Adapting search engine for organization using Adaptive Search Framework

Papon Yongpisanpop,[†1] Passakorn Phannachitta,[†1]
Masao Ohira[†1] and Kenichi Matsumoto[†1]

Recently people has relied on the utilize of web search engines to learn how to accomplish tasks, solve problems and gain information. But in the organization, using just the major search engines such as Google, Yahoo or Bing sometimes the search results are too various and not much relevance to the organization's related topic. In this paper, we propose a framework called Adaptive Search Framework. It can learn from the users' provided information and adapt itself to choose the more relevant and important web pages that are the in-organization's related topics. We also propose a search results re-ranking algorithm. The algorithm gives score based on the importance and popularity inside the organizations. Our preliminary results show that Adaptive Search Framework can learn and return more topic-relevant results to the organization at the top ranks. It helps users save much time in searching for an intended web page.

1. Introduction

In many organizations, employees are using search engines to learn how to accomplish the tasks, solve problems and gain information. There are a lot of conventional search engines for us to use, such as Google, Bing, Yahoo, etc. The existed problem is these major search engines return search results based on the relevance scores reflected on the popularity of the majority people in the world. But those results are too various and seems to be irrelevant to the topics inside the organization. So we think of what if there is an additional search engine's layer which can learn from what users have searched and adapt itself to return the top-rank results that are more relevant to the organization topic without modifying the conventional search engine itself.

In this work we propose Adaptive Search framework. It is a framework designed to implement on the top of normal search engines. It allows the search engine to adapt itself to the organization. It learn from the user-provided information to return more relevant results to the organization's interested topics.

†1 Nara Institute of Science and Technology

2. Background

2.1 Web Search Engines

Web search engine searches web pages from a specified keyword and returns a list of the keyword-relevant webpages. Typically, a web search engine operates in the following order: 1. Web crawling 2. Indexing and 3. Searching. Crawling and Indexing are operated 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. Web indexer then extracts each term and create the index database. (for example, terms are extracted 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 and the index database will serve the users' queries as a snapshot of the whole web page set. The web crawler then start the operation again for the next cycle, and the index database will be updated next time at the end of the cycle.

2.2 Search results clustering

Search results annotation and clustering are the key solution in this paper. The proposed of the utilize of search result clustering is to group the web pages search result into the same cate-

gory. Since some keyword may return a high degree in variation, for example the keyword "Apache" can return a set of link to Apache tribe, Apache helicopter, Apache software foundation, etc. Grouping them into a category would make users easier in finding their needed web pages.

There are several number of search result clustering tools i.e. Apache Carrot^2, Vivisimo, and IBM Mapuccino. In this research we applied the use of Apache Carrot^2. It is an open-source library augmented with a set of supporting applications. They make us able to build a search results clustering engine simply without a limitation in any term of uses. Such an engine will organize your search results into topics, fully automatically and without external knowledge such as taxonomies or pre-classified content. Since Carrot^2 is an online-mode cluster, using it for clustering needed only url, title, and snippet fields. It may be lack of content for achieving a high accuracy clustering result.

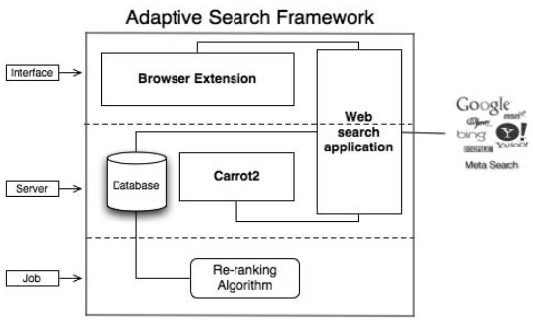


Fig. 1 Model for architectural design

3. Adaptive Search Framework

Adaptive Search framework is implemented on the top of search engines. It collects data (bookmarks, clicked, links, categories) when users search and interact with web browser. Every time users search on Adaptive Search Framework, the information provided helps the framework understand what users are trying to do. The more users search, the more framework can learn. It use that information to adapt the experience so users spend less time searching and accomplish what users want to do.

3.1 Architecture

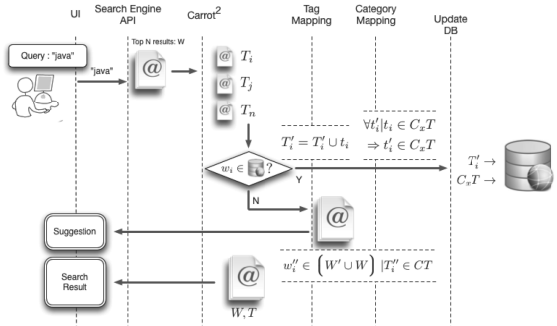
In figure 1 shows the design architecture of Adaptive Search Framework. The framework is separated into 3 layers. The top layer is an interface. Users use it to search and obtain the results just as the normal search engines. we decide to develop browser extensions to support users to organize search results. The purpose of server layer is to return relevant results that related to the keyword and also related to the topics in organization . This layer communicates with outside search engines by send keyword that users input and obtain the results. Once we have the results, we use carrot^2 to cluster the results to categories. Then we use the categories to search in the database to find if there is any webpages that related to the categories. After, we return the results back to users, we will store webpages and tags which users interact with. The job layer is scheduled to calculate score for users and web page by using data from the database.

3.2 Data Flow

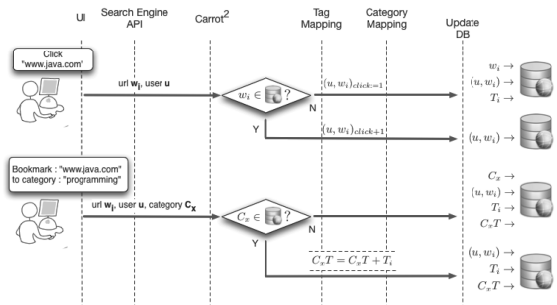
To explain our framework data flow, we break down the user interaction into three cases: searching, clicking a result link, and bookmarking a page. Each case is illustrated in figure 3.2.

3.2.1 Searching

For the first searching case in figure 2(a). After a user pass a



(a) Searching data flow



(b) Clicking the search result and bookmarking data flow

Fig. 2 Proposed Framework's Data Flow

query q from the search interface, the query will be sent through the conventional search engine API, and the top n result will return. Let W denotes a returned top n search result web composed with $w_1, w_2, ..., w_k$. We send the set W to Carrot^2 API and then each of w_i's clustering labels will be returned. We called these as tags. We denote T_i as a set of tags corresponded to w_i. At this step, we check each of w_i if it has been stored in our database. The storing condition will be explained later. Case a w_i has been stored in our database, all the tags $t|t \in T_i$ will be update to the stored w_i' as T_i'. That is $T_i' = T_i' \cup T_i$. Then all $t|t \in T_i$ will be updated to each category C_x in C_xT where C_x is the corresponded category to t_i'. At last, we store T_i' in our database.

Case that the w_i has not been stored in the database, it will be processed for the suggestion feature. First we union all the T_i into T ($T = \bigcup_1^n T_i$) and list all C_x which are corresponded to t_i into CT ($CT = \bigcup_1^n C_xT | \exists t_i \in C_xT$). If we do reverse mapping in this step, we will obtained T' whose all the t_i' are in CT. We then reverse map again form T' and get a set of web page W' who contains at least one t_i' corresponded to CT.

For instant, we have already collected all the suggested web page in the database. We need to combined the some records from current search result set (W) in to the suggested web page set (W'). Since we have precessed T', we can map each t' into W and obtain the suggestible web page list from W.

At last we composed a recommended web page set W'' from $w_i'' \left(\in W' \cup W) | T_i'' \in CT \right)$, and we render W as the original search result from the conventional search engine.

3.3 Clicking a Result Link

After a user click a search result link, we will pass the url of the clicked link w_i and the user's identity agrument u to the framework. If w_i has never been clicked from any user, it won't be in the database. In this case we will initial the number of clicks as 1 and then store the user-click w_i in the database as a tuple (u, w_i). We also store the corresponded tags T_i to the database. Case that w_i is already existed. we just increase the total click of tuple (u, w_i) by one, and then store it back to the database.

3.3.1 Bookmarking a Web Page

For the latest usage case; bookmarking a web page, we assumed that all the bookmarked web page must be clicked through from our search framework user interface. Then we need not to check the existent of w_i. Parameters in this bookmark case are the w_i, u, and the bookmark category C_x. At first we bind u with w_i and store (u,w_i) as a bookmark record. If C_x has just been created right before user bookmarked, we have to store it 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_xT, and store all of them in the database.

3.4 Re-ranking Algorithm

The re-ranking algorithm is derived from Kleinberg's HITS algorithm[4]. In this algorithm, to calculate the authority of weight of a web page p, the sum of hub values of all pages q pointing to p and the sum of weights of all users r visited p are combined to form the final authority weight of p. The hub weight is similarly calculated. The weight of user r is calculated by summing up authority and hub weight of all pages he has visited or bookmarked. Then the weights of web pages and users reinforce each other in an iterative way according to our second assumption. In the algorithm, β is the parameter to adjust if the system needs to weight more score on when the users clicks on the page. and α is the parameter to adjust if bookmark is more importance to give more weight score to the user.

$$
\begin{aligned}
a(p) &= \beta \textstyle\sum_{q\to p} h(q) + (1-\beta) \sum_{r\to p} u(r) \\
h(p) &= \beta \textstyle\sum_{p\to q} a(q) + (1-\beta) \sum_{r\to p} u(r) \\
u(r) &= \alpha \left(\textstyle\sum_{r\to p} a(p) + \sum_{r\to q} h(q) \right) \\
&\quad +(1-\beta) \left(\textstyle\sum_{r\to y} a(y) + \sum_{r\to z} h(z) \right)
\end{aligned}
$$

4. Experiments

4.1 Experimental Setup

We conducted experiments in order to answer two questions to solve the problem of variety and irrelevant results that returned from search engines: (1) Does the framework will be able to suggest results more related to the topic inside organization and (2) Does Re-ranking algorithm re-rank webpages based on the importance inside organization. The experiment were implemented on top of 2 major search engines which are Google and Bing. We deployed it in the server inside our Software Engineering laboratory in Nara Institute of Science and Technology.We let our members inside the laboratory use it. We used the data from the database to find the results and answers the questions.

4.2 Experimental Results

4.2.1 Suggestion results related to the topic inside organization

As it described in section 3.2, Adaptive Search Framework will be able to suggest the webpages that related to keyword and the topics inside organization. In this experiment, we use "java" as a query. The results in table 2 shows the comparison of the webpages that returned between our search engine that implemented our framework and normal search engines. We can assume that the webpages that return from our framework are related to java and also related to Software Engineering laboratory more than the webpages that returned from normal search engines. The reason is because those webpages has been analyzed by the majority of members inside the laboratory. From the results, it can be proved that the framework will be able to return more relevant results to the keyword and topics based on the popularity in the organization.

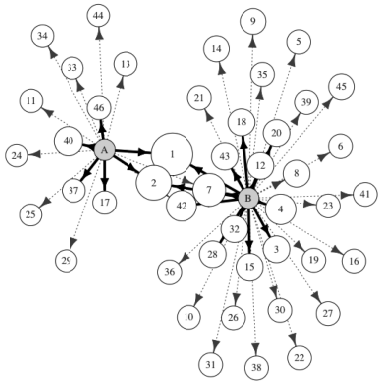


Fig. 3 The interaction between users and webpages.

1	0.071	www.javaranch.com
7	0.068	www.vogella.de/articles/Eclipse/...
2	0.479	en.wikipedia.org /wiki/JavaBean
42	0.037	junit.sourceforge.net /doc/faq/faq.htm
4	0.037	www.ehow.com/how-5347487 -resolve-language-stack-overflow-error.html

Table 1 Score and rank of top 5 webpages that related to "java" by using our re-ranking algorithm.

4.2.2 Re-rank webpages based on the importance inside organization

We use data described in figure 3 to re-calculate score for the webpages inside Software Engineering Lab based on the popularity of webpages. There are 2 users that search for "java" which are user A and B. The blue dashed line is represented when the users click on the webpages. The black bold line is represented when the users bookmark the webpages. The results from table 1 shows the top 5 webpages which have the highest score among the "java" category. As you can see that www.javaranch.com (ID=1) is the first rank because it has been bookmarked by 2 users and it also has been clicked several times.

Adaptive Search	**Google**
www.javaranch.com	www.java.com/getjava
www.vogella.de/articles/Eclipse/ar..	www.java.com
en.wikipedia.org/JavaBean	en.wikipedia.org/wiki/Java-(programming-language)
junit.sourceforge.net/doc/faq/....	www.java.net
www.ehow.com/how-5347487- resolve-language-stack-over....	www.oracle.com/technetwork/java...

Table 2 Comparison top 5 results between Adaptive Search and Google

In table 2 shows the comparison between results that come from search engine that implemented Adaptive Search Framework which is using inside our laboratory and normal search engines(Google). There are some webpages that we could not see them in the very first top rank when you are trying to search it in the normal search engine. The reason is because they are more related to the topics in organization and also has been analyzed by users.

5. Discussion

In the research paper called "Learn from Web Search Logs to Organize Search results"[10] has shown the comparison between the cluster-based method and log-based method. The cluster-based method has to rely on the keywords extracted from the snippets to construct the tag for each cluster. The log-based method use the center of each star cluster[10] as the label for the corresponding cluster.

From the table 3, the log-based method gives more readable and more specific labels because it generates labels based on users' queries. But in general, when we apply our framework into the

Log-based method	Cluster-based method
jaguar animal	jaguar, auto, accessories
jaguar auto accessories	jaguar, type, prices
jaguar cats	jaguar, panthera, cats
jaguar repair	jaguar, services, boston

Table 3 Label comparison between Log-based method and Cluster-based method

organization, we cannot gather all the members search log data to create a structure information for the framework to learn from. The framework will have to learn a bit by bit from the users and when we gain enough web search logs from the members, we will also cluster our links in the organization by using that web search log together with the clustering engine which we believe it will be more accurate to give us the label of webpages.

6. Concluding Remarks

In this paper, we studied the way to improve search engine to return search results those are more relevant and important to the topics in organization. We created a framework that can be learn from the data provided by users in the organizations. Integrated Adaptive Search Framework into search engines and use it inside the organization could benefit both users and organization. Users will save a lot of time searching and will be able to get search results those are relevance to the organization. Also organization will be able to sustain the web information legacy. In the future, we would develop a collaborative search function that allow users to collaborate with others while helping searching in the same topic. and also we would optimize the re-ranking algorithm and explore more sophisticated features to improve our framework.

Acknowledgments This research is conducted as part of the Next Generation IT Program, Grant-in-aid for Young Scientists (B), 22700033, 2011, and Grant-in-Aid for Scientific Research (B), 23300009, 2011 by the Ministry of Education, Culture, Sports, Science and Technology, Japan.

References

1) Eugene Agichtein, Eric Brill, and Susan Dumais. Improving web search ranking by incorporating user behavior information. In *Proc of SIGIR '06*, pages 19–26, 2006.
2) Shenghua Bao, Guirong Xue, Xiaoyuan Wu, Yong Yu, Ben Fei, and Zhong Su. Optimizing web search using social annotations. In *Proc of WWW'07*, pages 501–510, New York, USA, 2007.
3) Claudio Carpineto, Stanislaw Osiński, Giovanni Romano, and Dawid Weiss. A survey of web clustering engines. *ACM Comput. Surv.*, 41:17:1–17:38, July 2009.
4) JonM. Kleinberg. Authoritative sources in a hyperlinked environment. *J. ACM*, 46:604–632, September 1999.
5) Dan Morris, Meredith RingelMorris, and Gina Venolia. Searchbar: a search-centric web history for task resumption and information re-finding. In *Proc of CHI '08*, pages 1207–1216, 2008.
6) MeredithRingel Morris and Eric Horvitz. Searchtogether: an interface for collaborative web search. In *Proc of UIST '07*, pages 3–12, 2007.
7) Craig Silverstein, Hannes Marais, Monika Henzinger, and Michael Moricz. Analysis of a very large web search engine query log. *SIGIR Forum*, 33:6–12, September 1999.
8) Kazunari Sugiyama, Kenji Hatano, and Masatoshi Yoshikawa. Adaptive web search based on user profile constructed without any effort from users. In *Proc of WWW '04*, pages 675–684, 2004.
9) Jidong Wang, Zheng Chen, LiTao, Wei-Ying Ma, and Liu Wenyin. Ranking user's relevance to a topic through link analysis on web logs. In *Proc of WIDM '02*, pages 49–54, 2002.
10) Xuanhui Wang and ChengXiang Zhai. Learn from web search logs to organize search results. In *Proc of the SIGIR '07*, pages 87–94, 2007.
11) Papon Yongpisanpop, Masao Ohira, and Ken-ichi Matsumoto. Community search: a collaborative searching web application with a user ranking system. In *Proc of OCSC'11*, pages 378–386, 2011.