Community Search: A Collaborative Searching Web Application With a User Ranking System

Papon Yongpisanpop, Masao Ohira, Ken-ichi Matsumoto

Graduate School Of Information Science, Nara Institute of Science and Technology
8916-5, Takayama, Ikoma, Nara, Japan
+81-743-72-5312
{ papon-y, masao, matumoto } @ is.naist.jp

Abstract. People are using search engine in daily life. But most of the tools that we have today treat information-seeking tasks as a transient activity. In this research paper we introduce a web application system that provides collaborative function and experts finding system. We develop a system that will help user to organize search result and to do the collaboration with others. With the new iterative algorithm, users will also gain more percentage of needed result and the system will be able to suggest the experts related to the search keyword.

Keywords: Search Engine, web mining, web search interface, collaborative search

1 Introduction

Although search engines have become a part of our daily lives and allow people to access the vast quantities of information available online, tools that we have these days typically treat information-seeking tasks as a transient activity, and consider each of a user's search and browse action as unrelated events rather than as part of a larger task. For example, if a user is searching for a car to buy in the future, he/she will perform several low-level interactions with a web browser and one or more search engines (e.g., typing the URL of a search engine site, querying for "Nissan Skyline," clicking a result, hitting the back button, submitting a new query and so on). In other words, tools such as web browsers and search engine sites are designed to support single-user scenarios. They provide the contents of the web page from the query of a user, but they don't help the user to decide which results are good and useful.

In the past few years, Microsoft researchers started to study these problems. They separated the topic into two main problems and created tools for solving the problems: *Searchbar* [1] and *SearchTogether* [2]. *Searchbar* is a plug-in for the Microsoft Explorer browser to keep track of and organize a search query, so a user can continue the same search at a later time without having to start over.

SearchTogether is software that runs on the Windows operating system and provides a means for **collaborative search** in which users can help each other when searching together for specific topics. However, These two tools still do not cover the entire problem in supporting people's information seeking activities. The problem with the first tool is the user is still doing the individual search. The program just creates an interface and organizes the contents to avoid the same task in the future. The problem with the second tool is you need to have software running on the Windows operating system, which means it is not really convenient for users of other operating systems. Another problem is sometimes a search engine will return the results based on the score that the algorithm calculate it from how many time that this page has been clicked which mean that there is no guarantee for the users that the result will match for his/her purpose. Also when user can't find any good result, sometimes user needs to ask some one who has an expertise about the field.

In this paper, we propose the concept Community Search, which is a community based web technology that acts itself as a Meta search portal and allows users to manage search result contents. It provides the collaborative searching function for users to share search results with others and to get the contents that has been analyzed by real users in the community. It also suggests a user the existences of experts who are related to user's search keywords, so that user can have reliable information the experts recommend. Community Search is implemented using our new iterative algorithm. This algorithm is derived from HITS (Hyperlink-Induced Topic Search) algorithm [3] that is the most famous algorithm for search engine. The purpose of this algorithm is to help users get a better search result by iteratively running it. The algorithm is used to collect keywords and links which connect the web pages to users and web pages to web pages. Then it weights and gives the score for each web page and also ranking user's relevance to a topic through link analysis on web logs We need to make our algorithm to have enough accuracy so that can drive the community search to return good results and be able to suggest the experts related to the keyword that a user is searching for.

The rest of this paper is organized as follows. In section 2, we present related work on organize search query, collaborative searching and expert finding. In section 3, we describe the concept of Community Search. In section 4, we present the proposed approach of Community Search's collaborative searching function and our new iterative algorithm that can calculate score for users and web pages. Then, we show how Community Search classifies experts. In section 5, we will give an example on usage scenario. In section 6, we will discuss about the usage of Community Search. And finally the conclusion and future work will be the last part in section 7.

2 Related Work

A number of web pages ranking [8], automatic expert finding and collaborative search system have reported it so far. For example in automatic expert finding system, HelpNet [5] and Expert/Expert-Locator (EEL) [6][7]. [5] allow user filled information to provide the expertise profile and [6][7] use a representative collection of user's technical document to build the expertise index. There are a lot of previous system that use textual analysis as the basis for expert finding but [9] uses graph analysis to find interlinked cliques for interest group and [10] that uses users' visit path record of experts. So our study will focus on [9][10]'s technique and adapt it into our iterative algorithm. As we mentioned earlier about 2 of Microsoft research that we have been studied them, which are *SearchBar* [1] and *SearchTogether* [2]. Searchbar is a user interface for web search, including browsers and search engine sites, which is a system for proactively and persistently storing query histories, browsing histories, and users' notes and ratings in an interrelated fashion. For the SearchTogeter is software designed to enable either synchronous or asynchronous remote collaborate when searching the web. Both of these systems are designed to help user to make searching more productive in the interface level. But what we need in our research is to develop a system that can be able to help user to do the collaboration, organize search result contents and also classify the experts out of the community. Then we have found a research called "Ranking User's Relevance to a Topic through Link Analysis on Web logs" [4], which is close to what we need to make Community Search smart enough to return a better search result and be able to classify experts. Basically this research derived HITS algorithm [3] to a link analysis algorithm, which evaluates the web user's level of expertise on a given topic by his/her web browsing record.

3 Community Search

Community Search is a platform independent, web-based technology that allows users to manage search result contents and also provides the collaborative searching function for users to share and obtain contents from others. For the advantages, Community Search provides users with a means for gathering people who are interested in the same topic and/or share the context of work/hobby/everyday life, it will change the old style of searching and make searching more productive and useful.

The structure of Community Search is separated into two parts. Front-end which provides collaborative searching and search result organize functions for the users. Back-end part, which is using our new iterative web pages and users, ranking system that will be able to calculate score for web pages and users for the better search result. Also the new iterative algorithm will be able to classify experts that related to the keyword that users use for search out of the community, we first describe the general model and the link analysis algorithm and then we describe the procedure of how the algorithm giving score for web pages and users and also classify the experts from users habit.

When user uses Community Search to do the search, our system will keep track of what action user does. For Example, user might query, "Ruby on Rails". And then click on a few links to look at the content of the pages. User might also bookmark the page that he/she thinks that is match with his/her purpose. Community Search will use that information to calculate the score and ranking the web pages and users. Also will be able to classify the expert among the users. Next, we will explain our iterative algorithm that derived it from HITS algorithm [3].

4 Implementation

We separated the implementation of the Community Search into 2 parts. First is the front-end part, we designed the interface based on 2 main functions, which are *search result organize function* and *collaborative searching function*. In this part we need to study a user behavior that using the system from the previous researches, which are SearchBar [1] and SearchTogether [2]. And then we will design the interface that fits for Community Search using Ruby on Rails[1]. Second is the back-end system is implemented using our new iterative ranking algorithm. This algorithm is derived from HITS (Hyperlink Induced Topic Search) algorithm [3] that is the most famous algorithm for search engine. The purpose of this algorithm is to help users get a better search result by iterative running it. The algorithm is used to collect keywords and links which connect the web pages to users and web pages to web pages. Then it weights and gives the score for each web page and also ranking user's relevance to a topic through link analysis on web logs.

The data in Community Search can be model as a directed graph, as shown in figure [1]. In this model there are two kind of entities represented by nodes, which are users and the web pages. There are 3 kinds of directed edges between the nodes. The dark lines with white arrow head are represented the links between web pages. The dash lines are when the users bookmark the web page. And the dark lines with small arrow head are represented the links from users to web pages (when user clicks to visit the page). In this graph, the edges between web pages are hyperlinks links and the edges from a user to a page indicates that the user visited or bookmarked the page. The directed edges can be assigned numeric weights to reflect the different importance. To define the expert, the visit and bookmark edge from those users may be assigned with higher weights.

[1]Ruby on Rails is an open-source web framework that's optimized for programmer sustainable productivity.

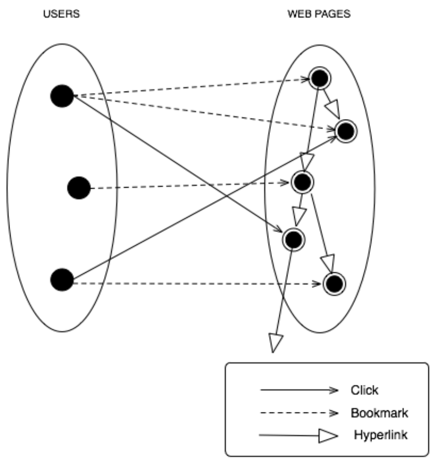

Fig. 1. The directed graph model showing the interaction between users and web pages in Community Search

4.1 The Iterative Algorithm

As we mentioned, this iterative algorithm is derived from Kleinberg's HITS algorithm [3], There are three assumptions in calculating the importance of users and web pages. The first assumption is that the more high quality web pages the user has visited, the user will gain more experience. The second assumption is the more frequently a web pages is cited by other pages and visited by experienced users, the higher quality the web page is. And the last assumption is the importance/quality of the users and pages may reinforce each other in an iterative way.

$$a(p) = \beta \sum_{q \to p} h(q) + (1-\beta) \sum_{r \to p} u(r)$$

$$h(p) = \beta \sum_{p \to q} a(q) + (1-\beta) \sum_{r \to p} u(r)$$

$$u(r) = \left[\alpha \left(\sum_{r \to p} a(p) + \sum_{r \to q} h(q) \right) + (1-\alpha) \left(\sum_{r \to y} a(y) + \sum_{r \to z} h(z) \right) \right]$$

Fig. 2. The iterative ranking algorithm that runs on the background of the Community Search to ranking user and giving score for web pages.

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 weights of all pages he/she 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, $\boldsymbol{\beta}$ is the parameter to adjust if the system needs to weight more score on when the user clicks on the page. And $\boldsymbol{\alpha}$ is the parameter to adjust if bookmark is more importance to give more weight score to the user.

4.2 How to classify experts

Community Search can identify experts for a given query topic and rank users' level of expertise by collecting the keywords that user used to search and then categorize web pages that users interact with (click and bookmark). Then the iterative ranking algorithm will calculate score for users and web pages. The result from the algorithm will be able to tell which topic users have been experienced on.

To do this, we need to create a program to collect keywords and be able to categorize them into their own categories that we already prepared beforehand in the system. This program divided into two parts. The first part as show in the figure [3] is to create a model that will be able to predict which category the keywords should be. The model that we will use to predict which category the keywords should belong to is needed to be train first. The step of training the model has shown in the figure [3]. First we have to prepare all the possible categories that we can have in the system. For example, if you are using the Community Search in the Software Company, most of your category should involve with software and programming languages. After we define the possible categories that we should have. Then we will build up the model based on the hypothesis that if the documents have a lot of the same keywords of the category; it means that this document should be in the category. For example, the documents in "Java Programming" category should have a lot of "java" keywords inside the document. So in figure [3], We send the list of our category to search in the Google. Then we check the web pages that returned from the Google to see what keywords that the web pages contain and we use the Bayesian network[2] to build the classification model.

[2]Bayesian network is a probabilistic graphical model that represents a set of random variables and their conditional dependencies via a directed acyclic graph.

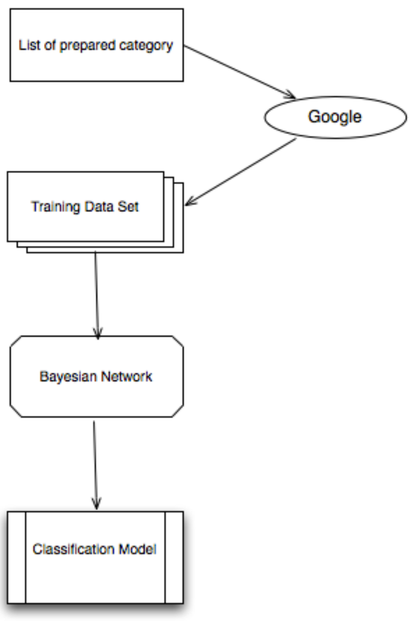

Fig. 3. Showing the step of how to build a classification model

The second part as shown in the figure [4] is to predict which category the keyword should be by using the trained model from the figure [3]. After we got the classification model from figure [3], Then when user insert the keyword to search, we will send that keyword to Google again and give the return result from the Google to the model that has already been trained. Finally The System will now be able to tell which keyword should goes to which category

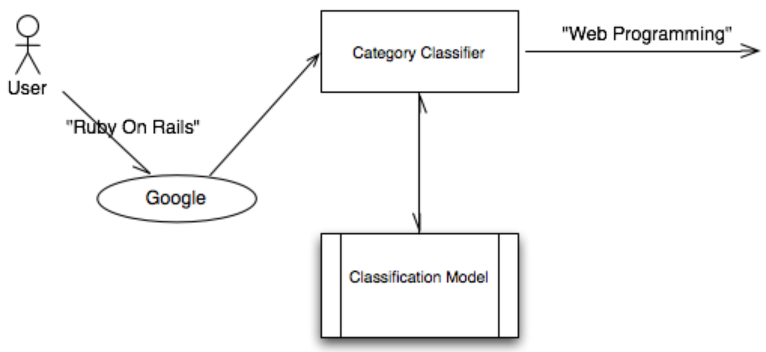

Fig. 4. Show the steps of the classify process. User inserts the query, which is "Ruby on Rails". Then the system use Google to search the result from the keyword. At last using to trained classification model to categorize the category of the keyword, which is "Web Programming".

5 Example Usage Scenario

Community Search will work best if we deploy it into a specific organization. Because of The topics that use to classify experts can be limited due to which type of the organization. Also the collaborative function will be really useful in the situation when you have to search as a team.
For example, we have a car dealer company. And these is a situation that a customer call and ask for a new type of a car, which our company doesn't have any information yet.

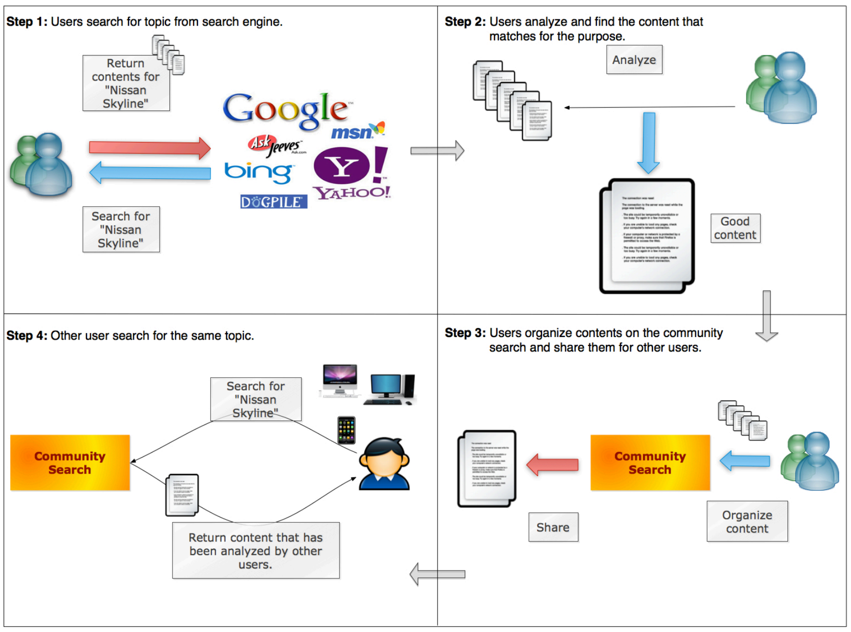
Step 1: Users search for topic from search engine.
Return contents for "Nissan Skyline"
Search for "Nissan Skyline"
Google
msn
Ask Jeeves
bing
Y!
YAHOO!
DOGPILE
Step 2: Users analyze and find the content that matches for the purpose.
Analyze
Good content
Step 3: Users organize contents on the community search and share them for other users.
Community Search
Share
Organize content
Step 4: Other user search for the same topic.
Search for "Nissan Skyline"
Community Search
Return content that has been analyzed by other users.

Fig. 5. Briefly illustrates the idea and flow of user behavior in Community Search

[Step 1] User search for a Nissan Skyline, which is the car that customer wants to know the information. Then obtain search result contents about the car from search engine.
[Step 2] User analyzes the contents of the results and may find useful information.
[Step 3] User can organize the contents by using Search Organize Function in the Community Search. And also user can be able to share this information out to the Community Search. Other users who have the same interests into this information can use it.

[Step 4] Next time, when a customer calls and asks for "Nissan Skyline" again to the difference user. The new user can search and use the contents that have been shared by the user before. Or when the user cannot really find the information that he/she needs. Community Search will be able to suggest experts who related with the topic.

6 Discussions

In this section, we discuss about the usage of Community Search. As we said in the example usage scenario part that Community Search will work best if deploy in some sort of organization. First reason, in the classify experts system part; we need to limit the keywords in the system because we need to prepare the category of the keywords that can be possible for user to search beforehand. For example, if you deploy Community Search in a car dealer company. The limitation of keywords would be something only about cars (truck, four-wheel drive, sedan, etc.) So the system doesn't need to collect the unnecessary keywords to classify the expert. In the car dealer company, the system will have the experts that involves only with cars. Second reason, in the organization, people are always assigned to do the task as a team. In this case, the collaborative function will work best in the situation that they need to do the research together.

The system needs sometime to collect web logs data to analyze and calculate score for web pages and users before it can classify the experts and re-ranking web pages. When users use Community Search, the system will gain more data to analyze and increase the percentage of accuracy.

7 Conclusion & Future Work

We presented Community Search, a collaborative searching web application with a user ranking system. The goal of this study is to make a system that can help users to search more productively. In our study, we found that the interface design of Community Search will be able to help supporting the group of organization to do the collaborative searching and organize search result. Also we found the percentage of gaining needed search result contents increases by using the community search's iterative 2algorithm.

For the future work, right now we completed the program of iterative ranking algorithm that will be running in the background of the Community Search. Next step, we need to make an interface, which is the front-end part of the system that helps users to organize search result contents and do the collaborative searching. In this part we need a really well design for the interface and we will use Ruby On Rails, which is a web application framework to build up our interface system.

8 References

1. Morris, D., Morris, M.R. and Venolia, G., SearchBar: A Search-Centric Web History for Task Resumption and Information Re-finding, Proceedings of Proceeding of the twenty-sixth annual SIGCHI conference on Human factors in computing systems (CHI'08), pp.1207-1216, 2008.
2. Morris, M.R. and Horvitz, E., SearchTogether: An Interface for Collaborative Web Search, Proceedings of the 20th annual ACM symposium on User interface software and technology (UIST'07), pp.3-12, 2007.
3. Kleinberg, J., Authoritative sources in a hyperlinked environment, Journal of the ACM (JACM), Vol.46, No.5, pp.604-632, 1999.
4. Jidong W., Zheng C., Li T., Wei-Ying M. and Liu W., Ranking User's Relevance to a Topic through Link Analysis on Web Logs, Proceedings of the 4th international workshop on Web information and data management
5. M. E. Maron, S. Curry and P. Thompson, An Inductive Search System: Theory, Design and Implementation, IEEE Transaction on Systems, Man and Cybernetics, vol. SMC-16, No. 1, pp.21-28, 1986
6. L. A. Steeter and K. E. Lochbaum, An Expert/Expert Locating System based on Automatic Representation of Semantic Structure, in Proceedings of the Fourth IEEE Conference on Artificial Intelligence Applications, Computer Society of the IEEE, San Diego, CA, pp.345-349, 1988
7. L. A. Steeter and K. E. Lochbaum, Who knows: A System Based on Automatic Representation of Semantic Structure, In RIAO'88, Cambridge, MA, pp.345-349, 1988
8. S. Brin and L. Page. The anatomy of a large-scale hypertextual Web Search engine. In 7th WWW conference, Brisbane, Australia, Computer Networks and ISDN Systems, Volume 30 Issue 1-7, 1998
9. M. F. Schwartz and D. M. Wood, Discovering Shared Interests Using Graph Analysis, Communications of the ACM, vol. 36, no. 8, pp.78-89, 1993
10. A. L. Cohen, P. P. Maglio, R. Barrett, The Expertise Browser: How to Leverage Distributed Organizational Knowledge, presented at Workshop on Collaborative information Seeking at CSCW'98, Seattle, WA, 1998