Anatomy of a Collaborative Search Engine

Jingyu Sun, Xueli Yu
College of Computer and Software
Taiyuan University of Technology
Taiyuan, China
sunjingyu@tyut.edu.cn,
yuxueli@tyut.edu.cn

Ning Zhong
Dept. of Life Science and Informatics
Maebashi Institute of Technology
Maebashi, Japan
zhong@maebashi-it.ac.jp

ABSTRACT

We present ExpertRec, a collaborative/social Web search engine. With ExpertRec, users share experts' search histories (search expertise) through a Web browser toolbar or a proxy browser. As compared to a current web search engine, there are two challenges in ExpertRec: one is to supply a right teamwork environment to satisfy users' collaborative search; other is to identify search expertise through utilizing users' search histories and so on. Firstly, we implement a Mozilla Firefox toolbar (a Firefox extension), which can integrate with mainstream search engines like Google, Yahoo!, et al., to meet users' teamwork needs. And it allows users to generate high-quality tags, votes, comments over current Web including search histories, personal archival content in local host typically beyond the reach of existing Web 2.0 social tagging system. Then, a CBR (case-based reasoning)-based recommendation engine is designed to build recommendations and its core is a scalable method to identify search expertise based on a hierarchical user profile in order to improve users' search quality. In addition, a novel recommendation form is adopted through merging recommendations into return-list by a search engine with the help of ExpertRec toolbar. We describe the architecture, user interface, main techniques and algorithms of ExpertRec, and our primary evaluation is introduced.

Author Keywords

Collaborative Web search, Social search, Search Expertise, User profile, Case-based reasoning.

ACM Classification Keywords

H.5.2 Information Interfaces and Presentation: Miscellaneous—*Group and Organization Interfaces - computer-supported cooperative work.*

INTRODUCTION

As the amount of information on the Web continuously grows, search engines handles about 1 billion queries per day now! Web search has become one of the prominent information behaviors. But it is generally considered to be a solitary activity for satisfying users' individual needs and current mainstream search engines like Google, Yahoo!, Bing, Baidu, et al. are designed for solo use. However, many search tasks in both professional and casual settings can benefit from the ability to jointly search the Web with others[8], that is, collaborative Web search. It often occurs in some search tasks, i.e. travel planning, literature search, technical information and so on, and is becoming a staple way to improve search quality by users' collaboration[11].

In addition, popular Web browsers including Microsoft Internet Explorer (IE) and Mozilla Firefox, et al. are only fit for solo use. They lack some features for collaborative Web search, such as to allow users to storage their right search histories and related Web pages about their search tasks and to share them with others. Some demo collaborative Web search engines, i.e. Heystaks [9], SearchTogether [7], S^3 [6], CoSearch [1] and so on, are developed to study fundamental models and techniques of collaborative Web search.

In this paper, we introduce a collaborative/social Web search engine, named ExpertRec, which allows every user to tag, vote, and share his/her search histories and related Web pages. It integrates with mainstream search engines like Google, Yahoo!, Bing, Baidu, et al. through a Web browser toolbar or a proxy browser. Current version depends on a Mozilla Firefox toolbar to catch search histories and display recommendations when a user searches in a Firefox Web browser. A CBR-based recommendation engine is designed to build recommendations according to three recommendation rules, and its core is a scalable method to identify search expertise based on a hierarchical user profile. In addition, as a novel Web search engine, it is designed to work in cooperation with rather than competing against mainstream search engines like Google, Yahoo, Bing, Baidu, at el.

The rest of the paper is organized as follows. The next section introduces the architecture and interface of ExpertRec. Then, main techniques and core algorithms of ExpertRec are discussed in details. Finally, a evaluation is presented, and main related work is reviewed.

SYSTEM OVERVIEW

Architecture

With the development of Internet and software technology, mainstream Web browsers, i.e. Microsoft Internet Explorer

2nd International Workshop on Social Recommender Systems in conjunction with CSCW 2011, March 19-23, 2011, Hangzhou, China.

(IE) and Mozilla Firefox, et al., allow users extend their basic functions through developing a toolbar with plug-in technology or Firefox extension. For example, when someone browses a Web page in Firefox browser, a special toolbar can capture his/her click actions and extract the title, url and others of the Web page, i.e. return-list by a search engine. In addition, all data extracted can be uploaded to a recommendation engine server for processing with the help of some software development technologies like Ajax[1], Javascript[2], et al. and recommendations also be downloaded and merged with the return-list by a search engine. With the help of such a toolbar, current Web browsers and search engines are combined to support collaborative Web search in a convenient way for users.

In recent years, with the popularity of mobile Internet devices, i. e. IPad, IPhone and so on, more and more people start using mobile search. However, current mainstream mobile browsers like Safari[3], UC browser[4] and so on are limited to support teamwork environment through developing a right plug-in. Usually, we can add some special functions to a mobile Web browser in order to capture users' search histories and support mobile collaborative Web search. We call such a extended Web browser as a proxy browser and take it as another convenient way to support collaborative Web search.

Based on above ideas, we design the system architecture of ExpertRec. As shown in Figure 1, it takes the form of two basic components: a client agent and a back-end server. Users have the alternative of a proxy browser or a client side browser toolbar as a client agent. The ExpertRec proxy browser is being designed and developed on two mobile platforms including Android and Apple. The ExpertRec toolbar is implemented as a Firefox extension, which can integrate with mainstream search engines like Google, Yahoo!, Bing, Baidu, et al. The back-end server includes a content sever, a CBR-based recommendation engine and a Web portal.

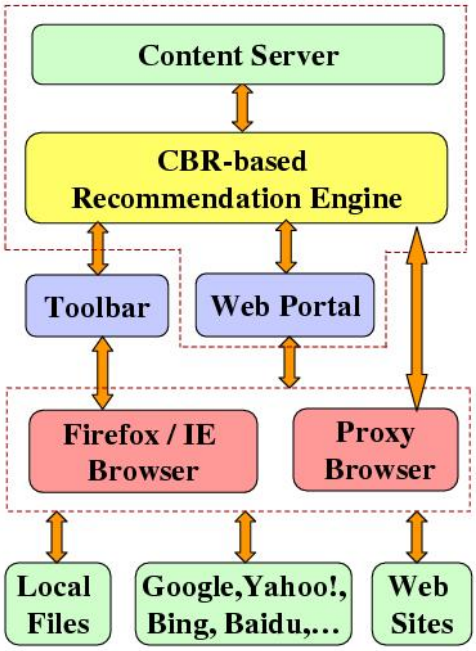

Figure 1. System Architecture of ExpertRec

Interface and Functions

(1) Toolbar

The ExpertRec toolbar can be installed in anyone Firefox Web browser and is the core of ExpertRec. Figure 2 shows its main functions for users to support collaborative Web search. They are,

- to allow users to create search histories bases (named ExpBases) and share them with others;
- to capture search results in some Web search engines, such as Google, Yahoo, Bing, Baidu and so on;
- to allow users to register, login and invite buddies;
- to provide a range of ancillary services, i.e. the ability to tag, vote and share a Web page; and
- to manage the integration of ExpertRec recommendations with the default result-list.

Particularly, the toolbar includes five parts when one logins: an ExpBase list, a group of buttons for tagging, sharing and voting Web pages, a drop-down menu about ExpBase, a drop-down menu about community and a button which is used to start or pause the integration of ExpertRec recommendations with the default result-list returned by Google, et al. Usually, every registered user can create several ExpBases for himself/herself with two types: private and public. The private ExpBase is only used to record his/her clicks in the Google result-list, tagged Web pages and so on in order to remind him in the future, but search histories in the public ExpBase can be shared with other users. As a result, some similar former search histories could be recommended through integrating with the default Google result-list when he/she queries in Google shown as Figure 3. Additionally, every user can invite his/her friends to join ExpertRec and share his/her search experiences with them.

Figure 3 is a snapshot to display recommendations based on our proposed recommendation rules. In addition, search results from default search engine are re-ranked through combining the user profile and search expertise about one topic identified by our proposed method.

(2) Back-end Server

The content sever is used to store search histories, favorites, et al. uploaded by the ExpertRec toolbar and recommendations. Especially, the former is taken as raw data to be used for organizing into search cases, then search cases are inputs of the recommendation engine and recommendations

[1]http://en.wikipedia.org/wiki/Ajax_programming
[2]http://en.wikipedia.org/wiki/JavaScript
[3]http://www.apple.com/safari/
[4]http://en.wikipedia.org/wiki/UC_Browser

Figure 2. Main functions of the toolbar when some logins.

Figure 3. Google search results with ExpertRec promotions.

are built according to our proposed method. The recommendation engine outputs recommendations according to three recommendation rules in the following.

- First rule is that a recent search case associated with the query and appeared in his/her profile will be prompted in first place of promotion-list and used to remind him/her in order to avoid to browse repeatedly.
- Second one is that a search case with a largest *expScore* value is recommended in second place of promotion-list and is taken as his/her a possible interesting new Web page.
- Third rule is that a re-ranking of search results list is provided and returns visited before are marked for reminding.

The Web portal is been designed and developed and provides several functions, such as to maintain search experiences, to add/delete friends and so on, for users and administrators. Particularly, They include following functions,

- to allows users to register, login and logout;
- to allow users to manage their interests, list of friends and analyze their social networks;
- to allow users to maintain search cases, i. e. to delete, move them and so on;
- to allow administrators to index individual pages against query/tag and positive/negative votes; and
- to allow administrators to analyze tags, votes, Web pages through some effective data mining methods, such as spectral clustering [13], peculiarity oriented mining [15] and so on.

CORE TECHNIQUES

Representation of Search Experiences

ExpertRec is a CBR system in nature and distinguishes itself from traditional CBR systems since there is a novel way to organize and represent users' search experiences (including search histories, favorites and so on).

Definition 1. A search case c is denoted by $e = \{title, queries, tags, votes, URLwords, snippet, selected{-}frequency\}$. Where, c denotes a search experience and is a summary of a Web page including its title, queries, tags, votes, URLwords, snippet and selected-frequency; $snippet$ denotes its abstract or description; $URLwords$ denotes the keywords extracted from its URL; $select - frequency$ denotes the total number of times it is selected.

Definition 2. An ExpBase E is usually related to one search task or topic and includes several users' cases and is denoted by a set $E = \{expbaseid, creatorid, name, tags, type, description\}$ and includes k search cases: $\{e_1, e_2, ..., e_k\}$ built according to some users' experiences.

Definition 3. An $ExpDB$ is a super case bases and includes m ExpBases.

As a result, ExpertRec provides a two-level case bases for managing search cases. In addition, every public ExpBase is usually created by a trusted expert, and ExpertRec allows some users to join it and share its search cases.

Search Case Retrieval

In order to retrieve a similar search case, we adopts Jaccard similarity [5]. For a query q and a search case c in a selected ExpBase E, the ranking score is computed by

$$Sim(q, c) = |q \cap c|/|q \cup c|. \quad (1)$$

Where, we choose terms in $title, queries, tags, URLwords, snippet$ of a search case. Before retrieval, we take some preprocessing steps like stop words removal and stemming. In fact, each case is treated as a list of terms after preprocessing.

Constructing a Hierarchical User Profile

In ExpertRec, any users' experiences, i.e. search histories and favorites could be organized into search cases and the data source for user profiles. Our hypothesis is that terms that frequently appear in snippets, titles, queries and tags of search cases can represent topics that interest users, and votes and selected-frequency of search cases influence degree of interests. Here we use an approach in [14] proposed by Xu et al. to build the hierarchical user profile based on frequent terms. In the hierarchy, general terms with higher frequency are placed at higher levels, and specific terms with lower frequency are placed at lower levels.

C represents the collection of all search cases in one ExpBase. $C(t)$ denotes all cases covered by term t, i.e. all cases in which t appears, and $|C(t)|$ represents the number of cases covered by t. A term t is $frequent$ if $|C(t)| \geq minsup$, where $minsup$ is a system-specified threshold, which represents the minimum number of cases in which a frequent term is required to occur. Each frequent term indicates a possible user interest. In order to organize all the frequent terms into a hierarchical structure, relationships between the frequent terms are defined below.

Assuming two terms t_1 and t_2, the two heuristic rules used in this approach are summarized as follows:

1. ***Similar terms:*** Two terms that cover the document sets with heavy overlaps might indicate the same interest. Here we use the Jaccard function [5] to calculate the similarity between two terms: $Sim(t_1, t_2) = |C(t_1) \cap C(t_2)|/|C(t_1) \cup C(t_2)|$. If $Sim(t_1, t_2) > \delta$, where δ is another user-specified threshold, we take t_1 and t_2 as similar terms representing the same interest.
2. ***Parent-Child terms:*** Specific terms often appear together with general terms, but the reverse is not true. For example, "Semantic Web" tends to occur together with "Web", but "Web" might occur with "Web intelligence" or "Web 2.0", not necessarily "Semantic Web". Comparatively speaking, "Web" is a general term, but "Semantic Web" is a specific term. Thus, t_2 is taken as a child term of t_1 if the condition probability $P(t_1|t_2) > \delta$, where δ is the same threshold in Rule 1.

Rule 1 combines similar terms on the same interest and Rule 2 describes the parent-child relationship between terms. Since $Sim(t_1, t_2) \leq (t_1|t_2)$, Rule 1 has to be enforced earlier than Rule 2 to prevent similar terms to be misclassified as parent-child relationship. For a term t_1, any case covered by t_1 is viewed as a natural evidence of users' interests on t_1. In addition, cases covered by term t_2 that either represents the same interest as t_1 or a child interest of t_1 can also be regarded as supporting cases of t_1. Hence supporting cases on term t_1, denoted as $S(t_1)$, are defined as the union of $C(t_1)$ and all $C(t_2)$, where either $Sim(t_1, t_2) > \delta$ or $P(t_1|t_2) > \delta$ is satisfied.

Based on the above rules, a hierarchical user profile can be automatically built in a top-down fashion. The profile is represented by a tree structure, where each node is labelled a

term t, and associated with a set of supporting documents $S(t)$, except that the root node is created without a label and attached with a user's name of C, which represent all personal cases. Starting from the root, nodes are recursively split until no frequent terms exist on any leave nodes. Two algorithms are used to build it shown as **Algorithm 1** and **Algorithm 2**.

Algorithm 1 :$BuildUP(n, C, minsup, \delta)$

Input: a node n, supporting cases C, thresholds $minsup$ and δ.
Output: A user profile U.
Steps:
1. $Split(n, C, minsup, \delta)$;
2. for each child c_i labelled t_i of node n do
3. $BuildUP(c_i, S(t_i), minsup, \delta)$;
4. return U with a tree structure.

Algorithm 2 :$Split(n, S(t), minsup, \delta)$

Input: a node n labelled term t, supporting cases $S(t)$, thresholds $minsup$ and δ.
Steps:
1. generate the frequent term list$\{t_i\}$ with $|C(t_i)| \geq minsup$
sorted by the descending order of frequency.
2. for each term t_i do
3. if $Sim(t_i, t_k) > \delta$, where $k < i$,
4. set the node label as t_i/t_k, and $S(t_i/t_k)$=$S(t_i) \cup C(t_k)$;
5. else if $P(t_i|t_k) > \delta$, where $k < i$,
6. keep the node label as t_k, and $S(t_k) = S(t_k) \cup C(t_i)$;
7. else
8. create a new node with label t_i, and $S(t_i) = C(t_i)$;
9. calculate $Sup(t_i)$ for each node with label t_i, and sorted them in a descending order.

Choosing Expertise

With the hierarchical user profile constructed above, every term with supporting search cases can be detected. In the following discussion, "topic" and "term" are indistinguishable in the context of the user profile. The support of an topic of a term t is $Sup(t)$, and $S(t)$ represents all the supporting cases for term t. $\sum Sup(t) = |C|$ is for all terms t on the leave node, where $|C|$ represents the total number of supports received from a user's search cases. In addition, our hypothesis is that a term t with lager $sup(t)$ represents a user's familiar topic and partial search cases in $S(t)$ are his/her valuable experiences.

The hierarchical user profile constructed is taken as an indicator of the user's possible familiar topics. According to probability theories, the possibility of a term can be calculated as $P(t) = Sup(t)/|C|$. With the context of information theory, the amount of information about a certain topic of the user is measured by its *self-information* [4]. For any term t,

$$I(t) = log(1/P(t)) = log(|C|/Sup(t)). \quad (2)$$

This measure has also been called *surprisal* by Myron Tribus [12], as it represents the degree to which people are surprised to see a result. More specifically, the smaller $Sup(t)$ is, the larger the self-information associated with the term t is, and the search case including term t is more valuable as it is a special search case for a user. This leads to two parameters for specifying the requirement of recommendation.

minFamiliar. The user profile above is organized from high-level to low-level. Terms associated with each node become increasingly specific as the list progresses, and same level terms are sorted from left to right in descending order of their supports. A threshold of $minFamiliar$ is defined to measure users familiar topics on both vertical and horizontal dimensions. With a specified $minFamiliar$, any term t in the user profile with $P(t) = Sup(t)/|C| \geq minFamiliar$, will be taken as a user's familiar topic.

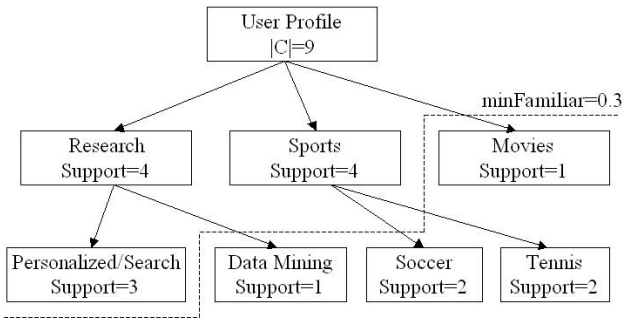

Figure 4. A hierarchical user profile

Figure 4 ia an example of the hierarchical user profile. Firstly, the possibility of every topic is calculated, for example, P("Sports")=Sup("Sports")$/|C| = 4/9 > 0.3$. When $minFamilar = 0.3$, topics above broken line are taken as unfamiliar topics. The complete user profile is denoted as U, and $U[Fam]$ represents the familiar part of U, that is, the part above $minFamiliar$. Since the support for terms decreases monotonically travelling horizontally and vertically, the $U[Fam]$ will be a connected subtree of the complete user profile stemming from the user profile root. With the threshold $minFamiliar$, the system will know exactly which topics a user is familiar with.

For conventional, $U[Fam]$ is transformed into a list of weighted terms and the weight of each term in $U[Fam]$ is estimated by applying the concept of IDF (Inverse Document Frequency). Given a term t, the weight of t, denoted by w_t, is calculated as:

$$w_t = log(|C|/Sup(t)), \quad (3)$$

where $|C|$ represents the total number of search cases of $U[Fam]$, and $Sup(t)$ is the support of this term on the node in E. The user profile is expressed by a list $< t, w_t >$, where t is a term in $U[Fam]$ and w_t is the weight. For example. The list is $< research, 0.301 >, < sports, 0.456 >, < personalized/search, 0.523 >$.

In order to choose expertise, we construct a clustered user profile with the same approach, named U_{expert}, through utilizing all cases of an ExpBase in ExpertRec. However, since

different users maybe visit same Web pages, $votes$ and $Selected-Frequency$ of a search case indicate degree of users' interest on it. So the number of supporting cases for a term t is adjusted as:

$$Sup(t) = \sum_{c \in S(t)} (\beta \cdot h(c) + \gamma \cdot v(c)). \quad (4)$$

Where $h(t)$ represents $Selected - Frequency$ of the search case c , $v(c)$ represents the score of the search case c and is defined as the minute of the number of supporters and non-supporters in ExpertRec, β and γ are two weight parameters and calculated as:

$$\beta = \eta \cdot \frac{|C|}{\sum_{c \in C} h(c)}, \gamma = (1 - \eta) \cdot \frac{|C|}{\sum_{c \in C} v(c)}, \quad (5)$$

where the parameter $\eta \in [0, 1]$ and usually is 0.8.

expScore When a user inputs queries, a set of terms, in a search engines when the toolbar is running, it would capture and upload the terms to a recommendation server. The server would combine them and his/her profile to find valuable expertise through travelling U_{expert}. But which expertise are most valuable for him/her? In our opinions, search cases, which don't appear in his/her profile but include terms with larger self-information according to U_{expert}, are more valuable. So we firstly choose such search cases, which support topics of terms appeared in queries, in U_{expert}. Then a score, named $expScore$, for every search case, is computed by $\sum_t I(t)$, where t denotes a term which appears in the queries and the search case at the same time. A search case with larger $expScore$ is recommended preferentially.

Re-ranking Steps

In ExpertRec, re-ranking list is built when a query is submitted to the recommendation server in five steps:

1. The expert profile of every ExpBase is built and represented by a set of $< t, w_t >$ pairs in the recommendation engine server.

2. The toolbar captures a query and the search results returned by a search engine and they are uploaded to the recommendation engine server. Each result comprises of a set of links related to the query, where each link is given a rank from the search engine, called DefaultRank.

3. For each of the returned link l, a score called $EScore$ is calculated by the expert profile as follows:

$$EScore(l) = \sum_t w_t \times f_t, \quad (6)$$

   where t is any term in the expert profile, and f_t is the frequency of the term t in the snippet of the link l. An $ERank$ is assigned to each link according to its $EScore$, and the link with the highest $EScore$ will be ranked first.

4. Re-ranking results by combining ranks from both DefaultRank and ERank. The final rank, EERank (Expertise Enhancing Rank), is calculated as:

$$EERank = \alpha * ERank + (1 - \alpha) * DefaultRank, \quad (7)$$

   where the parameter $\alpha \in [0, 1]$ indicates the weight assigned to the rank from the expert profile. If $\alpha = 0$, the expert profile is ignored, and the final rank is decided by the expert profile instead of the search engine when $\alpha = 1$.

5. The toolbar downloads the final ranking of the search results and recommends them to the user.

In addition, the recommendation engine is a configurable platform and all programs were implemented in Java. In future work, some other methods, such as expert finding [2, 3], would be implemented.

EVALUATION

In this paper, our main aim is to describe features and main technologies of ExpertRec. All experiments are conducted with following questions:

- As a CBR-based Web search recommender system, ExpertRec is interesting and easy to use because it allows users to create multiple case bases (ExpBases) and share search histories with others. But do users actually take the time to create ExpBases and do they share them with others?
- As a search assistant, ExpertRec is interesting because it promises to improve Web search by facilitating collaboration among searchers. But do users benefit from this collaboration? Do they respond positively to ExpertRec recommendation? Do they benefit from their own search experiences or those of others or a mixture of the two?
- As a special expert system, ExpertRec is interesting because it allows common users to share search experiences of experts. But How about the relationship between search quality and expertise identified by our proposed method?

In particular, we invite 20 participants who are chosen from different research groups in our labs to use and evaluate ExpertRec during the period October 2009-October 2010. They are with high levels of computer literacy and familiarity with Web search.

During the period of testing ExpertRec, about 30% participants create or join more than 10 ExpBases and about 30% participants only join ExpBases created by others. Furthermore, most of ExpBases store more than 40 search experiences. In total more 200 ExpBase were created and more than 8000 search experiences were produced. As a result, most of users are willing to utilize sharing features. In detail, We carry out an investigation about attitude for ExpertRec. 18 participants can accept the system to capture search histories, 16 like most of the function provided by ExpertRec Toolbar and have a good trial feeling. They are interested in some functions of toolbar, i.e. the vote (18 like), tagging (16 like), sharing by Email (15 like), and so on.

Through the creation and sharing of ExpBase, participants ought to find useful recommendations. According to access logs about clicked recommendations, we construct a user social network to show the relationship between recommendation producer and consumer who re-selects it. We

analyze this network and find that re-selected recommendations come from about 80% users. We can conclude that ExpertRec can help users find their wants through recommending search experiences.

Furthermore, we conducted an experiment to explore the relationship between search quality and expertise identified by our proposed method and results shown in [11]. In our future work, we will explore another method to identify expertise through computing the out-degree for every user in the user social network and users with higher out-degree taken as expertise.

RELATED WORK

To our knowledge, Heystaks[5] is only one similar system to ExpertRec introduced in [9]. In order to capture search experiences, ExpertRec implements a Firefox toolbar with similar features with Heystaks toolbar, but ExpertRec toolbar adds a few new features, i.e. search ExpBase, re-ranking recommendation. However, the main difference between ExpertRec and Heystaks is that ExpertRec adopts a novel method utilizing expertise. Furthermore, ExpertRec is been designed for mobile platform including Android and Apple.

CONCLUSIONS

Collaborative Web search is a promising way to improve search quality by users' working in cooperation. However, this approach requires a convenient way for users to work together. For this goal, we designed ExpertRec, a novel recommender system. It can utilize search expertise and integrates with mainstream search engine like Google via a browser toolbar. The toolbar allows users to tag, share and vote Web pages (including search histories and local Web pages). Search expertise is identified by analyzing a hierarchical user profile and recommended according three recommendation rules. Primary evaluation shows that ExpertRec provides some functions users like.

In our future work, we would extend it to support mobile search through developing a proxy browser in Android and Apple platform. In addition, we would extend CBR-based recommendation server through using a new case base maintenance approach for Web-scale CBR [10], and some new outlier detection algorithms [15] may be used to mine a possible interesting search case for users. Furthermore, evaluation will be thoroughly discussed.

Acknowledgment

The authors would like to acknowledge the following support for their research on ExpertRec: Youth Natural Science Foundation of Shanxi (No. 200821024); National Natural Science Foundation of China (No. 60873139). And special thanks for Xianhua Li and PhD Jian Yang.

REFERENCES

1. S. Amershi and M. Morris. Cosearch: A system for co-located collaborative web search. *CHI 2008*, pages 1647–1656, 2008.
2. K. Balog, L. Azzopardi, and M. de Rijke. Formal models for expert finding in enterprise corpora. *In Proceedings of SIGIR*, pages 43–55, 2006.
3. D. W. Chen, J. Tang, and et al. Discovering the staring people from social networks. *In Proc. of WWW2009*, pages 1219–1220, 2009.
4. T. M. Cover and J. A. Thomas. Elements of information theory, 1st edition. *Wiley-InterScience, New York, NY*, 1991.
5. D. Eduardo. On clustering and evaluation of narrow domain short-text corpora. *PhD thesis*, 2008.
6. M. Morris and E. Horvitz. S3: Storable, shareable search. *Interact 2007*, pages 120–123, 2007.
7. M. Morris and E. Horvitz. Searchtogether: An interface for collaborative web search. *IUIST 2007*, pages 3–12, 2007.
8. M. R. Morris. A survey of collaborative web search practices. *Proceeding of the twenty-sixth annual SIGCHI conference on Human factors in computing systems*, pages 1657–1660, 2008.
9. B. Smyth and P. Champin. The experience web: A case-based reasoning perspective. *In Grand Challenges for reasoning from experiences, Workshop at IJCAI'09*, pages 566–573, 2009.
10. J. Y. Sun, X. L. Yu, and et al. A model for personalized web-scale case base maintenance. *AMT2009*, pages 442–453, 2009.
11. J. Y. Sun, X. L. Yu, and N. Zhong. Collaborative web search utilizing experts' experiences. *Web Intelligence 2010*, pages 120–127, 2010.
12. M. Tribus. Thermostatics and thermodynamics. *D. Van Nostrand, New York, NY*, 1961.
13. U. von Luxburg. A tutorial on spectral clustering. *Statistics and Computing*, 17:395–416, 2007.
14. Y. Xu, B. Zhang, Z. Chen, and K. Wang. Privacy-enhancing personalized web search. *In Proceedings of WWW*, pages 591–600, 2007.
15. J. Yang, N. Zhong, Y. Y. Yao, and J. Wang. Local peculiarity factor and its application in outlier detection. *KDD 2008*, pages 776–784, 2008.

[5]http://www.heystaks.com