Adaptive Search Framework: Better Search Result for Community

Papon YONGPISANPON[1*]・Masao OHIRA[3**]・Akinori IHARA[2*]・Kenichi MATSUMOTO[4*]

[*]Graduate School of Information Science, Nara Institute of Science and Technology
[**]Department of Computer and Communication Sciences, Wakayama University

[1]Graduate Student・[2]Asistance Professor・[3]Associate Professor・[4]Professor

［Abstract］
近年、各方面において、組織内の知的生産活動をITC技術を用いて支援する取り組みが本格化している。多くの組織において、知的労働者は、情報収集や問題解決を行う際に、GoogleやYahoo、Bingなどの検索エンジンを利用しているが、それら一般的な検索エンジンを利用することが必ずしも最適ではないことが多い。一般的な検索エンジンは、世界中のユーザの様々なコンテキストや目的を内包する情報の集合を順位付けて提示するものであり、組織内に特有の問題解決や知的生産活動に必要な情報を見つけ出すのが難しい場合が少なくないためである。この問題を解決するために、本研究では、適応型情報検索フレームワークASF (Adaptive Search Framework) を提案する． ASFは、組織内ユーザが情報検索を行う度に検索結果を保存し学習を繰り返すことで、問題解決や情報収集を目的として組織内の他のユーザが検索を行った際に、その組織で最も関連のある情報を検索結果を返すための枠組みである。本稿では、ASFに基づいて構築した情報検索システムの有用性を確認するために行った実験について報告する。

［keyword］
Search Engine, Web Mining, Web Search Interface, Collaborative Search, Knowledge Management

1. Introduction

In these days, search engine has become a part of our daily lives. People use search engines as a tool to learn how to accomplish tasks, solve problems and gain information. There are many web search engines available, such as Google[1], Bing[2], or Yahoo[3]. Today, we all involve with at least one community. The term of community can refer to a usually small social unit that shares common values. For example, a graduate student has to become a member of a laboratory to do his/her research. We assume that the laboratory is his/her community. Since the advent of the Internet, the concept of community has less geographical limitation, as users can gather virtually in as online community and share common interests. To search for specific information related to topics in community by using major web search engines alone can cause irrelevant results because they return search result based on the on relevance scores reflecting the popularity of the results with the majority of people in the world not specific group of community. However, the problems of using major web search engines inside community can be distinguished into three major problems: (1) Major search engines return results that are sometimes too various and irrelevant to the topics of interest in the community because the search results are ranked based on the general users. (2) Users sometimes need to collaborate with other members when finding specific information to solve a problem in the community, but major web search engines do not provide a collaborative function for users. (3) Community needs to identify experts from users in order to share required information to solve problems efficiently, but most web search engines do not provide support for it.

These motivated us to consider what if there was an additional search engine layer that could learn from what users have searched before to return the top-ranked results that are more relevant to topics related to the community without modifying the conventional search engine itself.

In this study, we propose Adaptive Search framework (ASF), which is a framework designed to be implemented on top of

1 Google: http://www.google.com
2 Bing: http://www.bing.com
3 Yahoo: http://www.yahoo.com

major search engines. It collects and learns from user-provided information when users perform searches in order to return relevant results that related to the topics inside the community. It also includes a collaborative function for users to help other members to search, and the framework itself can classify who has expertise into some fields inside the community.

The rest of this paper is organized as follows. Section 2 describes existing technologies related to our study. Section 3 introduces our prototype system that has been implemented to address the problems we stated earlier. Section 4 describes an experiment to evaluate the usefulness of ASF. The last two Sections, Discussion and Conclusion, explore the results of the experiment and describe future work on ASF.

2. Motivation

2.1. Problem of Existing Custom Web Search Engines

Information in WWW is continuing to grow since Internet has become a part in our lives, but in the other hand, it has become increasingly difficult for users to find information that satisfies their individual needs. There are several approaches trying to solve this problem [8][9]. We indicate the gap of knowledge and possible limitation of existing custom search engines in Section 2.1.1, 2.1.2, 2.1.3 and 2.1.4.

2.1.1. Google Custom Search

Google custom search is a platform provided by Google that 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 of "lotus" to a brand of a 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 of collaborative search for users, 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.

2.1.2. Digg.com

Digg.com[4] 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 search results are ranked based on a majority score.

2.1.3. Bing: Adaptive Search

Bing is a web search engine developed by Microsoft[5]. In September 2011, Bing announced its newest feature, which was called "Adaptive Search". As explained by Adrian Cook[6], 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 provide more personalized results.

Bing still serves on individual purpose. It learns from the data collected from users, predicts the interest of each user from that information, and returns search results related to the topics of interest to that user. As Google Custom Search, Bing does not provide collaborative web search feature for users right now. But Bing is a little ahead of Google. It uses collaborative search behavior to rank results, which mean that Bing makes use of your click behavior while delivering search results for other users

---

[4] Digg.com: http://digg.com/

[5] Microsoft: http://www.microsoft.com/

[6] http://www.searchenginejournal.com/bing-adaptive-search/33515/

searching on the same keyword.

2.1.4. Eurekster.com

Eurekster[7] allows users to custom search portal and harness the knowledge, passion and behavior of online communities to improve the search experiences, while creating online assets for web publishers and enterprisers.

Eurekster launched a personalized social search in 2004 before Google and Yahoo, which is important next step for increasing relevancy. Users can build and customize their search portal on any topic, and share and distribute the social search to grow a community of interested users. The different from Google, Bing and Yahoo is that Eurekster's twisted on the concept of personalize to social which is to provide personalized results based not on who you are but who you know. Friends, colleagues and anyone in your Eurekster social network will influence the type of results you see. But in term search engine for organization, it would be useful to know who has an expertise for which field. Eurekster focuses only on social search technologies.

3. Adaptive Search Framework

We develop Adaptive Search Framework (ASF) to address the problems we stated earlier in Section 1. This section details our ASF: architecture, the data flows involved in using it and the re-ranking algorithm it employs.

RQ1: Can ASF return search results that more relevant to the topics of interest in the organization?
RQ2: Can ASF help user to collaborate with others when perform searching?

We describe the architecture of ASF in Section 3.1. Then in Section 3.2 shows the user interface design of ASF, Section 3.3 describes the data flow inside ASF. Lastly in Section 3.4, we will show you and explain our new iterative page re-ranking algorithm.

3.1. Architecture

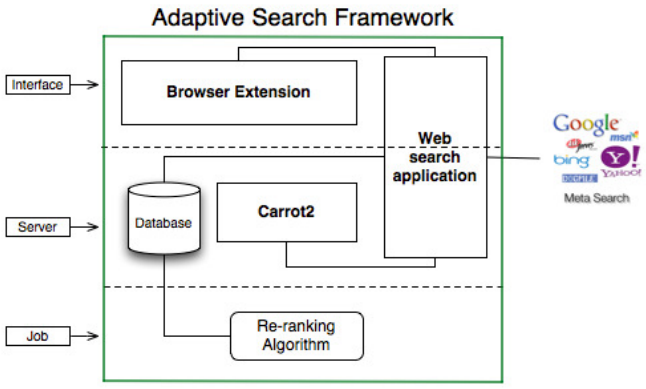


Figure 1: Architecture of Adaptive Search Framework

Figure 2 presents an overview of the ASF architecture. The framework is composed of three layers: interface, server, and job. Each layer serves different purposes and works independently from each other.

The top layer is the user interface layer, where users perform searching and obtain results as same as using normal search engines. We also developed a browser extension that allows the users to organize search results.

The middle layer is the server layer, which returns search results related to the keywords and topics focused in the com-

7 Eurekster: http://www.eurekster.com/

munity. This layer communicates with search engines which this case are Google and Bing by using their APIs. Then we submit a user's query and obtain search results. The frame- work uses Apache Carrot[2] to cluster the results into groups. Tags will be added to those groups to make them more descriptive and meaningful. Those data will be cumulatively stored in the ASF database. Then the server layer returns the search results containing both results from the search engines and results retrieved from data stored in ASF to the user via user interface. 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 called job layer is scheduled on a regular basis to calculate scores for users and web pages using data stored in the database.

3.2. User Interface Design

This section describes a user interface design and data selection for result page of ASF. Result page consists 3 sections. The design in Figure 2 shows the user interface of result page after user has submitted the search keyword to ASF. The Section 1 in Figure 2 we called a suggestion area. In this section shows only top-10 search results that have been analyzed by users inside the community and already stored inside our database. Also the results are already ranked by iterative re-ranking algorithm as present in Section 3.4. Section 2 is related topics section. This section lists all the topics that related to the community keywords that have been pre-defined ahead of time. The last section shows general results that retrieved from Google and Bing in each category that users select inside the related topics section.

After user hit search button to submit the keyword, ASF uses Google and Bing APIs to obtain the general results for the keywords. After we got the general result set, ASF uses Apache Carrot[2] to cluster results into categories. AFS will select categories that related to the topics in the community by using Tag Mapping method. So irrelevant results that we got from major web search engine will be in S3. S1 and S2 contain results that related to the topics inside the community. S1 is the results that are in top-10. S2 is the results, which are related to the topics, but not yet reach into the top-10. We will describe more detail in technique on our data flow in the next section.

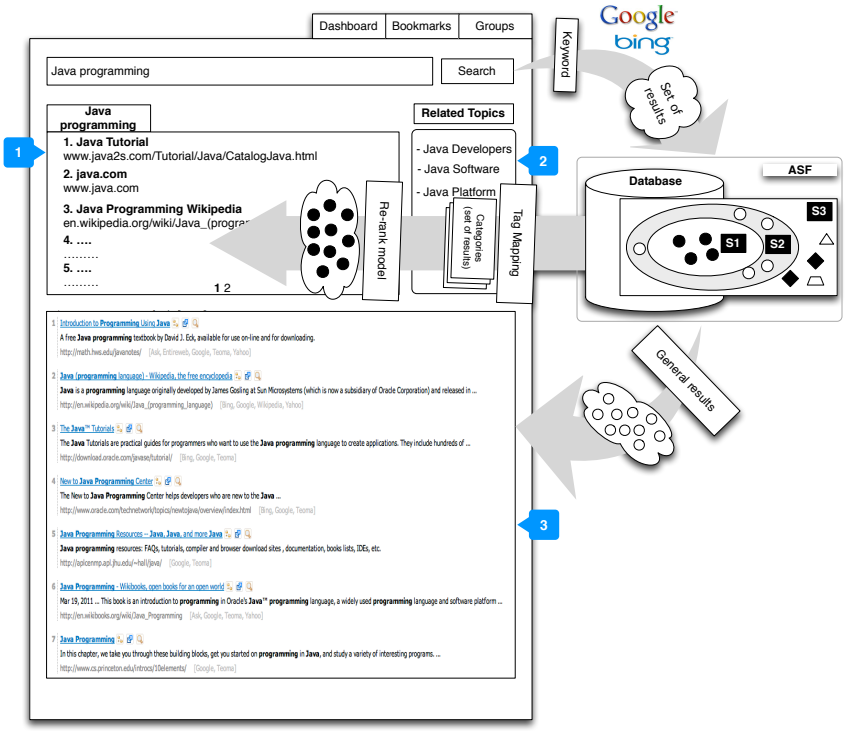


Figure 2: The UI design of Adaptive Search Framework

3.3. Data Flow

In ASF, users perform two main activities: searching and bookmarking. In this section, we explain implementation of our data flow by using two examples illustrated in Figure 3.

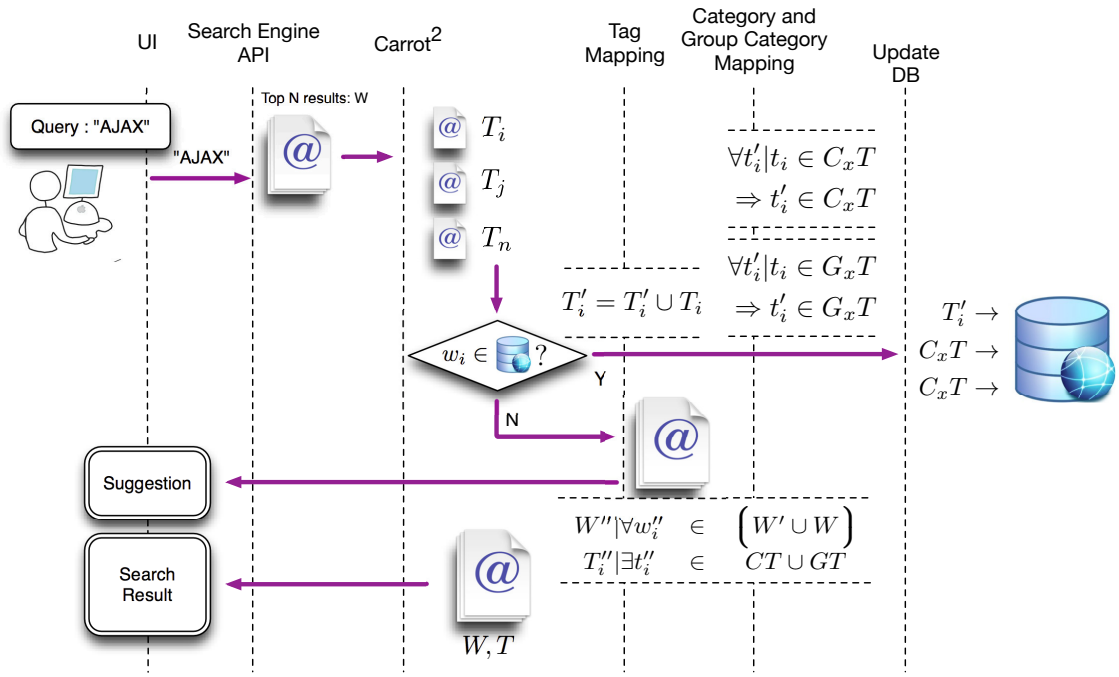
To begin with searching, as shown in Figure 3 (a), a user has submitted query q. It will be sent to conventional search engine API. Top n ranked web pages will be returned as a search result set W. Each w_i also contains three components: "URL," "title," and "snippet." For a further process, we send the whole set W to Apache Carrot[2] API where each component of w_i corresponding tags. At this step, we process two sets, W and T. Next, we check each member of W if it has already been stored in our database. Note that we will explain the store's condition in Section 3.3.1.

In the case where 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 which 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 ($C_x T$ and $G_x T$). That is $\forall t \mid t \in T_i$ will be updated to each category C_x in $C_x T$ linkage where C_x is its corresponded category to t_i.

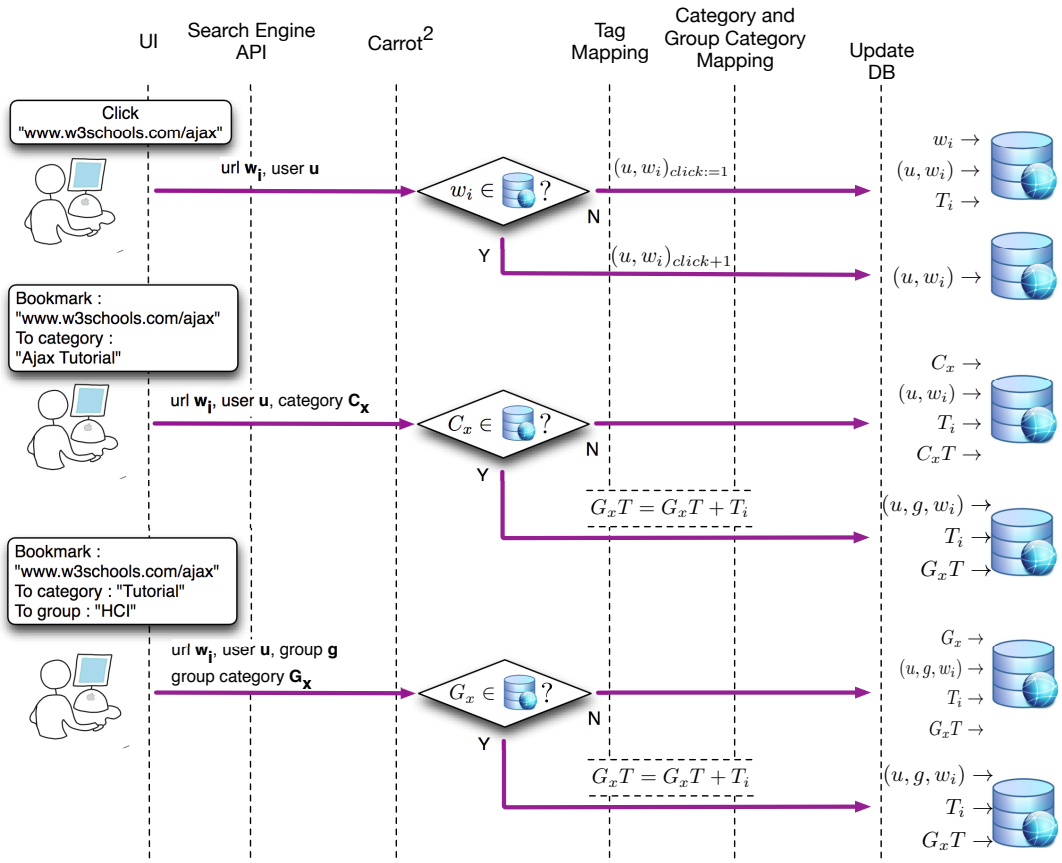
On the other hand, in the case where w_i is not stored in the database, it will only be processed for our suggestion feature. The web pages suggestion list consists of web pages in the database, which has at least one category corresponding to at least one tag of a web page returned from the top n ranking search result set. To achieve that, first of all, we merge all T_i into T ($T = \mathrm{U}_1^n T_i$) and list all C_x that have a link to t_i into CT ($CT = \mathrm{U}_1^n C_x T \mid \exists t_i \in C_x T$) and GT ($GT = \mathrm{U}_1^n G_x T \mid \exists t_i \in G_x T$). 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 processed 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 existed in the database, merging with W that satisfied the tag condition so that it will become $W'' \mid \forall w''_i \in (W' \cup W) \rightarrow w''_i \in W''$ pairs with $T_i'' \mid \exists t_i'' \in (CT \cup GT) \rightarrow t_i'' \in T_i''$. To show the original result set form the conventional search engine, we render only W in that area.

3.3.1 Clicking a Result Link

As shown in Figure 3(b), 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. If w_i has never been clicked by any user, its identity will not be stored in the database. So we need to create it with its initialized click counter by one. We then store user-click with w_i record as a tuple (u, w_i), and the w_i's corresponding tags T_i in the database. If w_i is already existed, we just then update tuple (u, w_i) by increased the click counter by one.



(a) Searching data flow



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

Figure 3: Proposed framework's data flow

3.3.2 Bookmarking a Web Page

We have two bookmarking types in our framework, bookmarking for oneself and for group. The difference between both of them is the feedback scope of suggesting a new web page to a user. Individual bookmarking only influences altogether group of the user.

In case when bookmarked web page has ben clicked through from the user interface, the identity of that web page w_i must be stored 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 correspond to the bookmark page w_i to C_x as C_xT, and store all of them in the database.

However, if a user chooses to bookmark any web pages 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 interface as query that allows tags to be processed as well as the ordinary routine.

3.3.3 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 difference between the bookmarked records 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.

3.4. The Iterative Page Re-ranking Algorithm

To calculate score for webpages based on user's interactions (clicks and bookmarks) inside the community. We need to create a new ranking algorithm. We derived our algorithm from Kleinberg's HITS algorithm [10] and Ranking user's relevance to a topic [refer]. As shown in Equation 1, in this algorithm, the authority weighting of a web page *p* is 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 omega2), individual bookmarks (weight by omega3), 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 omega2), or book- marked by himself (weight by omega3) or group (weight by omega4). Another term is indirectly influenced by the user *r's* weight, which comes from his group participation (weight by 1-omega4) the score in this term comes from web pages that all users in a group have bookmarked as a group.

$$a(p) = \omega_1 \sum_{q \to p} h(q) + (1-\omega_1)\left(\omega_2 \sum_{r \to p} u(r) + (1-\omega_2)\left(\sum_{s \to p} u(s) + (1-\omega_3)\sum_{t \to p} u(t)\right)\right)$$

$$h(p) = \omega_1 \sum_{p \to q} a(q) + (1-\omega_1)\left(\omega_2 \sum_{r \to p} u(r) + (1-\omega_2)\left(\sum_{s \to p} u(s) + (1-\omega_3)\sum_{t \to p} u(t)\right)\right)$$

$$u(r) = \omega_2\left(\sum_{r \to i} a(i) + \sum_{r \to j} h(j)\right) + (1-\omega_2)\left(\omega_3\left(\sum_{r \to k} a(k) + \sum_{r \to l} h(l)\right) + (1-\omega_3)\left(\omega_4\left(\sum_{r \to m} a(m) + \sum_{r \to n} h(n)\right) + (1-\omega_4)\left(\sum_{r \to o} a(o) + \sum_{r \to p} h(p)\right)\right)\right)$$

Equation 1: The Iterative Page Re-ranking algorithm, which is derived from HITS algorithm.

4. Experiments

To investigate the performance of ASF that can return users better search results based on topics in the community and user's interests, we conducted experiments to let participants used our ASF to search for results for target topics which involved with the community. The experiment focused on answering 2 research questions that we mentioned at the earlier in Section 3.

In Section 4.1, we describe the experimental design and procedure. Section 4.2 presents the results of a comparison ranking between ideal results, major search engine results and ASF results, and also show the post-interview results from users on the satisfaction of the ASF's usage.

4.1. Experimental Design and Procedure

We used a between-subjects design with two conditions: ASF condition and the major search engines condition. We deployed our ASF on Ubuntu server in Software Engineering Laboratory at Nara Institute of Science and Technology[8]. The framework was implemented on top of two major search engines, Google and Bing. To let ASF learn to return better results, we asked thirty participants who are either students or researchers from three research groups, which are software matrices, software maintenance and human computer interface to use the ASF for two weeks.

To measure how search results from ASF are better for members inside software engineering laboratory more than using major web search engines alone, we chose top 5 most searched keywords in the laboratory as shown in Table 1 as target topics. Then we asked twenty participants to do the blind selection on the results based on the topics and rate the score based on each criterion as shown in Figure 5. Fifteen participants used ASF for two weeks and the other five participants were new comers.

To measure the satisfaction of the ASF's usage, we interviewed thirty participants after using ASF for 2 weeks. We created a post-test survey containing eight simple questions. The range of the lowest rate to highest rate is from 0 to 7. These questions are carefully selected to show how ASF is effective.

4.2. Experimental Results

In this study we aim to answer RQ1 and RQ2, In Section 4.2.1, we show the results that compare ASF return results set with other major web search engine results and how effective of ASF that suggest websites that have not been viewed by users, but

[8] Nara Institute of Science and Technology: http://www.naist.jp/en/

are related to keywords and topics that the users are interested in to answer RQ1. In Section 4.2.2, we solve RQ2 by describing our collaborative function and show to ability of it to gain better result into the community.

4.2.1 Compare Results Set between Search Engines

1. Ajax tutorial	2. Software Metrics for Agile Software Development
http://www.w3schools.com/ajax/	http://blog.ness.com/spl/bid/72570/The-Two-Agile-Programming-Metric...
http://www.youtube.com/playlist?list=PLE0071B4091E8948D	http://www.methodsandtools.com/archive/archive.php?id=61
http://www.tutorialspalace.com/2012/01/35-useful-ajax-tutorials-for…	http://www.slideshare.net/2h74webere/agile-metrics-12275220
http://www.maxkiesler.com/2006/03/15/round-up-of-30-ajax-tutorials	http://www.slideshare.net/2h74webere/agile-metrics-12275220
http://www.codeproject.com/KB/ajax/AjaxTutorial.aspx	http://www.agileconnection.com/article/agile-development-and-softwar…

3. Web service	4. Empirical Study in Software Engineering
http://en.wikipedia.org/wiki/Web_service	http://www.idi.ntnu.no/grupper/su/publl/ese/zannier-studytypes-icse06.pdf
http://www.w3schools.com/webservices/	http://www.computer.org/portal/web/swebok/html/ch6
http://www.w3.org/TR/ws-arch/	http://userpages.umbc.edu/~cseaman/papers/tse99.pdf
http://ws.apache.org/	http://www.cs.umd.edu/~basili/presentations/2006/Role%20of%20E%
http://www.w3.org/DesignIssues/WebServices.html	http://www.slideshare.net/sarfraznawaz/empirical-research-methods-for-s...

5. Software Maintenance Technique
http://www.cs.ucf.edu/~turgut/COURSES/.../PA-chapter11.ppt
http://www.sciencedirect.com/science/article/pii/0950584991900257
http://www.computer.org/portal/web/swebok/html/ch6
http://www.cs.toronto.edu/~sme/papers/2007/SelectingEmpiricalMethods.pdf
http://www.idi.ntnu.no/grupper/su/publ/ese/zannier-studytypes-icse06.pdf

Table 1: Top-5 keyword result sets in Software Engineering Lab re-ranked based on 30 participants

Keyword: **Ajax Tutorial**

ASF/Ranking	URL
1	http://www.w3schools.com/ajax/default.asp
2	http://www.templatelite.com/ajax-tutorials
3	http://www.tutorialspalace.com/2012/01/35-useful-ajax-tutorials-for-web-developers/
4	http://www.maxkiesler.com/2006/03/15/round-up-of-30-ajax-tutorials/
5	http://www.codeproject.com/KB/ajax/AjaxTutorial.aspx

Google/Ranking	URL
1	http://www.w3schools.com/ajax/default.asp
2	http://www.xul.fr/en-xml-ajax.html
3	http://www.learn-ajax-tutorial.com/
4	http://www.tizag.com/ajaxTutorial/
5	http://code.google.com/edu/ajax/tutorials/ajax-tutorial.html

BING/Ranking	URL
1	http://www.w3schools.com/ajax/default.asp
2	http://tutorialajax.com/
3	http://www.ajaxtutorial.net/
4	http://soloscript.com/
5	http://msdn.microsoft.com/ja-jp/library/bb470455.aspx

Figure 4: Showing top-5 return result of "Ajax tutorial" from ASF, Google and Bing. The result that came from ASF are ranked by our iterative algorithm

To answer the RQ1, how good are the return results from ASF, we chose top-5 most search keywords in SE lab to measure our search results comparing to other search engines results. In table 1, we show the results of top-5 keywords that returned from ASF and have been re-rank by using our algorithm. In Figure 4, we show the example ranking of ASF, Google and Bing together on "Ajax tutorial" keyword. Then we blind out the source of result set for each keyword so participants won't

recognize that which set of results come from which search engine. Then we created a rating program that show up each set of result and also simple task that related to the keyword. Next we gain our participants to solve the tasks by using the each result set that shown up and rate them based on 4 criterions which are 1) Usefulness, 2) Easy to find solution, 3) Accuracy of information and 4) Liked.

As you can see in Figure 5 that participants seems to prefer the results that came from our ASF better than Google and Bing. The reason is that all the links that returned from ASF are re-ranked by using our new iterative algorithm, which based on the interests of our users inside the organization. In other hand, Google and Bing ranked the results based on the majority of people in the world. In term of an organization, the results from ASF can be considered better than the existing search engines.

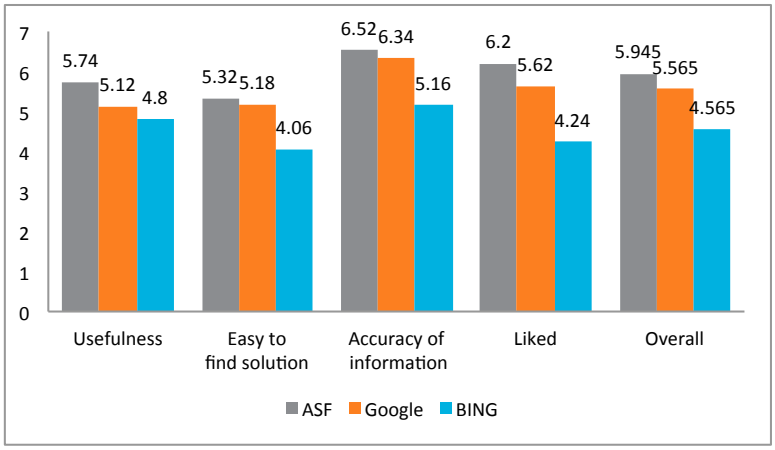


Figure 5: Average rating score from 20 participants for each criterion. 15 participants were participated on using ASF for 2 weeks. 5 participants were recruited later on as new comers. The score were calculated by allowed participants to observe each link and rate it by following our criterions. Participants seem to satisfied links from our ASF more than Google and Bing.

To help address more the first research question, we examined the ability of ASF 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 ASF would relate "Superman" and "Clark Kent". As shown in Figure 6, in the search page 1, 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 user clicked or bookmarked some of the websites about the superman movie, as shown in the page 2, 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, ASF provides information about movies based on the data from the previous search. Finally, in page 3 shows that when user searches for "Clark Kent," Superman's secret identity, ASF 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 ASF can suggest results better related to a user's interests and topics.

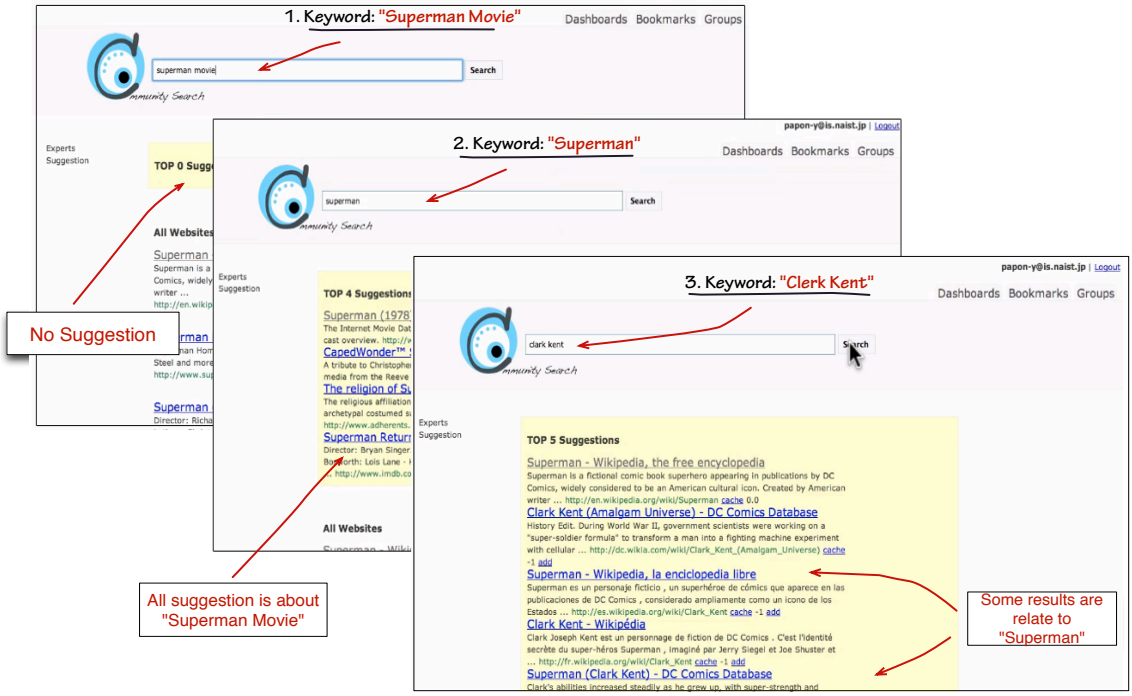


Figure 6: 1. At first, the framework cannot suggest anything when user searches for "Superman Movie" because it is a new topic for the framework. 2. After user has interacted with some of the websites that involve Superman, the framework can suggest some websites when user searches for "Superman" based on the previous search data. 3. Also, the framework can suggest Superman when user searches for Clark Kent

4.2.2 User Collaboration

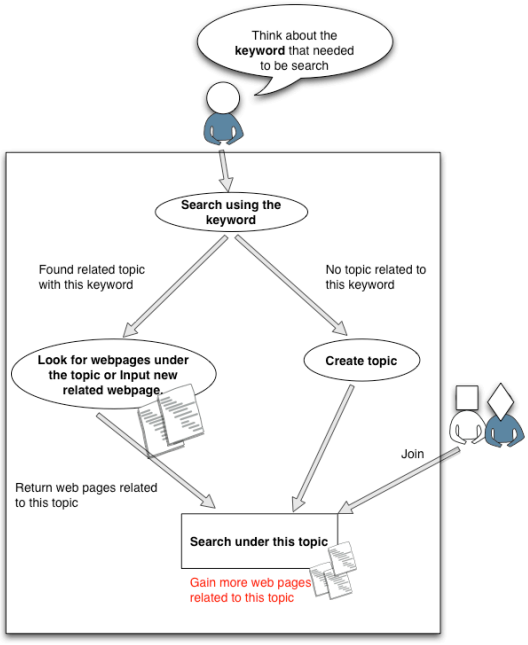


Figure 7: Collaboration flow and result page. On the left shows when user start to perform a search and do the collaboration with others. On the right shows the structure of the result page.

One of the most interesting applying collaborative functions into ASF is the creativity of the results. Figure 7 on the left shows the flow when user performs a search and collaborates with others. ASF allows user to create a new topic, which related in the organization. Once the topic has been created, users in the organization can help each other search under the topic to gain more relevant web pages. On the right of Figure 7 shows the structure of the result page. For example, user searches for "Java

programming", we assume that "java programming" topic has been already created inside the organization. In area 1, ASF will show the results, which have been analyzed by users and re-rank by the algorithm that they are useful links for the organization. Area 2 shows the related topics, which might also contain useful results. In Area 3 shows ordinary result, which obtains from regular search engines that we build on top.

To measure the collaborative function, we observed our participant's satisfaction by asking them these eight simple questions after using our system. As shown in Figure 8, first questions are to see *which search gives better results relate to an organization*. The result shows the search results from ASF satisfied our users the best because users thought that the search results gained from ASF are more related to the topics inside the organization.

The next two questions are *how easy to organize and share search results*, with search organization feature in ASF. Participants felt more comfortable to share and organize their search results among members inside their research groups.

Next is to show *how useful of information each search engine gives*. We expected this to be higher than Google, Bing and Yahoo and the result shows exactly what we expected. Most of search results that are shown up for participants have been analyzed by others users in the organization before one of participants said, "Many times when I search, most of the information that I needed can be found in the early beginning of the first result page."

Next is *time reducing while searching*, since the information that participants tried to search are found in the top ranks, participants felt that ASF helped them save a lot of time to search.

Next is to see *if participants enjoyed using out ASF*. Many participants said, "There are many useful features that Google, Bing and Yahoo don't have and also it gave back search results that has been already analyzed by people in the organization."

The last two is to check *how easy to learn and how easy to use*, with the many features inside our ASF. Participants felt that ASF helped them to get useful information. Based on post-test survey as shown in Figure 8 one participant said that "This framework provides organizations to have their own search engine". One person said, "It gave me better search results compared to major search engines". Another said, "It saved me a lot of time to search". We do note that the tasks that we had participants do was limited in scale, but they were enough to evaluate the system and prove our basic assumption of our ASF that we committed at the early section.

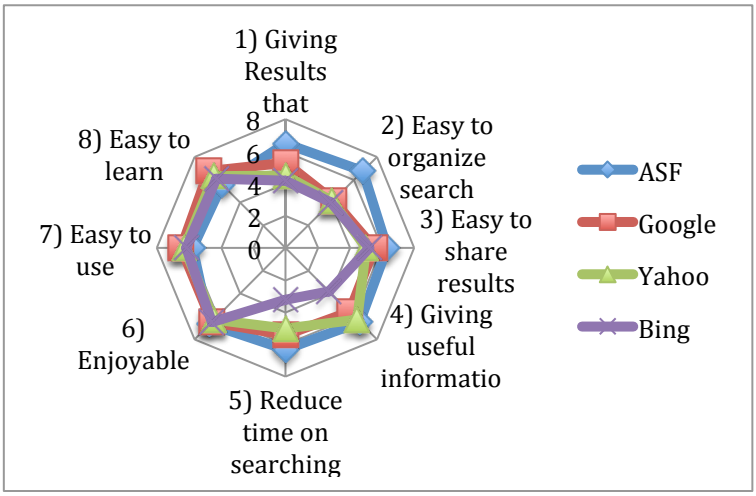


Figure 8: Participant ratings of ASF. The result has shown that participants seemed to satisfy with the overall usage of ASF.

5. Discussions and Future works

How can ASF return better-suited results to community? 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 Software engineering laboratory. In this relation graph, users are also separated into their own special groups. Figure 10 inside the square blue rectangle area, also shows website ids 8, 4, 3, 117, and 116, the top 5 results shown in Figure 4 for "Ajax tutorial" query. ASF selected these five websites for the suggestion box because they have a high ranking as shown in Figure 9 and are related to be "Ajax tutorial" keyword.

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/
117	0.0327	http://www.codeproject.com/KB/ajax/
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
..	..	..

Figure 9: Scores and ranks of webpages inside Software Engineering Lab. Using iterative re-ranking algorithm.

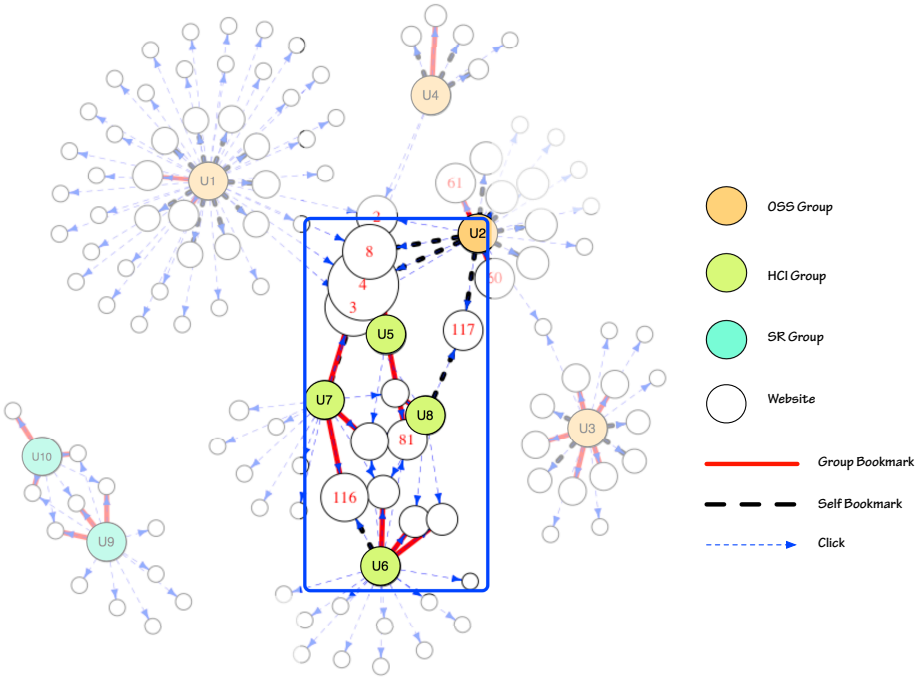


Figure 10: Relation graph between users and websites inside Software Engineering Laboratory.

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, ASF 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, and returns them as suggested websites.

Does the size of community affect ASF's effectiveness? Our experiments only show that ASF works effectively on a small-sized community. For a larger sized community, which may have, more than hundreds of member, we assume that people will produce noises inside the database. We plan to do more experiments on a larger sized community in the future.

What about more general topics? The results from the experiment shows that ASF yield better results when it is in a community that focuses on some specific area but it may not be optimal for the general topics. The information based on the majority of people in the world could return the best results for the general topics.

How can ASF serve members with different level of knowledge? The keys purpose of ASF are the get rid of irrelevant results that do not related to the topics inside the community and to help member reduce time locating the useful results. There are several aspects on effectiveness that can serve both high level of knowledge member and novice member. For example, high-level members can earn the benefits from using ASF to get rid of irrelevant results. They are also more likely to use ASF to contribute useful results to the community. For novice members, ASF helps them reducing time locating the results that are useful and relevant to the topics in the community.

Are we trying to replace major web search engine? No, we are not. What we are trying to do is to improve the search results selection method for a community. We suggest that the most effectiveness method for information seeking tasks is to using both major web search engine to search for general topics and ASF to search for specific results that related to the community.

For the future work, we would like to continue to improve the performance of ASF to reduce unrelated results (noise). At this point, ASF 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 will try to analyze web pages using different tags to reduce such extraneous results. Also we would like to do more experiment on the different type and size of organization. To determine whether the framework can satisfy people in real organizations and provide better search results than directly using major search engine. In this prototype of ASF does not yet identify experts. But the algorithm and method that give score and classify experts are already developed but because the system needs quite a lot of time to learn to classify the expert. As a next step, we will need to predefine a set of experts related to the organization and do time more on the experiment.

6. Related Work

Most of the previous research [8][9][12][24] on in search results mining has focused on the personal based ranking rather than the community. Our goal of this research is to creating a search engine that can extract knowledge for organizations based from the use of search engine. Clustering and annotating [7] 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 [9], Vivisimo[10], and IBM Mapuccino[11]. In this study, we use Apache Carrot^2 since it is an open source library augmented with a set of supporting applications. This allows us to build a clustering search 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 results. This limitation can be eliminate by our users inside the organization itself by manually categorize results into the topic.

7. Conclusions

In this paper, we described our study to help users and community to obtain search results that are more relevant to topics of interest within the community. We proposed and developed an Adaptive Search Framework that uses data provided by users performing ordinary searches, clicking on links, and bookmarking within a community to enrich and select responses to searches.

We performed a series of experiments with ASF, which suggests the framework could return more relevant results and

---

[9] Apache Carrot^2: http://project.carrot2.org/applications.html

[10] Vivisimo: http://en.wikipedia.org/wiki/Vivisimo

[11] IBM Mapuccino: http://www.research.ibm.com/topics/popups/innovate/java/html/mapuccino.html

additional results related to searchers. Using ASF inside a community could benefit both the users and the community. When the users search something, they can save time and obtain search results that are relevant to the topics of interest in the community. The community also can obtain information about topics of interest within the community, and maintain an community knowledge about web information. We believe that over time, users obtain better search results and help organizations gain better productivity via ASF.

[References]

[1] Janis Grundspenkis. Agent based approach for organization and personal knowledge modeling: knowledge management perspective. Jornal of Intelligent Manufacturing Vol. 18, Issue 4, pages 451-457, 2007.

[2] Sveiby, K.-E. What is knowledge management? http://www.sveiby.com.au/KnowledgeManagement.html, 2000.

[3] Tsui, E. Technologies for personal and Peer-to-Peer (P2P) knowledge management, CSC Leading Edge Forum Technology Grant Report, CSC Leading Edge Forum Technology Grant report, 2002.

[4] Razmerita, L.; Kirchner, K.; Sudzina, F. "Personal Knowledge Management: The Role of Web 2.0 tools for managing knowledge at individual and organisational levels", Online Information Review 33 (6): 1021–1039, 2009.

[5] Jo Smedley. Modelling personal knowledge management. OR Insight 22, pages 221-233, 2009

[6] Kirby Wright, Personal knowledge management: supporting individual knowledge worker performance. Knowledge Management Research & Practice, pages 156-165, 2005.

[7] S. Bao, G. Xue, X. Wu, Y. Yu, B. Fei, and Z. Su. Optimizing web search using social annotations. In Proc of WWW'07, pages 501–510, New York, USA, 2007.

[8] Kazunari Sugiyama, Kenji Hatano and Masotoshi Yoshikawa. Adaptive web search based on user profile constructed without any effort from users. In Proceedings of the 13th international conference on World Wide Web, pages 675-684, 2004.

[9] E. Agichtein, E. Brill, and S. Dumais. 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, pages 19–26. ACM, 2006.

[10] JonM. Kleinberg. Authoritative sources in a hyperlinked environment. J.ACM, 46:604–632, September 1999.

[11] Claudio Carpineto, Stanislaw Osin´ski, Giovanni Romano, and Dawid Weiss. A survey of web clustering engines. ACM Comput. Surv., 41:17:1–17:38, July 2009

[12] Susan Dumais, Edward Cutrell, and Hao Chen. Optimizing search by showing results in context. In Proceedings of the SIGCHI conference on Human factors in computing systems, CHI '01, pages 277–284, 2001.

[13] Andreas Hotho, Robert Jaschke, Christoph Schmitz, and Gerd Stumme. Information retrieval in folksonomies: Search and ranking. In The Semantic Web: Research and Applications, volume 4011 of Lecture Notes in Computer Science, pages 411–426. Springer Berlin / Heidelberg, 2006.

[14] Said Kashoob, James Caverlee, and Krishna Kamath. Community-based ranking of the social web. In Proceedings of the 21st ACM conference on Hypertext and hypermedia, pages 141–150. ACM, 2010.

[15] Yan Li, Xin-Zhong Chen, and Bing-Ru Yang. Research on web mining-based intelligent search engine. In Machine Learning and Cybernetics, 2002. Proceedings. 2002 International Conference on, volume1, pages 386–390, 2002.

[16] 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

[17] MeredithRingel Morris and Eric Horvitz. Searchtogether: an interface for collaborative web search. In Proc of UIST '07, pages 3–12, 2007.

[18] Adaptive Search. http://www.bing.com/community/site blogs/b/search/ archive/2011/09/14/adapting-search-to-you.asp.

[19] GoogleCustom Search. http://www.google.com/cse/

[20] 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.

[21] 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

[22] Jaime Teevan, Christine Alvarado, Mark S. Ackerman, and David R.Karger. The perfect search engine is not enough: a

study of orienteer- ing behavior in directed search. In Proceedings of the SIGCHI conference on Human factors in computing systems, CHI ’04, pages 415–422. ACM, 2004

[23] 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

[24] Xuanhui Wang and ChengXiang Zhai. Learn from web search logs to organize search results. In Proc of the SIGIR ’07, pages 87–94, 2007.

[25] 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.

[26] Shenghua Bao, Guirong Xue, Xiaoyuan Wu, Yong Yu, Ben Fei and Zhong Su, Optimizing web search using social annotations, In Proceedings of the 16th international conference on World Wide Web, pages 501-510, 2007.