A Personalized Metasearch Engine Based on Multi-Agent System

Meijia Wang, Qingshan Li, Yishuai Lin, Yingjian Li, and Boyu Zhou
Software Engineering Institute, Xidian University, China

Abstract: *By providing unified access to multiple underlying search engines, metasearch engine is an intuitiveway to increase the coverage of the WWW. Great progress has been made in this area, butthe previous studies ignore the perspectives of users. This paper proposes a personalization mechanism for metasearch engine based on multi-agent system to improve precision ratio. The proposed mechanism obtains user interests from click-through data, schedules the appropriate underlying search engines according to the expertness model, and merges results based on user interest distribution. Moreover, it also has the ability to provide personalized result recommendation. Compared with the baseline results, experimental results show that the proposed personalization mechanism performs better on precision. The proposed metasearch engine is feasible for providing useful search results more effectively.*

Keywords: *Metasearch engine, multi-agent system, personalized search.*

Received October 29, 2016; accepted August 26, 2018

1. Introduction

In big data age, search engine is a tool to help users find useful information on World Wide Web. However, it has faced with problem of low recall ratio [15]. The reasons are as follows:

1. An individual search engine only indexes a small coverage of web documents [10].
2. The overlapping among different search engines is very low.

Therefore, metasearch engine which integrates the search results from multiple search engines has been proposed to extend the information retrieval coverage and improve recall ratio [13]. However it is still limited by the precision. Because most of the proposed metasearch engines use the similarity between the query and web documents to search, ignoring the perspectives of users. To improve precision, personalization mechanism for metasearch engine which takes user preferences into consideration is significant. Generating schedule strategy, merging results and providing recommendation based on user interests is very helpful to meet user's requirements.

An intelligent agent is a piece of software that is reactive, proactive, perceptible and social. It has ability to observe and act upon an environment [17]. A multi-agent system is a system composed of multiple interacting intelligent agents [1]. Using multi-agent architecture to implement personalization mechanism for metasearch engine has notable advantages. User interests change overtime, agent has the ability to perceive the change of user interests actively and update in time. Moreover, it is also significant to merge results and provide recommendation more flexibly according to search context. In summary, agent improves the adaptability of metasearch engine.

In this paper, multi-agent based architecture is proposed to improve personalization mechanism for metasearch engine. Intelligent agent is utilized to analyze user interests, generate schedule strategy, merge results and provide recommendation that user might be interested in. Precision@N [3], DCG@N [20] and MAP@N [19] are used to evaluate the performance of the proposed metasearch engine.

The rest of this paper is organized as follows: In section 2, related work is reviewed and compared. In section 3, the architecture of the proposed agent-based personalization mechanism for meatsearch engine is introduced. Section 4 demonstrates the user interest model, including the obtainment and storage of user interest. The schedule strategy is described in section 5. Section 6 introduces our result merging method which considers user interest distribution. The result recommendation is described in section 7. Experimental results will be presented in section 8. Section 8 briefly illustrates the prototype of the proposed system named IM Search as well. Finally, the conclusions and further work are given in section 9.

2. Related Work

A large number of metasearch engines have been proposed, such as Profusion [7], Saavy search [9], Web Fusion [11, 13]. These metasearch engines can be grossly divided into two categories, Component-based architecture and Agent-based architecture. In this section, an overview of metasearch engines is investigated.

1. Component-based architecture

Helios [8] is an open source metasearch engine which consists of six components. The web interface is designed to allow users to submit queries and select the desired search engines. The local query parser and Emitter dispatches query into the appropriate format for the chosen underlying search engines. The engines builder maintains all the settings necessary to communicate with the underlying search engines. The HTTP Retrievers handle the network communications. The search results collector and parser is utilized to collect results, and return them in using XML. Merger and Ranker ranks all of results into a single list. Savvysearch [9] is a metasearch engine that learns to identify which underlying search engines are most appropriate for different queries, reasons about resource demands, and represents an iterative parallel search strategy as a simple plan. Finally, the returned results will be displayed to users. ProFusion [7] consists of five components. The user interface is utilized to interact with users, receiving user's request and providing several available options. The Duplicate Removal is designed to remove duplicated pages, using a few simple rules. The merge algorithms are responsible for merging results. The intelligent search Engine Selection allows metasearch to automatically select the best three search engines for a given query, based on the query's domain. The search result presentation is responsible for displaying the final results to users.

Component-based metasearch engines adopt the idea of object-oriented software engineering, encapsulating each function into function module. Each module interacts with others through messages to complete all functions of metasearch engine, leading to the characteristics of "limited autonomy, fixed encapsulation and interactive monotonicity". Therefore component-based meta-search engines are lack of intelligence, dynamics and adaptability.

2. Agent-based architecture

In order to improve the intelligence and adaptability of Profusion, Fan and Gauch [6] proposed a multi-agent architecture for ProFusion. The architecture consists of four kinds of agents. The dispatch agent is responsible for communicating with users and dispatching queries to the search agent and the learning agent. The search agent interacts with the underlying search engines, reporting search results, confidence factors, and time-out values of the underlying search engines to the dispatch agent. The learning agent is in charge of the learning and development of the underlying search engines. The guarding agent is invoked when a search engine is down and it is responsible for preventing the queries to non-responsive search engines and detecting when the search engine is back online. WebFusion [11, 12, 13] is composed of two layers of different types of agents. Each agent in the first layer is used to communicate with a specific underlying search engine. The master agent in the second layer is responsible for collecting results which returned by agents in the first layer. Arzanian *et al*. [2] proposed a multi-agent based personalized metasearch engine using automatic fuzzy concept networks. The metasearch engine consists of user agent, search agents group and personalization agents group. The user agent is responsible for communicating with users, predefining concepts vector and user's profile. The search agents group is composed of four agents, Google agent, Yahoo agent, Ask agent and Msn agent. Each of them is in charge of communicating with a specific underlying search engine. The personalization agents group is composed of FCN1 agent, FCN2 agent and ranking agent. The FCN1 agent is utilized to generate fuzzy concept network for a user's profile. The FCN2 agent merges all of results and generates a fuzzy concept network for results list. The Ranking agent receives the concept network which was sent by FCN2 agent and completes the ranking process.

Because of the characteristics of agent, such as autonomy, sociality, reactivity and proactiveness [14], the Agent-based architecture performs better than the Component-based architecture. Some metasearch engines based on multi-agent system have been proposed, however the previous research is still short of personalization level. In this paper, a multi-agent based personalization mechanism for metasearch engine is proposed, mining user interests and providing the useful search results more effectively.

3. Proposed Multi-Agent Architecture

The architecture of the proposed personalization mechanism for metasearch engine is composed of seven roles played by seven related agents.

- *Interface Role*: this role is played by the InterfaceAgent, which will be activated when the Interface Agent is launched. Interface Role interacts with users. It has ability to get user id and query words, and display all of results to the current user.
- *UserInterest Role*: this role is played by the UserInterest Agent, it has ability to obtain, update and read user interests.
- *UserGroup Role*: this role is played by the UserGroup Agent. The required abilities are as follows:
  1. According to user personal information and query words, cluster users into different groups from two dimensions, and update user group information.
  2. Read group information to which user belongs.
- *ResultRecommendation Role*: this role is played by the ResultRecommendation Agent, which will be activated when the Result Recommendation Agent is launched. It has ability to read click through data

of group members for the current user, and generate recommended results.

- *Search Role*: this role is played by the Search Agent, which will be activated when the Search Agent is launched. The required abilities are as follows:
  1. Calculate expertness model of underlying search engines.
  2. Record the states of underlying search engines.
  3. Analyze query words of the current user.
  4. Generate schedule strategy according to expertness model.
- *SE Role*: this role is played by the SE Agent, which will be activated when the Search Agent is launched. It has ability to monitor the states of underlying search engines, analyze query words of the current user, and generate schedule strategy according to expertness model.
- *ResultMerge Role*: this role is played by the ResultMerge Agent, which will be activated when the Result Merge Agent is launched. It has ability to remove all duplicate results returned by underlying search engines and rerank all of search results into a single list.

The multi-agent based architecture works as shown in Figure 1: a user requests a query to the Interface Agent. Then Interface Agent sends the query to the Search Agent, and the ResultRecommendation Agent. The Search Agent receives the query, generates the schedule strategy and sends the query to the SE Agent which will be scheduled. After getting the query, the SE Agents communicate with underlying search engines to complete the search task, and then pass on the returned results to the ResultMerge Agent. Meanwhile, the Interface Agent also passes on user Id to the UserGroup Agent and user interest agent. The UserGroup Agent gets the information of groups to which the current user belongs, and sends it to the ResultRecommendation Agent. After getting the query and user group information, the ResultRecommendation Agent generates the recommended results and sends them to the Interface Agent. According to user Id, the UserInterest Agent obtains user interest factor and passes it on to the ResultMerge Agent. Then the ResultMerge Agent merges all of results returned by the SE Agent into a single list, and returns the list to the InterfaceAgent for displaying. The UserInterest Agent analyzes the click-through data to obtain user interests. After obtaining all of results, the Interface Agent is responsible for displaying these results to users.

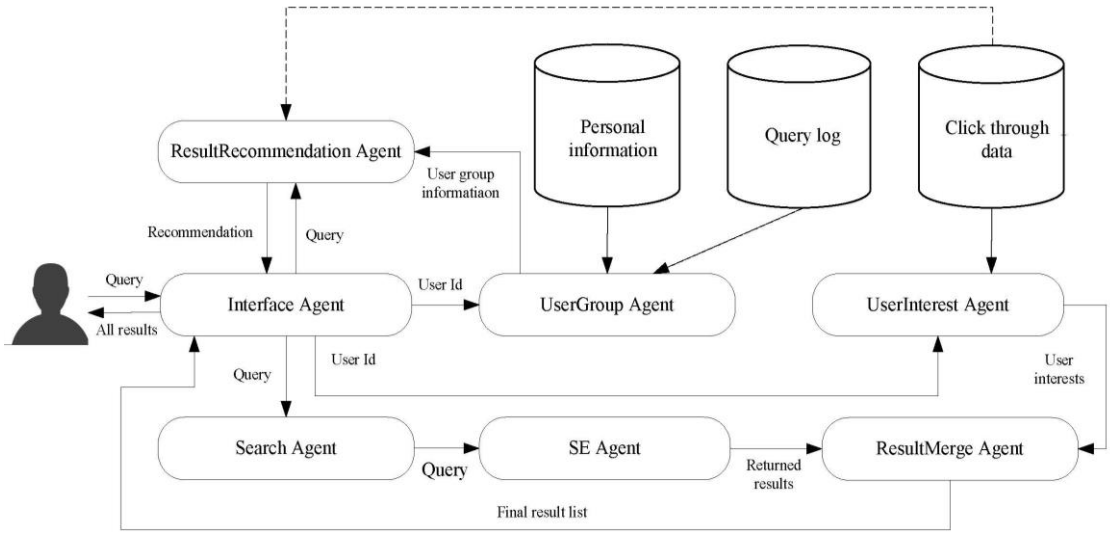

Figure 1. Working diagram of the proposed multi-agent architecture.

4. User Interest Model

It has been proved that a user will click the interested web pages in most cases. For this reason, user interest is obtained based on click through data. For each record of a user, word segmentation technology is used to extract the useful information and get rid of stop words (and, or, a, et al.). Take the words with high weight value as the user interest words (In this paper, words with high weight value means words appeared with high frequency). In accordance with the Sogou corpus for text classification, all documents could be divided into these nine topics: literature, finance, employment, sports, tourism, education, IT, health, and military. The naive Bayesian formula is utilized to decide the topic to which user's interest words belong, as shown in Equation (1).

$$C_f = argMax(P(c_j) \times \prod_1^c P(x_i|c_j)) \quad (1)$$

Where C_f is the final topic, $P(c_j)$ is the prior probability of the topic c_j, $P(x_i|c_j)$ denotes the class conditional probability of characteristic quantities x_i which belongs to c_j, and $\prod_1^c P(x_i|c_j)$ calculates the

class conditional probability of characteristic quantities x_i in all topics.

Furthermore, the topic frequency corresponding to user's interest words is used to represent user interest factor. User interest model is represented as user interest tree which consists of three layers, as shown in Figure 2. The first layer is user's ID which identifies a user, the second layer is the topics which user interested in, and the third is the user interest words with weight values.

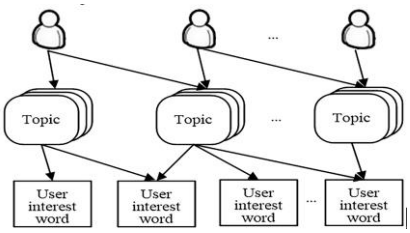

Figure 2. User interest model.

It is proved that user interest decays with time which is similar to human memory. Therefore, the forgetting factor is utilized to ensure the interest words which user focused on at present are of highest weight, as shown in Equation (2):

$$F(k) = e^{-\frac{log_2(\Delta t)}{f}} \tag{2}$$

Where f is the half-life indicating that after f days, user interest is forgotten half. Δt represents the time period between latest updating of interest word k to the current time.

The weight value$\omega(k)$of user interest word k will be updated by Equation (3):

$$\omega(k) = \omega(k) * F(k) \tag{3}$$

5. Schedule Strategy

In order to evaluate the ability of underlying search engines, the expertness model is constructed, as shown in Table 1, here n=9, it represents the nine topics, including literature, finance, employment, sports, tourism, education, IT, health, and military. m is the number of underlying search engines. e_{ij} indicates the expertness of underlying search engine i about topic j, calculated from two aspects: concept lattice and click-through data. The formula is shown in Equation (4).

$$e_{ij} = \sum_{i=1}^{m} C_{ij} \times \frac{U_{ij}}{\sum_{i=1}^{m} U_{ij}} + C_{ij} \tag{4}$$

Where e_{ij} is theexpertness of the underlying search engine i about topic j. C_{ij} is the expertness of underlying search engine i about topic j which is calculated based on concept lattice. While U_{ij} is the expertness calculated based onclick-through data.

Table 1. The expertness model of underlying search engines.

	Topic 1	**Topic 2**	**…**	**Topic n**
SE_1	e_{11}	e_{12}	…	e_{1n}
SE_2	e_{21}	e_{22}	…	e_{2n}
…	…	…	…	…
SE_m	e_{m1}	e_{m2}	…	e_{mn}

For a query, the documents returned by most of underlying search engines are more important than others. Therefore, the expertness of an underlying search engine has the characteristic as follows: The top ranked documents retrieved by these search engines are the same, or quite similar. Based on concept lattice [16], the degree that search engines support each other is calculated. The data structure of a search result consists of URL, title and abstract. Take the URL set of all search results as the object set of formal context. Word segmentation is conducted for all the titles and abstracts of the results, then take the keywords as the attribute set of formal context. The binary relation between the object set and the attribute set is defined as whether the search result corresponding to the URL in object set contained the keywords in attribute set. The method proposed by Du *et al*. [4] is used to construct concept lattice. If there are two concept nodes in a concept lattice, concept $A=(URL_A, Keyword_A)$ and concept $B=(URL_B, Keyword_B)$, the similarity between $Keyword_A$ and $Keyword_B$ is calculated by Equation (5):

$$Sim(Keyword_A, Keyword_B) = \frac{1}{n \times m} \sum_{i=1}^{n} \sum_{j=1}^{m} sim(Keyword_{A_i}, Keyword_{B_i}) \tag{5}$$

Where n is the number of keywords of node A, while m is the number of keywords of node B.

The similarity between concept A and B is calculated by Equation (6).

$$Node_{sim(A,B)} = \frac{|URL_A \cap URL_B|}{max(|URL_A|, |URL_B|)} \times (1-\omega) + sim(Keyword_A, Keyword_B) \times \omega \tag{6}$$

Where ω is the weighting coefficient, this paper set ω ro 0.8.

The similarity between concept lattice A and concept lattice B is calculated by Equation (7), that is the degree which search engines support each other.

$$Concept_{sim(A,B)} = Node\_sim(max(A), max(B)) + Node\_sim(min(A), min(B)) + \sum_{i=1}^{L-2} Node\_sim(max_i(A), max_i(B)) \tag{7}$$

Where *max(A)* and *max(B)* is the biggest node of concept lattice A and concept lattice B respectively, while *min(A)* and *min(B)* are the smallest nodes.

Finally, from Equation (8), the expertness score of search engine i about topic j is obtained based on concept lattice. *SE_set* is the set of search engines.

$$C_{ij} = \sum_{a \in SE\_set - \{i\}} Concept\_sim(i, a) \tag{8}$$

For each topic, several keywords are selected and passed on to the search engines, constructing concept lattices according to the returned results. Then the expertness of each underlying search engine about a specific topic will be obtained.

Click-through data [5] reflects the users' satisfaction of search engine. Therefore the click-through data is also used to obtain the expertness of underlying search engines. For each record of click-through data, find the topic to which belongs.

According to Equation (9), for record *a*, the expertness score of the search engine is calculated.

$$W_a = 0.01 \times max\{(100 - rank), 0\} / click \quad (9)$$

Where *click* represents the click rank of this record, it is inversely proportional to W_a. *rank* is the rank of the URL of record *a* in all returned results. If the *rank* is bigger than 100, W_a will be set to 0. *100-rank* is utilized because URLs ranked at the bottom are mostly invisible to users.

If there are *m* records belonging to topic *j* in the click-through data of a search engine *i*, the expertness score of the search engines *i* about topic *j* is calculated by Equation (10):

$$U_{ij} = \frac{1}{m}\sum_{a=1}^{m} W_a \quad (10)$$

Finally, based on Equation (4), the expertness model of each underlying search engine is obtained.

When user submits a query, metasearch engine obtains the topic to which the query belongs, then according to the expertness model, the appropriate underlying search engines are selected to complete the search task.

6. Result Merging

Most search engines generate the same set of results for an identical query from different users and display in the same way. However, different users focus on different topics. For instance, two users issue the same query, "apple." The user who is interested in mobile phone prefers the results about iPhone. The other users may prefer the results about fruit. In order to obtain the personalized result list, it is necessary to organize the same results with different order for each user. Therefore, taking user interest into consideration is important when ranking the documents returned by underlying search engines. For the registered users, it is easy to get their interests. Based on the above idea, a personalized result merging method is proposed.

For a user, the final score assigned to document *d* is calculated by Equation (11):

$$rank(d) = \begin{cases} \frac{\frac{\sum_{i=1}^{m} r_i}{mn\left(\frac{k}{10}+1\right)^n}(1-c\frac{n}{m})}{tan(inf_d)}, & \sigma > T \\ \frac{\sum_{i=1}^{m} r_i}{mn\left(\frac{k}{10}+1\right)^n}\left(1-c\frac{n}{m}\right)(1-inf_d), & \sigma < T \end{cases} \quad (11)$$

Where inf_d is user interest factor for document d, calculated by the cosine similarity between the current user interests and the topics of *d*. Moreover, if inf_d equals to 0, it will be set to infinitesimal value. *m* is the number of underlying search engines employed by meta search engine, *n* is the number of underlying search engines who returned *d*. *c* is the weighting coefficient, according to experiment, *c* is set to 0.4, $(1 - c\frac{n}{m})$ is intended to improve the scores of documents which seldom appeared but are important to the user. *k* is the number of documents that each underlying search engine returned. *ri* is the ranking position of d in the search engine *i*. σ represents the stand deviation of user interests, calculated by Equation (12). *T* is the threshold, if $\sigma<T$, it indicates that user interests are evenly distributed on different topics, therefore, user interests make a less impact on document *d*, linear function is used to calculate the score of *d*. While if $\sigma>T$ if, it means that user interests are significant for the document, tangent function is used.

$$\sigma = \sqrt{\frac{\sum_{i=1}^{N}(c_i - \bar{c})^2}{N}} \quad (12)$$

Where *N* is the number of topics, c_i is the interest factor about topic *i*, $\bar{c}$ is the mean value of user interest factor about all topics.

In the proposed method, the document which obtains the lowest score will be best ranked.

7. Result Recommendation

Providing result recommendation will help users to find useful information with less effort. Sharing search experience between similar users is an effective way to generate recommendations. Therefore, the proposed personalized mechanism clusters users into different groups from two dimensions: personal information and query data. The personal information is explicit data, which obtains from the registration of a user. When a user registers in the metasearch engine, he/she will be asked to providing personal information, including gender, address, and occupation. The query log is implicit data, users also can be clustered based on query words.

First, the user model is constructed, as shown in Equation (13):

$$U = < Q, C, P > \quad (13)$$

Where *Q* represents the query data that user passed on to metasearch engine. *C* represents the returned documents which user clicked for a certain query, came from click-through data. *P* is the personal information that user submitted to the system.

For personal information, it is easy to classify users into different groups, users with the same gender, address and occupation are in the same group. While for query data, take the log-likelihood rating as the similar function, and the query similarity between users will be obtained. Furthermore, Density-Based Spatial Clustering of Applications with Noise (DBCSAN) [18] is used to cluster users based on query similarity. Above all, all of users will be clustered from their personal information and query data respectively.

In most cases, users only click relevant documents. Moreover, if the i^{th} document meets user's requirement, he/she will never browse the next document. Therefore, from the click-through data, the relevant documents for a query will be found. In order to get a better document recommendation, ranking all of the

relevant documents is necessary. For user u_c, the score of recommended document r which has been clicked by user u_b is calculated by Equation (14):

$$score(r) = \omega_p \cdot sim_p(\mathrm{u}_c, \mathrm{u}_b) + \omega_q \cdot sim_q(\mathrm{u}_c, \mathrm{u}_b) + \omega_i \cdot sim_i(\mathrm{u}_c, \mathrm{u}_b) \quad (14)$$

Where ω_p , ω_q and ω_i are the weight coefficients, $sim_p(\mathrm{u}_c, \mathrm{u}_b)$ is the personality similarity between u_c and u_b, calculated by Equation (15). $sim_q(\mathrm{u}_c, \mathrm{u}_b)$ is the query similarity between u_c and u_b, which has been mentioned before, and $sim_i(\mathrm{u}_c, \mathrm{u}_b)$ is the interest similarity between u_c and u_b, calculated by cosine similarity.

$$sim_p(u_i, u_j) = \frac{d_{u_i} \cdot d_{u_j}}{\|d_{u_i}\| + \|d_{u_j}\| - d_{u_i} \cdot d_{u_j}} \quad (15)$$

Where d_{ui} and d_{uj} represents the personal information vector of $user_i$ and $user_j$ respectively.

In a conclusion, when user submits a query to metasearch engine, the group to which he belongs will be obtained. Take the documents clicked by the group members who have requested the same query before as the document recommendation. Then user similarity is used to rank all of recommended documents. That is, a user who is more similar to the current user, his/her clicked documents will be best ranked.

8. Experimental Results

In this section, the performance of the proposed personalization mechanism is discussed. Based on the above architecture and methods, we implements a WWW metasearch engine called "IM search", which combines the search engines "Youdao", "Baidu", "Bing", "Yahoo", and "Sogou". The homepage is shown in Figure 3.

Figure 3. Homepage of IM search.

By using P@N and DCG@N metrics, the performance of "IM search" is compared with the five employed underlying search engines, and the two metasearch engines, SvvySearch and VROOSH.Two users are designed to log in IM Metasearch, User1 and User 2. User 1 is interested in sports, finance, education and tourism. User 2 is interested in education, sports and military. For the same query "lincoln", the results pages for user1 and user2 are shown in Figures 4 and 5 respectively. It is obvious that these two users have different retrieval results. Because a user who is in the same group with user1 submitted the query "lincoln" and clicked some relevant documents before, the result list for user1 has the recommended result that user1 might be interested in.

Figure 4. Returned pages for user 1.

Figure 5. Returned pages for user 2.

We take search engine baidu, bing, youdao, sogou, yahoo and meta search engines Savvy Search and VROOSH as the baselines to evaluate the effectiveness of IM Search. 50 sample queries are selected, including 7 navigational queries, 30 informational queries and 13 transactional queries. Some of these queries are listed in Table 2. Six students in our laboratory are invited to register in "IM search", request these selected queries and judge the relevance of the returned results. For navigational queries, Precision@Nis used to evaluate the accuracy of the results. In the experiment, N is set to 1. If the first document returned by the "IM search" is the result that user want, then Precision@N=1, else Precision@N=0. The experimental result is shown in Figure 6. It is obvious that most of these engines perform with the same accuracy. The reason is

navigational search query has a clear objective, and the document that users want to get will be ranked at the top by each search engine. For informational queries and transactional queries, DCG@N(Here N=5) is used to evaluate the performance of "IM search". The experimental result is shown in Figure 7. We can see, for most of uses, "IM search" performs better than all of other search engines. But for student 4, the result is unlike, the reason may be that the user interest is evenly distributed, it would slightly affect the rank positions of all documents. "IM search" performs the same accuracy with the underlying search engine "Baidu".

Table 2. Sample queries.

navigational query	informational query	transactional query
BBC Homepage	software engineering	free music downloads
Cornell University	mental health	Online Games
Bing	intellectual property	download Driver Booster
Facebook	android service tutorial	Thinking in Java pdf download
YouTube	industrial pollution	WeTransfer app

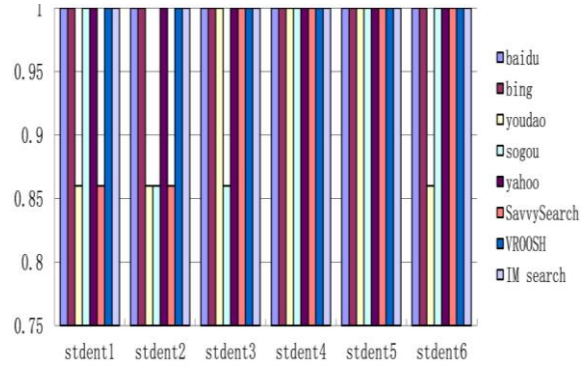

Figure 6. Mean value of P@N for the 7 navigational queries judged by the six students.

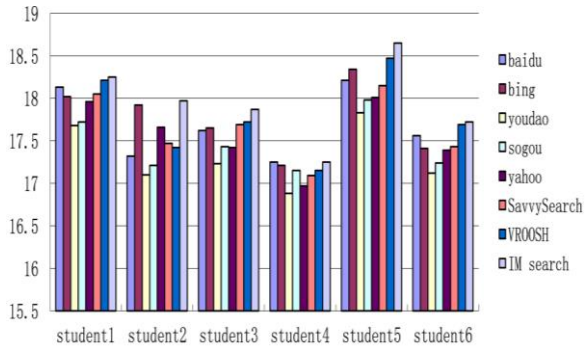

Figure 7. Mean value of DCG@N for the 43 non-navigationalqueries judged by the six students.

8.1. Experimental Results of the Proposed Schedule Strategy

To calculate the expertness model of underlying search engines, 20 representative query items are selected for each topic. These query items are submitted to Baidu, Youdao, Yahoo, Bing and Sogou respectively. Take the first 20 returned documents of each underlying search engine as the raw data to construct the concept lattice. Meanwhile, select 5000 click-through data from the query log of Sogou Lab to simulate the data for these underlying search engines. Then the expertness based on click-through data will be obtained. The final expertness of each underlying search engine is illustrated in Figure 8.

The experiment is designed to evaluate the expertness model. Selecting 20 query items randomly, manual annotation is utilized to judge the relevance of the first 30 returned documents for each underlying search engine, the score is calculated by Equation (16).

$$R = (2N_{rtd} + N_{udd})/2N_{rdd} \quad (16)$$

Where, N_{rtd} represents the number of relevant documents, N_{udd} is refers to undecided documents, N_{rdd} is refers to the number of returned documents.

Take the mean value of the selected 20 query items as the final relevance score of each underlying search engine for each topic, as shown in Figure 9. By comparing Figures 8 and 9, it can be seen that in most cases, the relevance of the underlying search engines is proportional to the calculated expertness model. When there is great difference in relevance between underlying search engines the expertness model is more able to describe this gap, find the most suitable engine.

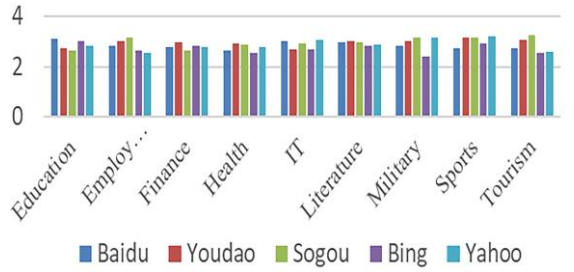

Figure 8. Expertness of each underlying search engine.

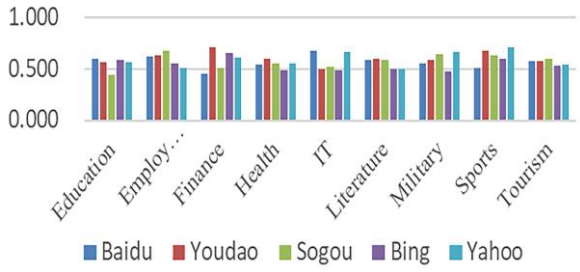

Figure 9. Relevance of each underlying search engine.

8.2. Experimental Results of The Proposed Result Merging Method

In order to evaluate the performance of the proposed result merging method, take Borda Fuse method and method proposed by Arzanian (written as AFCN) [2]as the baseline. Precision@N and DCG@N are used to analyze the results. We have selected 80 sample queries, including 12 navigational queries, 48 informational queries and 20 transactional queries. For each kind of queries, take the mean value of the results for different query items as the final score. Precision@N is used to evaluate the precision of the

results for navigational queries, DCG@N(Here N=10) is used for informational queries and transactional queries, as mentioned before. The experimental result is shown in Table 3. It is obvious that the proposed method outperforms Borda Fuse and AFCN for informational queries and transactional queries. For example, for a transactional query "java download", Table 4 shows top 6 URL appearing in Borda list. Table 5 shows the ranks the invited six users made. Table 6 shows the personalized ranking of AFCN for the six users. Table 7 shows the personalized ranking of our proposed result merging method for the six users. Shade box shows if personalized rank is equal to user checking's. It is obvious that our proposed method is better than Borda Fuse and AFCN.

Table 3. Precision of the proposed result merging method.

The type of queries	The Proposed Method(c=0.4)	Borda Fuse Method	AFCN
navigational queries(P@N)	1	1	1
informational queries(DCG@N)	18.02	17.62	17.98
transactional queries(DCG@N)	17.25	16.97	17.10

Table 4. Top 6 URL appearing in borda list.

	URL
1	http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html
2	https://www.java.com/en/download/
3	http://www.oracle.com/technetwork/java/javase/downloads/index-jsp-138363.html
4	https://www.java.com/zh_CN/download/
5	https://www.java.com/zh_CN/
6	http://www.oracle.com/technetwork/java/javase/downloads/index.html

Table 5. Ranking of the invited six users.

User1	User2	User3	User4	User5	User6
5	6	4	1	5	2
2	3	5	4	2	3
1	4	3	2	4	1
3	2	1	3	1	4
4	1	6	6	3	6
6	5	2	5	6	5

Table 6. Personalized ranking of AFCN.

User1	User2	User3	User4	User5	User6
5	5	2	1	5	2
1	3	4	3	4	1
3	6	3	2	6	6
2	2	5	6	1	4
4	1	6	4	2	3
6	4	1	5	3	5

Table 7. Personalized ranking of our proposed result merging method.

User1	User2	User3	User4	User5	User6
5	2	4	1	5	2
2	3	5	2	1	4
1	4	2	5	4	1
3	6	3	3	6	3
4	1	6	6	3	6
6	5	1	4	2	5

8.3. Experimental Results of The Proposed Recommendation

To verify the quality of recommendation, six students in our laboratory are invited to register in "IM search," request queries and judge the relevance of the result list with recommendation and without recommendation. Take MAP@5 as the estimation methods. For each query, only first 10 documents are selected. The mean values of MAP for different queries are calculated, as shown in Table 8.

Table 8. The mean values of MAP for the result lists.

User	The result list with recommendation	The result list without recommendation
user1	0.383	0.354
user2	0.352	0.259
user3	0.344	0.332
user4	0.219	0.219
user5	0.225	0.207
user6	0.445	0.399

From the Table, we can see, for most of users, the result list with recommendation get better performance. But for user4, the result is unlike, the reason is that most of query items he requested are belongs to navigational queries. The recommended documents are at the top of result list without recommendation.

9. Conclusions

This paper presents personalization mechanism for metasearch engine based on multi-agent system. By collecting user's click-through data, the metasearch engine has the ability to mine user interests, schedule the appropriate underlying search engines, and obtain the personalized result list. According to the group members' behaviours, the proposed personalization mechanism generates recommended results for users as well. Experimental Results show that the proposed metasearch engine performs better than the employed underlying search engines, metasearch engine Savvy Search and VROOSH. The personalization mechanism improves precision for metasearch engine. It is helpful for users to find their required information more convenient and effectively. But there are still open issues ahead needed to address:

- User interestsare obtained based on the click-through data. But others user behaviors, such as the browsing time, download history are also significant for analyzing user interest.
- Recommend personalized query words for different users.

Acknowledgement

This work is supported by the National Natural Science Foundation of China (61672401 and 61373045), the Fundamental Research Funds for the Central Universities of China (JB171001, JBZ171004, BDY221411, and K5051223008), the Pre-Research Project of the "Thirteenth Five-Year-Plan" of China (315***10101 and 315**0102), and the Technology Program of Xi'an (2017073CG/RC036 (XDKD004)).

References

[1] Alkhateeb F., AI-Fakhry A., Maghayreh E., Alijawarneh S., and AI-Taani A., "A Multi-agent-based System for Securing University Campus," *International Journal of Research and Reviews in Applied Sciences*, vol. 2, no. 3, pp. 223-231, 2010.

[2] Arzanian B., Akhlaghian F., and Moradi P., "A Multi-Agent Based Personalized Meta-Search Engine Using Automatic Fuzzy Concept Networks," *in Proceedings of 3rd International Conference on Knowledge Discovery and Data Mining*, Phuket, pp. 208-211, 2010.

[3] Craswell N., *Precision at n*, Springer, 2009.

[4] Du Y., Pei Z., Xiang D., and Li K., "New Fast Algorithm for Constructing Concept Lattice," *in Proceedings of International Conference on Computational Science and its Applications*, Kuala Lumpur, pp. 434-447, 2007.

[5] Dupret G., Murdock V., and Piwowarski B., "Web Search Engine Evaluation Using Click Through Data and A User Model," *in Proceedings of International Conference on World Wide Web*, Banff, 2007.

[6] Fan Y. and Gauch S., "An Adaptive Multi-Agent Architecture for the ProFusion* Meta Search System," *in Proceedings of Webnet 97-World Conference on the WWW, Internet and Intranet*, Toronto, pp. 1-2, 1997.

[7] Gauch S., Wang G., and Gomez M., "ProFusion*: Intelligent Fusion from Multiple, Distributed Search Engines," *Journal of Universal Computing*, vol. 2, pp. 637-649, 1996.

[8] Gulli A. and Signorini A., "Building an Open Source Meta-Search Engine," *in Proceedings of the 14th International Conference on World Wide Web*, Chiba, pp.1004-1005, 2005.

[9] Howe A., "A MetaSearch Engine that Learns Which Search Engines to Query," *Ai Magazine*, vol. 18, no. 2, pp. 19-25, 1997.

[10] Keyhanipour A., Moshiri B., Kazenmian M., Piroozmand M., and Lucas C., "Aggregation of Web Search Engines Based on Users' Preferences in WebFusion," *Knowledge-Based Systems*, vol. 20, no. 4, pp.321-328, 2007.

[11] Keyhanipour A., Moshiri B., Piroozmand M., and Lucas C., "WebFusion: Fundamentals and Principals of a Novel Meta Search Engine," *in Proceedings of International Joint Conference on Neural Networks*, Vancouver, pp. 4126-4131, 2006.

[12] Keyhanipour A., Moshiri B., Kazenmian M., Piroozmand M., and Lucas C., "A Multi-Layer/ Multi-Agent Architecture for Meta-Search Engines," *in Proceedings of International Conference on Artificial Intelligence and Machine Learning*, Cairo, 2005.

[13] Meng W., Yu C., and Liu K., "Building Efficient and Effective Metasearch Engines," *Acm Computing Surveys*, vol. 34, no. 1, pp. 48-89, 2001.

[14] Pan A., Yeung K., Moon K., Leung S., and Pan A., "Exploring the Potential of Using Agent-based Technology in Information Communication in Apparel Supply Chain Management," *in Proceedings of IEEE International Conference on Industrial Informatics*, Singapore, pp. 433-438, 2006.

[15] Sahoo P. and Parthasarthy R., "An Efficient Web Search Engine for Noisy Free Information Retrieval," *The International Arab Journal of Information Technology*, vol. 15, no. 3, pp. 412-418, 2018.

[16] Tonella P., "Using A Concept Lattice of Decomposition Slices for Program Understanding and Impact Analysis," *IEEE Transactions on Software Engineering*, vol. 29, no. 6, pp. 495-509, 2003.

[17] Vafadar S. and Barfourosh A., "Towards Intelligence Engineering in Agent-Based Systems," *The International Arab Journal of Information Technology*, vol. 12, no. 1, pp. 94-103, 2015.

[18] Wei X., Shi X., Kim S., Patrick J., Binkley J., Kong M., McClain C., and Zhang X., "Data Dependent Peak Model Based Spectrum Deconvolution for Analysis of High Resolution LC-MS Data," *Analytical Chemistry*, vol. 86, no. 4, pp. 2156-2165, 2011.

[19] Wu S. and Mcclean S., "Information Retrieval Evaluation with Partial Relevance Judgment," *in Proceedings of British International Conference on Databases*, Belfast, pp. 86-93, 2006.

[20] Zhou K., Zha H., Xue G., and Yu Y., "Learning the Gain Values and Discount Factors of DCG," *IEEE Transactions on Knowledge and Data Engineering*, vol. 26, no. 2, pp. 391-404, 2012.

Meijia Wang received the M.E. degree in College of Information Engineering from Northwest A&F University. Now she is a PhD. Candidate in Xidian University. Her main research interests include Agent-oriented software engineering, data analysis, and social network analysis.

Qingshan Li received his Ph.D. degree from Xidian University. Now he is a professor, PhD supervisor in Software Engineering Institute, Xidian University. His main research interests include agent-oriented software engineering, self-adaptive system, and data analysis.

Yishuai Lin received the Ph.D. degree from Université de Technologie de Belfort-Montbéliard. Now, she is a lecture in Software Engineering Institute, Xidian University, Her main research interests include agent-oriented software engineering, knowledge management, and product design.

Yingjian Li received the M.E. degree in Software Engineering Institute from Xidian University. His main research interests include Agent-oriented software engineering, information retrieval, and data analysis.

Boyu Zhou received the M.E. degree in Software Engineering Institute from Xidian University. His main research interests include Agent-oriented software engineering, and information retrieval.