiment to test our hypothesis that the Adaptive Search Framework can help users in an organization be more productive while using a search engine. The experiment focused on answering the following two questions:

- Are major search engines suitable for using in organizations?
- Can the Adaptive Search Framework suggest results better related to a user's interests and topics?

Experimental Setup

We deployed our Adaptive Search Framework on a Ubuntu server inside the Software Engineering Laboratory at the Nara Institute of Science and Technology. The Framework was implemented on top of 2 major search engines, Google and Bing. 10 members of the laboratory, from three research groups, participated in the experiment. Four people are members of the human computer interaction (HCI) group, four are from the open source software (OSS) group, and two are members of the software reviews (SR).

Results

Data from the database provides results indicating answers to the questions. To answer the first question, concerning the suitability of major search engines for use in organizations, we ranked and compared results returned by the Adaptive Search Framework with results returned by the two major search engines, Google and Bing. The results returned by our framework were ranked using the iterative algorithm described in the section Re-Ranking Algorithm.

ID	URL	ASF	Google	Bing
4	http://www.w3schools.com /ajax/default.asp	1	1	1
3	http://www.templatelite.com /ajax-tutorials/	2	100+	100+
8	http://www.tutorialspalace.com /2012/01/35-useful-ajax-tutorials-for-web-developers/	3	80	100+
116	http://www.maxkiesler.com /2006/03/15/round-up-of-30-ajax-tutorials/	4	74	100+
117	http://www.codeproject.com /KB/ajax/AjaxTutorial.aspx	5	71	100+

Table 1. Comparison return results ranking by using "Ajax tutorial" for the keyword between ASF:Adaptive Search Framework, Google and Bing

As shown in table 1, using the keyword "Ajax tutorial" to search the Adaptive Search Framework, Google, and Bing results in very different rankings. Table 1 shows the top five results returned by the Adaptive Search Framework, and their

rankings in the Google and Bing searches. This result indicates that the most useful links for users in the Software Engineering Laboratory were more highly ranked in the results returned by the Adaptive Search Framework. As shown in figure 4, these five links are particularly important for members of the HCI group. This result reflects the fact that major search engines do not adapt their results to topics of specific relevance inside an organization, while the Adaptive Search Framework is designed to perform that adaptation.

To help address the second question, we examined the ability of the Adaptive Search Framework to suggest websites that have not been viewed by users, but are related to keywords and topics that the users are interested in. As an example, we arranged a scenario to test whether the Adaptive Search Framework would relate Superman and Clark Kent. As shown in figure 3 in the search on the back, at first when a user searches for "Superman movie" there was nothing in the suggestion box because this keyword was new to the framework. However, after a user clicked or bookmark some of the websites about the Superman movie, as shown in the middle search picture, another search for "Superman" resulted in several websites in the suggestion box related to the Superman movie because the framework knows that this user is interested in the Superman movie. So when this user searches for Superman, the Adaptive Search Framework provides information about movies based on the data from the previous search. Finally, the front picture in figure 3 shows that when a user searches for "Clark Kent," Superman's secret identity, the Adaptive Search Framework can suggest that the user should also look for Superman. However, a similar search for "Clark Kent" on the major search engines provides top ranked results only related to Clark Kent. This indicates that the Adaptive Search Framework can suggest results better related to a user's interests and topics.

DISCUSSION

In this section, we discuss the results and give additional data to support the discussion.

How can the framework return better suited to an organization?

There are 2 reasons why we choose "Ajax Tutorial" as a keyword:

1. "Ajax tutorial" is a simple keyword. When a user searches for it using a major search engine, the user gets results based on the popularity of the majority of the people in the world, producing a ranking of results which is quite similar to other search engines.

2. The "Ajax tutorial" keyword is related to the Human Computer Interaction (HCI) group in the Software Engineering Laboratory. This makes it easy for members of the group to determine which websites provide a good tutorial.

To provide additional insight into the results related to the first question, we extracted data from the database to create a relation graph showing the users and web pages from the Software Engineering Laboratory. In this relation graph, users are also separated into their own special groups. Figure 4, inside the square blue area, also shows website ids 8, 4, 3, 117, and 116, the top 5 results shown in table 1 for the "Ajax tutorial" query. The Adaptive Search Framework selected these five websites for the suggestion box because they have a high ranking as shown in table 2 and are related to be "Ajax tutorial" keyword.

How does framework relate apparently unrelated URLs?

If a user searches for Clark Kent, most of the major search engines will not give results about Superman. Unlike normal search engines that depend on references, links, and keywords in the content, the Adaptive Search Framework uses tags and categories mapped to websites to find related websites. So when a user searches for "Clark Kent," the framework looks for tags and categories that match, finding commonalities with Superman. When the framework finds tags and categories, it selects websites from the database related to those tags and categories and returns them as suggested websites.

ID	Score	URL
4	0.0756	http://www.w3schools.com/ajax/default.asp
3	0.0564	http://www.templatelite.com/ajax-tutorials/
8	0.0525	http://www.tutorialspalace.com/2012/01/35-useful-ajax-tutorials-for-web-developers/
116	0.0436	http://www.maxkiesler.com/2006/03/15/round-up-of-30-ajax-tutorials/
2	0.0344	http://en.wikipedia.org/wiki/ajax/
81	0.0342	http://apchi2012.org/
60	0.0329	https://issues.apache.org/bugzilla/
61	0.0329	https://issues.apache.org/jira/secure/Dashboard.jspa
117	0.0327	http://www.codeproject.com/KB/ajax/AjaxTutorial.aspx
82	0.0299	http://hcii2011.org/
85	0.0233	http://www.tripwiremagazine.com/2010/07/30-very-useful-html5-tutorials-techniques-and-examples-for-web-developers.html
...	...	...

Table 2. Score and rank of all webpages inside Software Engineering using iterative re-ranking algorithm.

CONCLUSION AND FUTURE WORK

In this paper, we described our study of ways to help users and organizations to obtain search results that are more relevant to topics of interest within the organization. We proposed and developed an Adaptive Search Framework that uses data provided by users performing ordinary searches, clicking on links, and bookmarking within an organization to enrich and select responses to searches. We performed a small-scale experiment with the Adaptive Search Framework which suggests the framework could return more relevant results and additional related results to searches. Using the Adaptive Search Framework inside an organization could benefit both the users and the organization. The users can save time when

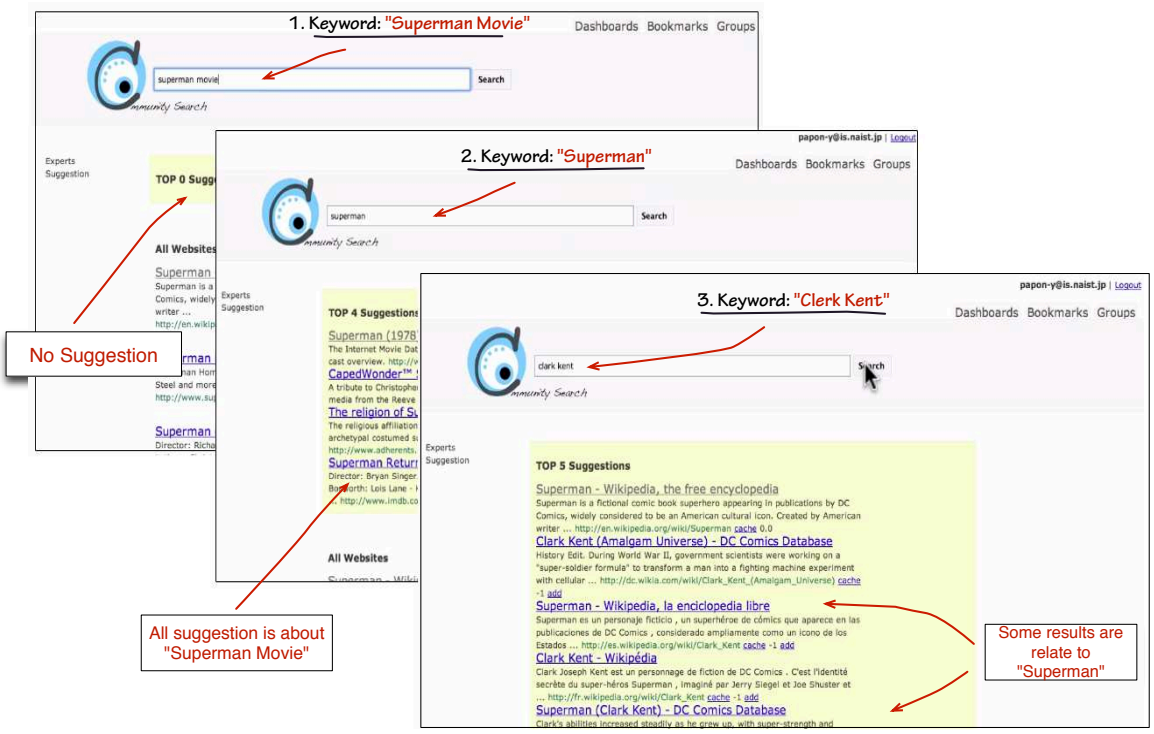


Figure 3. 1. At first, the framework cannot suggest anything when a user searches for Superman Movie because it is a new topic for the framework. 2.After a user has interacted with some of the websites that involve Superman, the framework can suggest some websites when a user searches for Superman based on the previous search data. 3. Also, the framework can suggest Superman when a user searches for Clark Kent

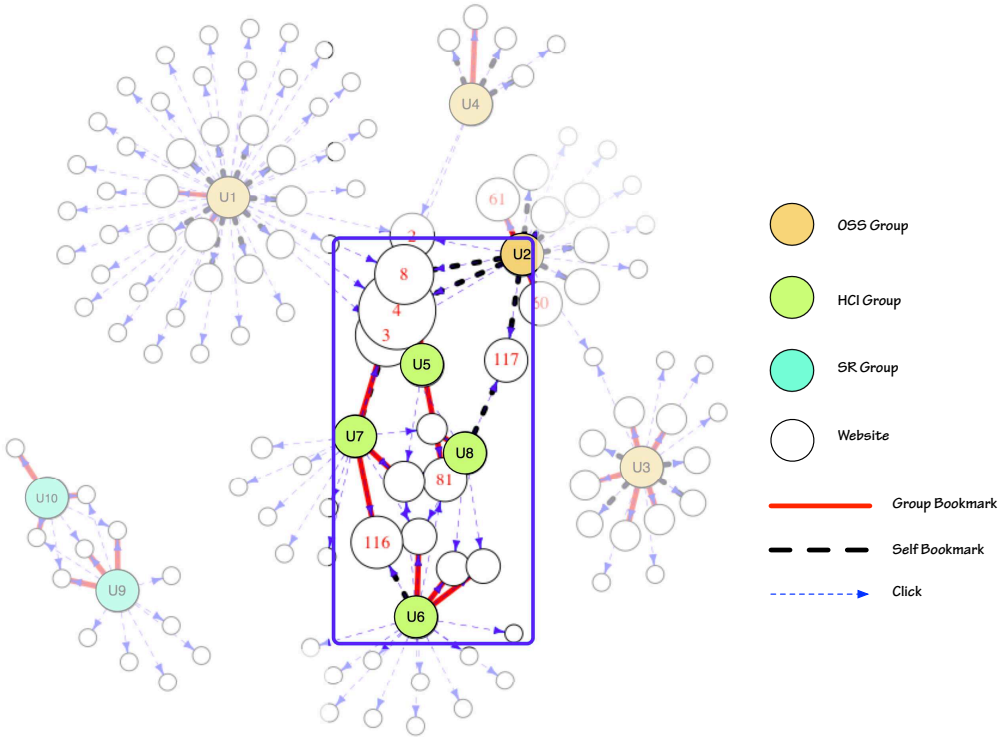


Figure 4. Relation graph between users and websites inside Software Engineering Laboratory.

searching and obtain search results that are relevant to the topics of interest in the organization. The organization also can obtain information about topics of interest within the organization, and maintain an organizational history and knowledge about web information. We believe that over time, use of the Adaptive Search Framework will let users obtain better search results and help organizations gain better productivity.

Future work will concentrate on the following

1. Improve the Adaptive Search Framework to reduce unrelated results (noise). At this point, the Adaptive Search Framework depends on the relationships of web pages, users, and tags to find results related to topics within the organization. However, using Carrot^2 sometimes results in extraneous results, noise, especially during the tagging of web pages. We would like to improve the Adaptive Search Framework to better analyze web pages using different tags to reduce such extraneous results.
2. The prototype Adaptive Search Framework does not yet identify experts within an organization. As a next step, the Adaptive Search Framework should be enhanced to suggest experts inside the organization related to keywords that a user is searching for. To do this, it may be necessary to predefine a set of experts related to the organization, or to develop a method to let the Adaptive Search Framework identify users and their expertise from their use of the system.
3. The Adaptive Search Framework should be deployed in several different medium size organizations, to determine whether the framework can satisfy people in real organizations and provide better search results than directly using major search engines. To do this, the framework should be used for at least 3 to 6 months, with data collection and analysis of the resulting relationships between users, organizational topics, and web pages.

REFERENCES

1. Agichtein, E., Brill, E., and Dumais, S. Improving web search ranking by incorporating user behavior information. In *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '06, ACM (2006), 19–26.
2. Bao, S., Xue, G., Wu, X., Yu, Y., Fei, B., and Su, Z. Optimizing web search using social annotations. In *Proc of WWW'07* (New York, USA, 2007), 501–510.
3. Bing. http://www.bing.com.
4. Carpineto, C., Osiński, S., Romano, G., and Weiss, D. A survey of web clustering engines. *ACM Comput. Surv. 41* (July 2009), 17:1–17:38.
5. Carrot^2, A. http://project.carrot2.org/index.html.
6. Digg. http://www.digg.com.
7. Dumais, S., Cutrell, E., and Chen, H. Optimizing search by showing results in context. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, CHI '01 (2001), 277–284.
8. Hotho, A., JŁschke, R., Schmitz, C., and Stumme, G. Information retrieval in folksonomies: Search and ranking. In *The Semantic Web: Research and Applications*, vol. 4011 of *Lecture Notes in Computer Science*. Springer Berlin / Heidelberg, 2006, 411–426.
9. Kashoob, S., Caverlee, J., and Kamath, K. Community-based ranking of the social web. In *Proceedings of the 21st ACM conference on Hypertext and hypermedia*, ACM (2010), 141–150.
10. Kleinberg, J. M. Authoritative sources in a hyperlinked environment. *J. ACM 46* (September 1999), 604–632.
11. Li, Y., Chen, X.-Z., and Yang, B.-R. Research on web mining-based intelligent search engine. In *Machine Learning and Cybernetics, 2002. Proceedings. 2002 International Conference on*, vol. 1 (2002), 386–390.
12. Mapuccino. http://www.research.ibm.com/topics/popups/innovate/java/html/mapuccino.html.
13. Morris, D., Ringel Morris, M., and Venolia, G. Searchbar: a search-centric web history for task resumption and information re-finding. In *Proc of CHI '08* (2008), 1207–1216.
14. Morris, M. R., and Horvitz, E. Searchtogether: an interface for collaborative web search. In *Proc of UIST '07* (2007), 3–12.
15. Search, A. http://www.bing.com/community/site_blogs/b/search/archive/2011/09/14/adapting-search-to-you.asp.
16. Search, G. C. http://www.google.com/cse/.
17. Silverstein, C., Marais, H., Henzinger, M., and Moricz, M. Analysis of a very large web search engine query log. *SIGIR Forum 33* (September 1999), 6–12.
18. Sugiyama, K., Hatano, K., and Yoshikawa, M. Adaptive web search based on user profile constructed without any effort from users. In *Proc of WWW '04* (2004), 675–684.
19. Teevan, J., Alvarado, C., Ackerman, M. S., and Karger, D. R. The perfect search engine is not enough: a study of orienteering behavior in directed search. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, CHI '04, ACM (2004), 415–422.
20. Vivisimo. http://vivisimo.com.
21. Wang, J., Chen, Z., Tao, L., Ma, W.-Y., and Wenyin, L. Ranking user's relevance to a topic through link analysis on web logs. In *Proc of WIDM '02* (2002), 49–54.
22. Wang, X., and Zhai, C. Learn from web search logs to organize search results. In *Proc of the SIGIR '07* (2007), 87–94.
23. Yongpisanpop, P., Ohira, M., and Matsumoto, K.-i. Community search: a collaborative searching web application with a user ranking system. In *Proc of OCSC'11* (2011), 378–386.