An Efficient Web Page Recommendation Based on Preference Footprint to Browsed Pages

Shuhei Hayashi, Yuuki Inoshita and Satoshi Fujita
Graduate School of Engineering, Hiroshima University
1-4-1 Kagamiyama, Higashi-hiroshima, Hiroshima, Japan
email:{sh, ino, fujita}@se.hiroshima-u.ac.jp

***Abstract*—This paper proposes a new scheme for web page recommendation which reflects the preference of each user to the recommended pages in an efficient and effective manner. The basic idea of the scheme is to combine the notion of preference footprint to browsed pages with the collaborative filtering. More concretely, we introduce the notion of "tags" similar to conventional SBS (Social Bookmark Service), and attach all tags associated with a user to a page when it is browsed by him. We implemented a prototype of the proposed scheme, and conducted preliminary experiments to evaluate the performance of the scheme. The result of experiments indicates that it takes less than 0.5 sec to reorder a list of 500 URLs received from a search engine according to the preference of users.**

I. Introduction

According to the rapid popularization of ICT (Information and Communication Technology) and the WWW, the number of web pages in the Internet explosively increases in recent years. In order to find a demanded page from such enormous number of pages, we usually utilize *search engines* such as Google and Yahoo!, in which the contents of web pages are periodically collected to a central server, and after receiving a query from a user, the server retrieves collected information to return a list of URLs matching the query. An increase of the number of web pages also increases the number of *hits* for a given query, which significantly increases the ratio of *unnecessary* pages in the search result. Thus, in order to exclude such unnecessary pages from the search result, most of conventional search engines try to give an appropriate "rank" to each page, and recommend each user "higher ranked pages" which seem to be relevant to the interest of many users.

However, such a popularity-based ranking scheme does not help to find highly specific pages, such as the pages interesting for primary school children and the pages that attracts attention in a leading-edge field. In order to resolve such problem of conventional search engines, several ideas have been proposed in the literature to reflect the preference of users to the page ranking. A representative of such approaches is SBS (Social Bookmark Service) [6], [7], in which each user is allowed to conduct an explicit annotation of tags to each page. Another approach is to focus on the browsing history of users, as in Google personalized search. Although such history-based approach would be effective to automatically acquire the preference of users, many internet users do not want to register his private information to the system, and to disclose his browsing history to the other users.

In this paper, we propose a new web page recommendation system which can reflect the preference of each user to the recommended result in an efficient and effective manner. The basic idea of the proposed scheme is to combine the notion of *preference footprint* to browsed pages with the collaborative filtering. To this end, we introduce the notion of "tags" similar to conventional SBSs, but in contrast to SBSs, we associate those tags to each user, and attach *all* tags associated with a user to a page when it is browsed by the user. Tags attached to pages are shared by all users. As a result, we could acquire the information on the distribution of interest of users relevant to the page, without forcing each user to explicitly designate "what kind of page it is" as in SBSs. It significantly reduces the load of users compared with conventional SBSs. In addition, it reduces the psychological resistance of users, since it needs no registration, and the preference of users is processed merely in a stochastic manner. We expect that such favorable properties of the proposed system motivate many internet users to use our system, which increases the chance of collecting a large amount of tags compared with conventional tag-based page recommendation systems.

The remainder of this paper is organized as follows. Section II outlines related work including an overview of collaborative filtering. Section III describes our proposed system, and the details of our prototype system are described in Section IV. Finally, Section V concludes the paper with future problems.

II. Related Work

Conventional page ranking schemes can be classified into two categories; i.e., page-centered schemes and user-centered schemes. The former schemes calculate the ranking of pages merely by referring to the information attached to each page, and the latter schemes try to refine a (page-centered) ranking by taking into account the information on each user; i.e., it tries to *personalize* the resultant ranking.

A. Page-Centered Schemes

A page-centered ranking scheme returns the same list of URLs to all users without considering the personal information of each user. An advantage of such approach is high reproducibility of the search result. However, it is generally difficult to find an unpopular page from such universal list, and in addition, the outcome of page-centered schemes is easily affected by an adversarial behavior of selfish users such as SEO (Search Engine Optimization) [11], [12].

B. User-Centered Schemes

In contrast to such schemes, a user-centered scheme takes into account the personal information of each user to obtain a personal ranking which is not (significantly) affected by the popularity of pages. There are two types of user-centered schemes; i.e., schemes which merely rely on the personal information of a single user, and schemes based on the sharing of personal information among several users.

An example of the first type is Rerank.jp developed by Yamamoto *et al.* [4]. This scheme reorders the search result obtained by Yahoo! JAPAN Web API[1] using editorial operations manually conducted by each user. More concretely, each user executes either "emphasis" or "delete" of keywords shown in a tag cloud, to change the importance of keywords in the search result (i.e., a page containing an emphasized word in its title or snippet will be given a high rank, and a page containing a deleted word will be given a low rank). Another example is Google personalized search, which allows each user to have a ranking according to his preference which is not disclosed to the other users.

A representative of the second type is the collaborative filtering. In this method, several users sharing similar interests are classified into a cluster, and the rank of a page is refined by referring to the preference (or reputation) of the other users in the same cluster. In the next subsection, we overview related work concerned with the collaborative filtering.

C. Collaborative Filtering

Recommendation systems based on the collaborative filtering can be classified into two categories by the type of information used in the clustering; i.e., recommendation based on explicit information and recommendation based on implicit information.

Explicit Information: The term "explicit" means that it is explicitly designated by the users as in annotation, or provided via appropriate feedback tools. In general SBSs, annotation is regarded as a private comment attached to the bookmark; i.e., it generally increases the load of users. Although such load of users could be reduced by using memorandum instead of annotation, it causes another problem since memorandum may contain ad hoc representation specific to the users. Sasaki proposed a method [5] to overcome such problem, by focusing on the collection of pages which are attached common tags by different users, rather than focusing on the ad hoc representation of those tags.

User profile is another source of explicit information used in many existing recommendation systems. However, it is hard to collect such private information in our case, since general internet users do not want to disclose there preferences, although the load of collection could be reduced by using a simple inquiry form [3] and semantic web technologies [9], [10].

Implicit Information: There are a lot of information recommendation systems based on an implicit information collected from the users, e.g., Amazon.com, YouTube, and RSS feed provided by goo. In those systems, a variety of *vita information* are used as the source of information, such as browsing history, purchase history, operation record, and retrieval words and phrases history. Intuitively speaking, those systems try to trace the "footprints" implicitly given by the users.

A drawback of such approaches is that the contents relevant to a user must be simultaneously contained in the history of (another) user, to realize an efficient recommendation. Another drawback is that we can not distinguish between intentional and random visits merely via a sequence of visits. Such a drawback could be partially overcome by measuring the browsing time (e.g., one can identify a browsing as an intentional one if the browsing time exceeds a predetermined threshold), but we can not measure such time at the server side, and a measuring at the client side causes an additional cost.

III. Proposed Method

In this section, we propose a new page recommendation system. This system is designed to collect user's preference without forcing any cost and stress to the users.

A. Source of User Information

The basic technique used in the proposed system is user tag and preference footprint.

User tag (U-tag, for short) is a keyword (or a keyphrase) representing the preference of each user, e.g., baseball, major league, and red sox. Each user can register any U-tag representing his preference to the system, if it is not registered. When a user browses a web page by clicking a hyperlink provided by the system, all U-tags associated with the user are attached to the URL and are recorded to the system.

Preference footprint (PF, for short) is a collection of U-tags attached to the URL of a page, which represents the characteristics of the page and is used to recommend a page to an interested user. Note that PF is different from annotation of tags in SBSs, in a sense that: PF indicates "what type of users browsed that page," while the meaning of conventional tags is "what is the characteristics of that page." In addition, in our system, every user is also associated with a set of U-tags (similar to web pages), which enables a direct evaluation of the similarity of pages and users via calculating the similarity of the corresponding sets of U-tags (see Section III-C for the details).

B. Basic Flow of Recommendation

The basic flow of the recommendation process is described as follows:

1) At first, each user registers a set of U-tags representing his preference, to the system. Note that each user can register several U-tags. Let $T(u)$ denote the set of U-tags associated to user u.
2) User u sends a query to a search engine through our system. See Figure 1 for illustration. After receiving a

[1] http://developer.yahoo.co.jp/start/

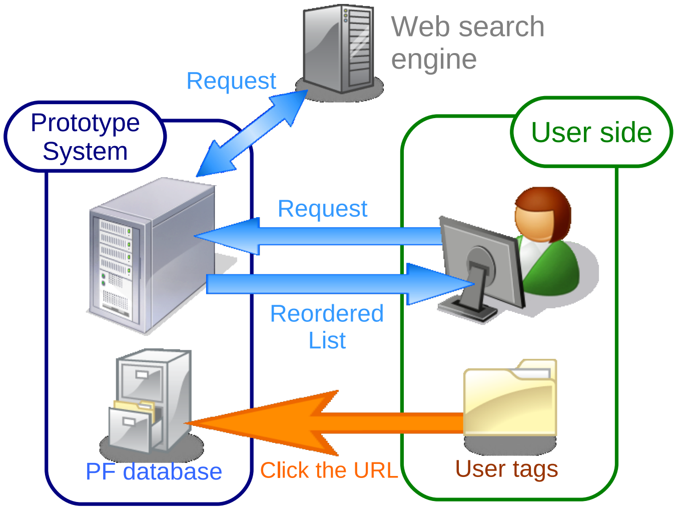


Fig. 1. Basic flow of recommendation.

list of URLs from the search engine, the system reorders the list according to the similarity between $T(u)$ and U-tags associated to each URL, and forwards the reordered list to the requesting user u.

3) The list of URLs shown to u is augmented with a list of corresponding PFs for each URL (if any). User u selects a URL in the list if he is interested in the contents of the web page. (At this point, u can refer to U-tags associated to URLs shown in the list, and he can use such tags to navigate the keyword search through our system. See Section III-D for the details.)
4) After receiving a URL selected by user u, the system adds U-tags contained in $T(u)$ to the set of U-tags associated with the selected URL.

In the following, we describe the way of calculating similarity between two sets of U-tags (Section III-C) and the way of clustering URLs according to the similarity between URLs (Section III-D).

C. Calculation of Similarity of Tags

We adopt the cosine similarity as the measure of similarity between two sets of U-tags. More concretely, we consider a vector space model (VSM) in which a set of U-tags is represented by a vector, and the similarity between two sets is evaluated by calculating the similarity between two corresponding vectors. Each coordinate in the VSM corresponds to a U-tag; i.e., a vector has a non-zero entry at the i^{th} coordinate iff the corresponding set contains the i^{th} U-tag. If it has a non-zero value, the value of the i^{th} coordinate is determined by applying a function known as tf-idf [2], in the following manner.

1) tf-idf: Let S be a multiset of U-tags associated with a URL or a user, and t be a U-tag contained in S. At first, function tf is defined as the number of occurrences of t in S (if S is a set of U-tags associated to a user, the value of function $\mathsf{tf}(t,S)$ is either zero or one). Next, function idf is defined such that a U-tag specific to the set takes a large value, and U-tags commonly contained in many sets take a small value. More concretely, function idf is defined as follows:

$$\mathsf{idf}(t) \quad = \quad \log(N/n_t) \qquad (1)$$

where N is the total number of multisets, and n_t is the number of multisets containing t. By definition, a popular tag which is contained in many sets takes a small value close to zero, and conversely, a rare tag which is contained in few sets takes a large value.

Function tf-idf, which is intended to represent the importance of t in S, is formally defined as follows:

$$\phi(t,r) \quad \overset{\text{def}}{=} \quad \mathsf{tf}(t,r) * \mathsf{idf}(t).$$

Using such notions, similarity between two multisets X and Y is defined as follows:

$$sim(X,Y) \quad \overset{\text{def}}{=} \quad \frac{\sum_{t\in T}\{\phi(t,X)*\phi(t,Y)\}}{\sqrt{\left(\sum_{t\in T}\phi(t,X)^2\right)*\left(\sum_{t\in T}\phi(t,Y)^2\right)}}$$

where T denotes the set of all U-tags.

2) Procedure: Without loss of generality, let us assume that our system has already known set $T(u)$ of U-tags associated with user u. After receiving a list of URLs as the result of query issued by user u, the system conducts a reordering of those URLs according to the similarity to $T(u)$; i.e., it sequentially calculates the similarity to $T(u)$ for each URL, and sorts those URLs in a non-increasing order of the similarity. Since $\phi(t,T(u))$ takes a value either zero or one for any t, in the calculation of similarity between X and $T(u)$, we may simply add $\phi(t,X)$'s for each $t \in T(u)$ (note that the denominator of the formula takes a constant value for given X and $T(u)$).

In order to speed up such selective additions, in the proposed system, we adopt Bloom filter [8] to avoid unnecessary search of non-existing elements (each vector is represented as a list of elements, since we could not bound the length of vector in advance).

D. Hierarchical Clustering of URLs

In addition to the reordering of a list of URLs, in the proposed system, we prepare a mechanism to navigate the search of a target page via indicating U-tags associated with each search result. Recall that the proposed system is designed to provide a higher rank to a URL if it is similar to the preference of the requester. In addition, by referring to the set of U-tags attached to the listed URLs, a user can recognize "which U-tag associated to him is actually used in the reordering." Ordering of URLs can change if another set of U-tags is used in the reordering, and it motivates a navigation of page ranking via the change of the reference set of U-tags which is initially set to the set of U-tags associated with the user.

In the proposed system, we realize such navigation via the change of reference set by introducing two new techniques, i.e., 1) hierarchical clustering of URLs and 2) identification of U-tags which characterize such clusterings. This is an extension of the method proposed in [2]. More concretely, a clustering of several subclusters corresponds to a *threshold* concerning to the similarity of those subclusters, where similarity of two subclusters is defined as follows:

$$sim(C_1, C_2) \quad = \quad \min_{x \in C_1, y \in C_2} \{sim(x, y)\} .$$

The maximum size of each cluster (i.e., the maximum number of subclusters contained in each cluster) is given as a parameter. Appropriate value of such parameter will be determined through extensive simulations, but it is left as a future work. Note that the computation time required for such calculation can be hidden in general situations, since we can calculate the similarity between any two URLs in advance. Although the similarity between two URLs should be recalculated periodically according to an increase of the number of U-tags associated with each URL, we have an intuition such that the period of such update is relatively long, e.g., few weeks and few months.

Calculated values of similarity among URLs are informed to the requester with a list of URLs, and actual clustering of URLs is conducted at the client side. At the side of the resulting hierarchical clustering, the system displays a set of U-tags associated with each cluster, in order to navigate the control of the search result towards a finding of a target page.

E. Observation

A key issue in our proposed scheme is how to collect a large number of U-tags from users having various preferences. As was described previously, the basic operation in our proposed system is to attach *all* U-tags to the selected URL. In other words, we do not consider any discrimination or filtering of U-tags in this operation, since it is difficult to realize a selective attachment of tags without the aid of voluntary users. Instead of taking such approach, we adopt an optimistic strategy such that an uncontrolled accumulation of U-tags causes an appropriate distribution of the frequency of U-tags reflecting the interest of the browsing users.

IV. Prototype System

We implemented a prototype system to demonstrate the availability of the proposed scheme. The program is written in PHP5 (server) and JavaScript (client). The environment of the server is as follows: Intel(R) Core(TM)2 Duo CPU E7400, 2GB Memory, Ubuntu/8.10, Apache/2.2.11, and PHP/5.2.9. As the search engine, we used Yahoo! JAPAN Web API, which returns at most 50 URLs for each query.

A. Data Structure

In the prototype system, a set of U-tags associated to each user is stored in a cookie in his personal browser, and each user can register a U-tag relevant to him either by specifying a keyword in a free description form, or by clicking U-tags which have already been registered by the other users. We adopt XML as the basic format of transmitted messages, because of its popularity and the ease of manipulation. PF (preference footprint) of each user is also described in the form of XML, and an attachment of PF to a URL is simply done by merging two XML forms.

TABLE I
Format of PF.

Item	Explanation
URL	Used as a page ID
Tag	Name of tag
Times	Number of annotations of the tag (i.e., tf value)

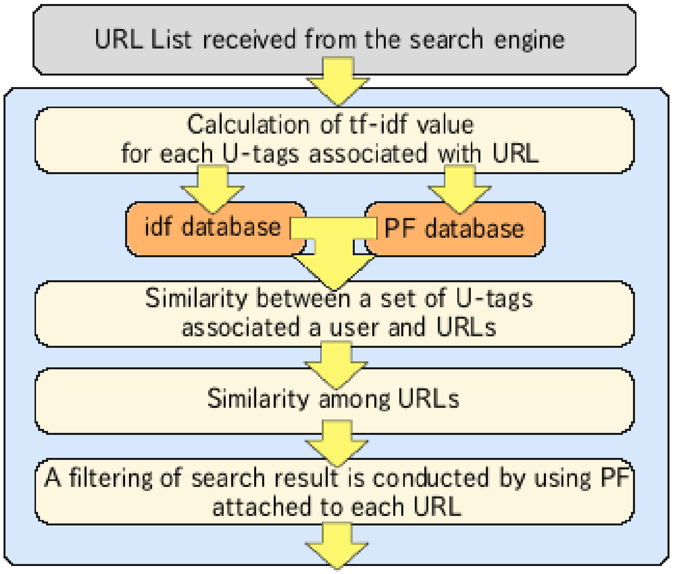

URL List received from the search engine
Calculation of tf-idf value
for each U-tags associated with URL
idf database
PF database
Similarity between a set of U-tags
associated a user and URLs
Similarity among URLs
A filtering of search result is conducted by using PF
attached to each URL

Fig. 2. Basic functions used in the proposed scheme are implemented.

As a concrete set of U-tags, we used livedoor clip datasets[2] in our experiments. This data set was published in December 2008 by livedoor clip[3] which is one of the most popular SBSs in Japan. Each record in the data set consists of four fields, i.e., user ID, the date of bookmarking, URL, and attached tags. In the experiments, we converted it into a data set such that: 1) each record in the set is indexed by URL, and 2) the record concerned with a URL contains all U-tags associated with the URL (note that the original data set contains several records corresponding to each URL). The resultant database, which will be referred to as PF database hereafter, consists of 200 thousand URLs and 150 thousand U-tags.

B. Basic Functions

Basic functions used in the proposed scheme are implemented as follows (See Figure 2 for illustration): 1) calculation of tf-idf value for each U-tag associated with URL is executed after receiving a message from a user; The tf and idf values are periodically updated. 2) similarity between a set of U-tags associated with a user and URLs contained in a URL list received from the search engine, is also calculated by the server and each client merely receives the result of such calculation, i.e., after receiving such information, each client conducts a reordering of the search result using JavaScript; 3) similarity among URLs, which is necessary to realize a

[2] http://labs.edge.jp/datasets/
[3] http://clip.livedoor.com/

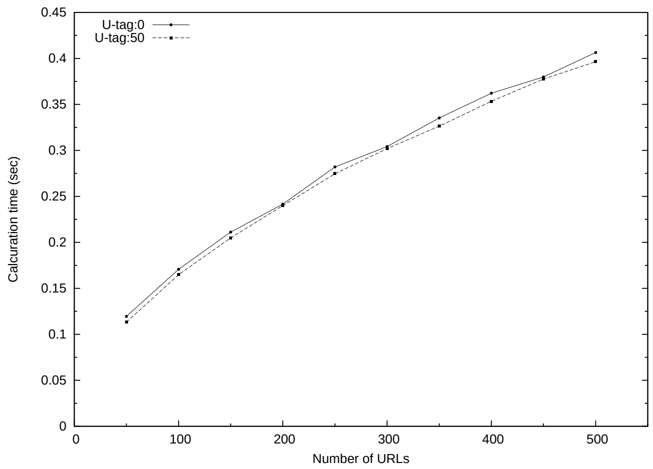


Fig. 3. Average calculation time of the similarity to the received URLs for each user.

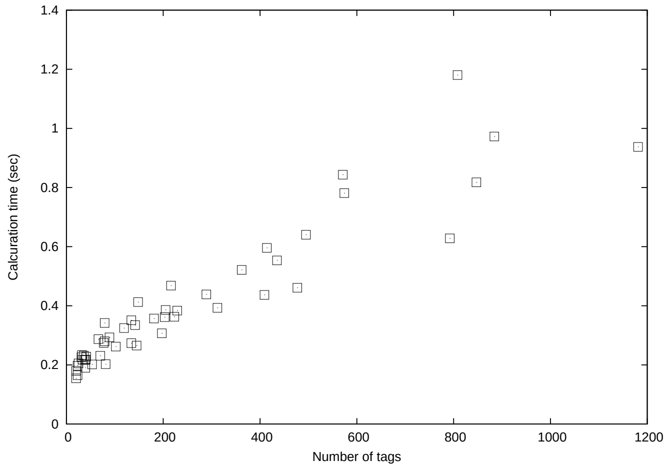


Fig. 4. Relationship between the number of U-tags and the overall calculation time.

hierarchical clustering of URLs at a client, is calculated by the server after completing the calculation of tf-idf (we are going to move this part to each client); and finally, 4) a filtering of search result is conducted by using PF attached to each URL.

C. Quick Retrieval of Record

In order to realize a quick access to a record in the database, in the prototype system, we divide the PF database into 256 files in the following manner: 1) each file is associated with two hex digits from 00 to ff, 2) each URL is mapped to a hex string by an appropriate hash function, and 3) each of those 256 files stores PF data concerned with URLs such that a prefix of the hex string corresponding to the URL matches hex digits associated with the file. In addition, as a way of detecting the non-existence of a particular data in the database, we adopt a Bloom filter of 164 bits (details of parameters are omitted in this extended abstract).

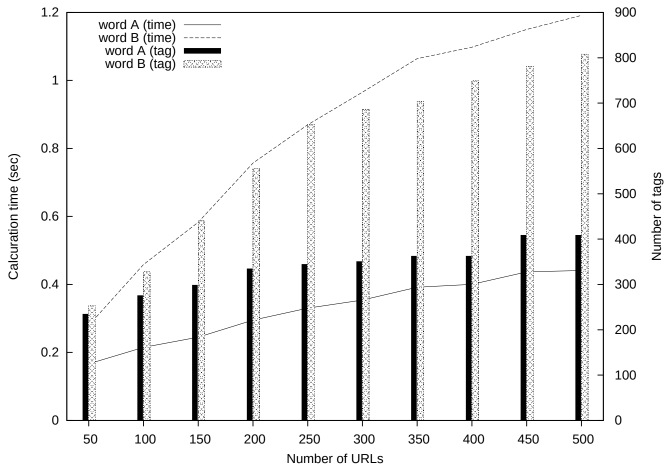


Fig. 5. Comparison of two query words.

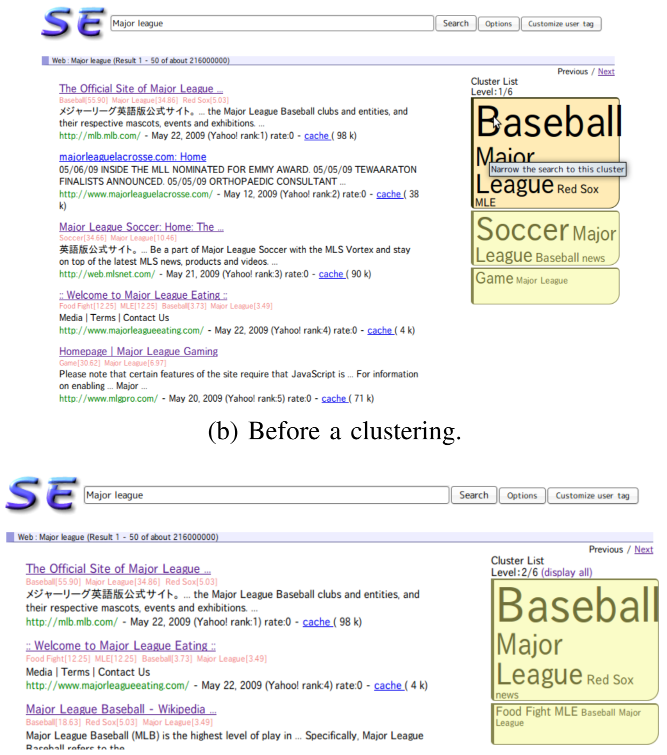
D. Evaluation

We conducted preliminary experiments to evaluate the performance of the prototype system. In the experiments, we consider two users u_1 and u_2, where user u_1 is attached 50 U-tags which are identical to "top 50" of the search word ranking of Yahoo! Japan[4], and user u_2 is attached no U-tags. We measure the time required for calculating the similarity between the set of U-tags attached to each user and the set of U-tags attached to URLs received from the search engine, by varying the number of received URLs from 50 to 500.

Figure 3 summarizes the result. The vertical axis of the figure represents an average calculation time over 20 runs, where "U-tag:50" indicates the calculation time for user u_1 and "U-tag:0" indicates the calculation time for user u_2. From the figure, we can observe that the calculation time monotonically increases as increasing the number of URLs contained in the search result (e.g., it takes 0.25 sec for 200 URLs and 0.4 sec for 500 URLs and), but it is not affected by the number of U-tags attached to each user. The rate of increasing the calculation time gradually decreases as increasing the number of URLs, which is due to the reduction of the percentage of URLs attached U-tags; i.e., in the data set used in the experiments, U-tags are attached to a limited number of (popular) URLs received from the search engine.

Figure 4 shows the calculation time of the scheme for each query, where each point in the figure corresponds to a query which is averaged over 20 runs. The vertical axis represents the total number of U-tags associated with the resulting URLs, and the vertical axis represents the total calculation time. This result indicates that the calculation time of the scheme depends on the number of U-tags contained in the search result independent of the number of URLs in the search result. In order to observe the relationship between those factors in more detail, we selected two query words w_A and w_B, and compared the calculation time and the number of U-tags concerned with those words by changing the number of URLs

[4] The period of counting the search words is from January 1 to June 30, 2008. See `http://searchranking.yahoo.co.jp/ranking2008firsthalf/` for the details



Web : Major league (Result 1 - 50 of about 216000000)

The Official Site of Major League ...
Baseball[55.90] Major League[34.86] Red Sox[5.03]
メジャーリーグ英語版公式サイト。... the Major League Baseball clubs and entities, and their respective mascots, events and exhibitions. ...
http://mlb.mlb.com/ - May 22, 2009 (Yahoo! rank:1) rate:0 - cache (98 k)

majorleaguelacrosse.com: Home
05/06/09 INSIDE THE MLL NOMINATED FOR EMMY AWARD. 05/05/09 TEWAARATON FINALISTS ANNOUNCED. 05/05/09 ORTHOPAEDIC CONSULTANT ...
http://www.majorleaguelacrosse.com/ - May 12, 2009 (Yahoo! rank:2) rate:0 - cache (38 k)

Major League Soccer: Home: The ...
Soccer[34.66] Major League[10.46]
英語版公式サイト。... Be a part of Major League Soccer with the MLS Vortex and stay on top of the latest MLS news, products and videos. ...
http://web.mlsnet.com/ - May 21, 2009 (Yahoo! rank:3) rate:0 - cache (90 k)

:: Welcome to Major League Eating ::
Food Fight[12.25] MLE[12.25] Baseball[3.73] Major League[3.49]
Media | Terms | Contact Us
http://www.majorleagueeating.com/ - May 22, 2009 (Yahoo! rank:4) rate:0 - cache (4 k)

Homepage | Major League Gaming
Game[30.62] Major League[6.97]
Please note that certain features of the site require that JavaScript is ... For information on enabling ... Major ...
http://www.mlgpro.com/ - May 20, 2009 (Yahoo! rank:5) rate:0 - cache (71 k)

(b) Before a clustering.

Web : Major league (Result 1 - 50 of about 216000000)

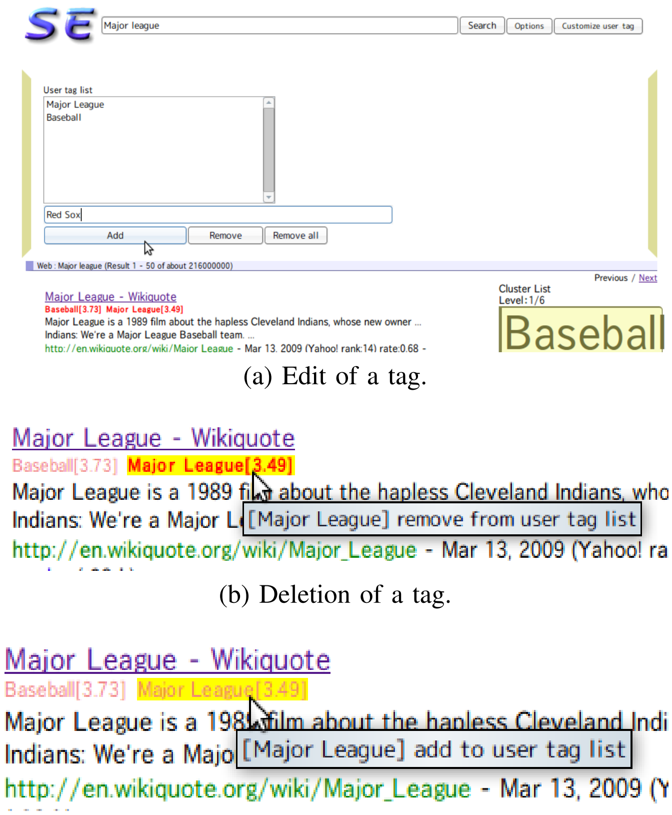
The Official Site of Major League ...
Baseball[55.90] Major League[34.86] Red Sox[5.03]
メジャーリーグ英語版公式サイト。... the Major League Baseball clubs and entities, and their respective mascots, events and exhibitions. ...
http://mlb.mlb.com/ - May 22, 2009 (Yahoo! rank:1) rate:0 - cache (98 k)

:: Welcome to Major League Eating ::
Food Fight[12.25] MLE[12.25] Baseball[3.73] Major League[3.49]
Media | Terms | Contact Us
http://www.majorleagueeating.com/ - May 22, 2009 (Yahoo! rank:4) rate:0 - cache (4 k)

Major League Baseball - Wikipedia ...
Baseball[18.63] Red Sox[5.03] Major League[3.49]
Major League Baseball (MLB) is the highest level of play in ... Specifically, Major League Baseball refers to the

(b) After clustering.

Fig. 6. Screen shot (1): Clustering.



Web : Major league (Result 1 - 50 of about 216000000)

Major League - Wikiquote
Baseball[3.73] Major League[3.49]
Major League is a 1989 film about the hapless Cleveland Indians, whose new owner ...
Indians: We're a Major League Baseball team. ...
http://en.wikiquote.org/wiki/Major_League - Mar 13, 2009 (Yahoo! rank:14) rate:0.68 -

(a) Edit of a tag.

Major League - Wikiquote
Baseball[3.73] Major League[3.49]
Major League is a 1989 film about the hapless Cleveland Indians, who
Indians: We're a Major L [Major League] remove from user tag list
http://en.wikiquote.org/wiki/Major_League - Mar 13, 2009 (Yahoo! ra

(b) Deletion of a tag.

Major League - Wikiquote
Baseball[3.73] Major League[3.49]
Major League is a 1989 film about the hapless Cleveland Indi
Indians: We're a Majo [Major League] add to user tag list
http://en.wikiquote.org/wiki/Major_League - Mar 13, 2009 (Y

(c) Insertion of a tag.

Fig. 7. Screen shot (2): Manipulation of tags.

in the search result. Figure 5 shows the result. The horizontal axis is the calculation time (left hand side) and the number of U-tags contained in the search result (right hand side). As shown in the figure, the difference of the calculation time is certainly proportional to the difference of the number of U-tags.

Screen shots of the prototype system are shown in Figures 6 and 7.

V. Concluding Remarks

In this paper, we proposed a new scheme for web page recommendation which reflects the preference of each user to the recommended result in an efficient and effective manner. We conducted preliminary experiments to evaluate the performance of the scheme and showed the feasibility of the scheme.

A future work is to improve the efficiency of the proposed scheme, by adopting a selective attachment of U-tags to the browsed pages, and/or by tuning parameters used in the scheme (e.g., the cluster size). We are also planning to open the prototype system for public use, in order to demonstrate the effectiveness of our web page recommendation scheme in the real world.

References

[1] Okkyung Choi, Sangyong Han, and Ajith Abraham. Integration of Semantic Data Using a Novel Web Based Information Query System. In *International Journal of Web Services Practices*, Vol1, No.1-2, page 21–29, 2005.

[2] Shepitsen Andriy, Gemmell Jonathan, Mobasher Bamshad, and Burke Robin. Personalized recommendation in social tagging systems using hierarchical clustering. In *Proceedings of the 2008 ACM conference on Recommender systems*, pages 259–266, 2008.

[3] Tak W. Yan, and Hector Garcia-Molina SIFT—A tool for Wide-Area Information Dissemination. In *Proceedings of the 1995 Usenix Technical Conference*, pages 177–186, 1995.

[4] Takehiro Yamamoto, Satoshi Nakamura, and Katsumi Tanaka. Rerank-By-Example: Efficient Browsing of Web Search Results. In *Proceedings of the 18th International Conference on Database and Expert Systems Applications*, pages 801–810, 2007.

[5] Akira Sasaki, Miyata Takamichi, Inazumi Yasuhiro, Aki Kobayashi and Sakai Yoshinori. Web Content Recommendation System Based on Similarities among Contents Cluster of Social Bookmark [in Japanese] . In *Information Processing Society of Japan*, pages 14–27, 2007.

[6] Delicious http://delicious.com/

[7] Netvouz http://www.netvouz.com/

[8] Burton H. Bloom Space/time trade-offs in hash coding with allowable errors. In *Communications of the ACM*, Vol 13, pages 422–426, 1970.

[9] Zhou Xujuan, Wu Sheng T., Li Yuefeng, Xu Yue, Lau Raymond Y. K., and Bruza Peter D. Utilizing Search Intent in Topic Ontology-Based User Profile for Web Mining. In *Proceedings of the 2006 IEEE/WIC/ACM International Conference on Web Intelligence*, pages 558–564, 2006.

[10] Alexander Pretschner, and Susan Gauch. Ontology based personalized search. In *Proceedings of the 11th IEEE International Conference on Tools with Artificial Intelligence*, pages 391–398, 1999.

[11] Dennis Fetterly, Mark Manasse, and Marc Najork. Spam, Damn Spam, and Statistics: Using statistical analysis to locate spam web pages. In *Proceedings of the Seventh International Workshop on the Web and Databases*, pages 1–6, 2004.

[12] Timothy Jones, Ramesh Sankaranarayana, David Hawking, and Nick Craswell. Nullification test collections for web spam and SEO. In *Proceedings of the 5th International Workshop on Adversarial Information Retrieval on the Web*, pages 53–60, 2009.