Modeling User's Non-functional Preferences for Personalized Service Ranking

Rozita Mirmotalebi[1], Chen Ding[1], and Chi-Hung Chi[2,3]

[1] Department of Computer Science, Ryerson University, Toronto, Canada
{rozita.mirmotalebi,cding}@ryerson.ca
[2] School of Software, Tsinghua University, Beijing, China
[3] Intelligent Sensing and Systems Laboratory – CSIRO, Hobart, Australia
chihungchi@gmail.com

Abstract. Modeling users' online behavior has great benefit for many e-Commerce web sites and search engines. In the context of software service selection, if we could understand users' personal preferences, we could rank the services in a more satisfactory way. Many users have some general preferences on the desired values of non-functional properties (e.g. provider history, service popularity, etc.) of services, even if they may not explicitly define them. In this paper, we propose to build user profiles on these non-functional preferences, and then use them to personalize the ranking results for individual users. Our experiment showed that personalized ranking could promote the services matching with the user preferred non-functional values to higher positions, making it easier for users to identify their desired services. We also tested how different factors impact the degree of improvement on the ranking accuracy.

Keywords: Service Ranking, Service Selection, Non-functional Preference, User Modeling, Personalization.

1 Introduction

When more and more providers publish and host their software services in the cloud, it becomes a challenge for users to find their desired services. Although cloud service directories, or service search engines such as Seekda[1] can provide the searching function over users' functional requirements, they often lack the support for the further selection among a list of functionally similar services. If this list is long, users are still overwhelmed by the amount of information they need to process.

When multiple services implement a same function, users often select them based on their non-functional properties, such as service cost, reputation of the provider, reliability of the service, etc. Although users may have explicit requirements (e.g. cost < $15/month) guiding the selection process, their implicit general preferences also play a key role in this process. For instance, one user may always choose the service from a provider with a good reputation, another user may normally choose the service

[1] http://webservices.seekda.com

C. Liu et al. (Eds.): ICSOC 2012, LNCS 7636, pp. 359–373, 2012.

with the highest reliability, and yet another user may balance between the cost and the response time. Personalized search could provide personalized ranking results to individual users based on their profiles, and it is a well accepted solution to deal with the information overload problem on the web [8]. For software service selection, if we could create users' personal profiles based on their non-functional preferences, these profiles can be used to build a personalized ranking list of candidate services. Here, we use the term non-functional preferences to refer to properties users have concerns on and the types of values they prefer (e.g. higher/lower values) on these properties.

There have been many research efforts on non-functional property based service selection. In these studies, users are normally required to specify their non-functional requirements for every single query. However, sometimes users may not want to spend time on defining them, or don't know how to define them, plus there are default requirements which always apply (e.g. always prefer a service with lower cost). In this paper, we propose to model users' general non-functional preferences for service selection. Since a personal profile is built for every user, there is no need to type in the general non-functional requirement each time any more. If the user has a different requirement for a query, he could always specify it explicitly, which would overwrite the one in his profile. In the paper, we assume there are existing matchmaking algorithms we can use so that we focus our study on the personalized ranking part. Also we only focus on the selection system which interacts with a human user instead of a software agent. This setting makes the profiling process feasible and meaningful.

People make their decisions based on various criteria. For service selection, there are some common criteria which are important to almost everyone, and there are also specific criteria which are only important to individual users or domains. It is definitely impossible to include every single criterion in the user model. However, our algorithm and the system design are generic enough so that we could always expand or tailor our user model for a particular domain or a user group. In the paper, we choose some common selection criteria whose values we could find in our dataset.

User preferences can be captured either explicitly or implicitly. We have carefully-designed user interfaces to elicit users' non-functional preferences explicitly. When the explicit profile is missing, we have the implicit profile with user's non-functional preferences inferred by checking the values of the non-functional properties of the services a user has invoked in the past. The profile could evolve over time by asking users to update their profiles or by observing the changing invocation patterns.

There are two major contributions of the paper. First, to the best of our knowledge, user modeling on their non-functional preferences for software service selection is a very new idea. Standard recommender systems cannot accomplish this task without major modifications. We also propose an approach which could implicitly infer user preferences based on their invocation histories. Second, personalized ranking based on users' general interests and preferences has been successfully used for web search, while it hasn't been applied to software service selection yet. In many existing personalized service ranking algorithms, personalization is not achieved through the long-term user profiles and not many properties are considered. Ours is unique in its focus on users' long-term preferences on multiple non-functional service properties.

The rest of the paper is organized as follows. We show a very simple motivating example in Section 2. Then we discuss the user modeling process in Section 3. Our personalized service ranking algorithm is explained in Section 4. In Section 5, we explain how we design the experiment and build a proper dataset to validate the importance of the personalized ranking. In Section 6, we review the related work. Finally in Section 7 we conclude the paper and list the future work.

2 A Motivating Example

Assume that *Bob* and *Dave* have registered in our service selection system and used our system for a while. When registration, *Bob* explicitly specified his non-functional preferences as: (provider name: *P*, service availability: high), which means he prefers services from a provider named *P*, and also services with high availability. *Dave* didn't specify any personal preferences. But his invocation history shows that out of 20 services he invoked, 18 services have high ratings, and 15 services are from well-established providers. So the system concludes his preferences as: (provider history: long, service rating: high). We also assume that all preferences are equally important to users.

Now both of them are searching for a report generating service. 4 services are returned and part of their non-functional property values are listed in Table 1.

Table 1. Non-functional property values of 4 services

	Provider Name	Provider History	Service Availability	Service Rating
s_1	P	< 1 year	90%	4
s_2	P	< 1 year	95%	3
s_3	Q	5 years	99%	4
s_4	R	2 years	70%	5

By checking two users' non-functional preferences, the system generates a different personalized ranking list for *Bob* and *Dave* (the detailed steps are explained later). The former is: s_2, s_1, s_3, s_4, and the latter is: s_3, s_4, s_1, s_2. With this personalized result, both users could find their desired services easily. Although this is an over-simplified example, it shows the benefit of the personalized ranking. In the real scenario when there are more services returned, the benefit would be more obvious.

3 Modeling User's Non-functional Preferences

3.1 Defining Non-functional Preferences

Non-functional property based software service selection is similar to the consumer purchase decision process [4]. There are many factors affecting this process. In this paper we are not aiming at identifying and defining them systematically. We just pick a few common purchase criteria (e.g. preference on a certain provider because of the brand loyalty) which we could find their values in our dataset and divide them into two categories: provider related, and service related. For the latter, a significant group is QoS (Quality of Service) properties [12]. Due to the restriction of our dataset,

we only choose two of them (i.e. availability, response time) in this paper. Normally when a user has a preference on a property, it is not one single fixed number, and instead, it could be a range of values expressed using some fuzzy linguistic terms, e.g. a service with a *good* reliability, a service with a *low* cost, or it could be a set of preferred values, e.g. a service from a few well-known providers *A*, *B* and *C*. Based on this observation, all of the non-functional preferences defined below are either sets or fuzzy values. The first four are provider related and the rest are service related.

- ***PN*** (Provider Names): the names of the user preferred providers (a set value). The provider name is either a company's legal name, or the top level domain name of the hosting web site. Users could have multiple preferred providers.
- ***PL*** (Provider Locations): the preferred locations of the providers (a set value). The location is the geographical location of either the company or the hosting web site. It could be further categorized into two types: ***PCT*** (Provider Continent) and ***PCY*** (Provider Country). Users could have preferences on one of them or both.
- ***PH*** (Provider History): the preferred history of the providers. The provider history is defined as the number of years the provider has been established. Preference on this property is usually defined as a fuzzy value such as short, medium and long.
- ***PP*** (Provider Popularity): the preferred level of popularity of the providers (a fuzzy value). The provider popularity is measured by the ratio of the number of invocations of services from the provider over the total number of invocations.
- ***SL*** (Service Languages): preferred languages of the service outputs (a set value).
- ***SH*** (Service History): the preferred history of the services (a fuzzy value). The service history is defined as the number of years the service has been offered.
- ***SF*** (Service Freshness): the preferred level of freshness of the services (a fuzzy value). The service freshness is measured by its latest update time. If a service is regularly maintained and updated, it is considered fresh.
- ***SP*** (Service Popularity): the preferred level of popularity of the services (a fuzzy value). It is measured by the ratio of the number of invocations of the service over the total number of invocations of all functionally equivalent services.
- ***SR*** (Service Rating): the preferred ratings of the services (a fuzzy value). The rating of a service is the average of all received ratings on the service.
- ***SC*** (Service Cost): the preferred cost level of the services (a fuzzy value). The cost of a service is the fee a user needs to pay in order to use the service.
- ***SD*** (Service Documentation): the preferred level of documentation of the services (a fuzzy value). It measures how much documentation a service provides to its users and its fuzzy values are none, partial and good.
- ***SA*** (Service Availability): the preferred availability of the services (a fuzzy value). Service availability measures the probability a service is operating normally and can be accessed by users successfully.
- ***SRT*** (Service Response Time): the preferred response time of the services (a fuzzy value). The response time of a service is the time from the end of the service request to the beginning of the response.

This list is far from complete. However, we can always include more properties when necessary because our framework is generic and extensible. Also right now we only

consider 2 types of preferences. It is possible to extend our system to include more types, such as dislike preferences, composite preferences, as specified in [3].

3.2 Data Collection for User Modeling

To understand how our system collects the user data and generates the personalized ranking result, we first explain its architecture model as shown in Figure 1. This architecture model is an extension to our previous work [15]. We use the user-side proxy to collect the service usage data for individual users. If a user fills the required preference forms, the explicit user profiling component will generate the user model. If the user does not fill the forms or leave some values undefined, the implicit user profiling component will check the services the user invoked in the past on their non-functional properties to see whether any general patterns can be observed, which will then be saved in the user model. In the selection stage, the personalized ranking component will rank the matching services based on the generated user profile.

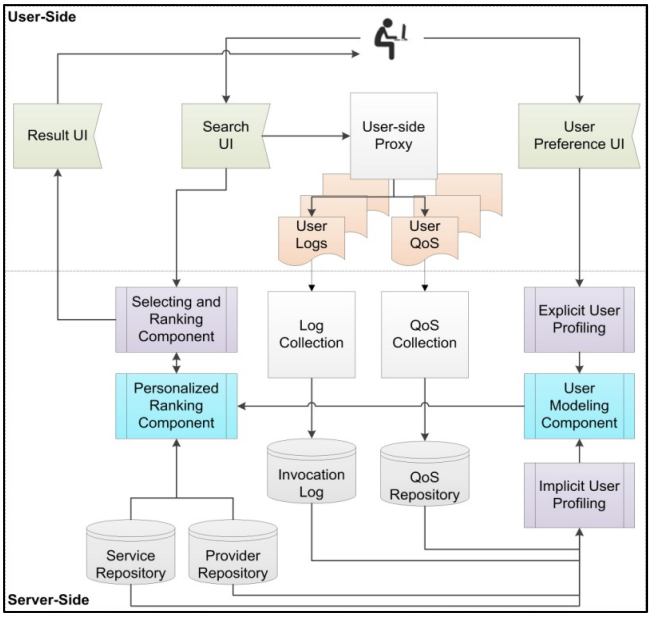

Fig. 1. Architecture of our personalized service ranking system

3.3 Implicit User Modeling

Below we list the notations we use to represent providers, services and users.

- P: the set of all the providers in the repository.
- p_i: the i-th software service provider in P. It has 4 properties – name, location, history, and popularity, and their values can be found in the provider repository. We use $p_i.n$, $p_i.l$, $p_i.h$, $p_i.p$ to represent these properties, and furthermore $p_i.ct$ and $p_i.cy$ for its continent and country.

- S: the set of all the services in the repository.
- s_i: the i-th software service in S. It has 9 properties – language, history, freshness, popularity, rating, cost, documentation, availability, and response time, and their values can be found in the service repository. We use $s_i.l$, $s_i.h$, $s_i.f$, $s_i.p$, $s_i.r$, $s_i.c$, $s_i.d$, $s_i.a$, $s_i.rt$ to represent these properties and $s_i.pd$ to refer to its service provider.
- U: the set of all the users of the system.
- u_i: the i-th user of our system. The user profile includes user's login information such as user id and password, user's non-functional preferences on 4 provider-related and 9 service-related properties, as well as the other useful information such as services invoked by this user. To refer to his preferences, we use $u_i.PN$, $u_i.PL$ ($u_i.PCT$, $u_i.PCY$), $u_i.PH$, $u_i.PP$, $u_i.SL$, $u_i.SH$, $u_i.SF$, $u_i.SP$, $u_i.SR$, $u_i.SC$, $u_i.SD$, $u_i.SA$, and $u_i.SRT$ respectively.

The non-functional preferences can be categorized into two groups based on their data types. The first group includes PN, PL, SL, and their values are all sets, e.g. $u_i.PN = \{A, B\}$ where A, B are provider names. The second group includes all of the rest, and they all have fuzzy values, e.g. $u_i.PH$ = long. Each group has its own way of calculating the implicit preference values.

In the first group, we use PN as an example, and other preferences follow the same calculation steps. Assume that the number of invoked services by u_i is NI_i, and among them, the number of services from p_j is NI_{ij}, we calculate the invocation frequency of provider p_j by user u_i as:

$$IFP_{ij} = \frac{NI_{ij}}{NI_i} \tag{1}$$

Then we compare the result with a predefined threshold T_{IF}, and if it is above the threshold, we consider p_j as user preferred and add it into $u_i.PN$. It is possible that we may have an empty set for $u_i.PN$. The threshold value could be chosen based on the statistical summary of the invocation history (e.g. the cut-off value to get top k providers), or set as a fixed value, e.g. 50% (a majority percentage).

For PL, we consider PCT and PCY separately. To calculate its invocation frequency, we need to count the number of services from a provider of a certain continent or country. For SL, we count the number of services in a certain language.

In the second group, we use PH as the example. There are 3 fuzzy values for the provider history. In our system, their value ranges are defined as: short ($\leq$ 1.5 years), medium (>1.5 years and $\leq$ 3 years), and long (> 3 years). The choice of these values is based on the dataset we have. It could be redefined for different datasets. Also the number of fuzzy values can be redefined.

For all the services u_i has invoked, we find their corresponding providers. Then for each provider p_j, if $p_j.h$ is in the range of short history (i.e. $\leq$ 1.5 years), the value of $NI_i(s)$ will be incremented by 1. The invocation frequency of providers with a short history by user u_i is defined as:

$$IFH_i(s) = \frac{NI_i(s)}{NI_i} \tag{2}$$

$IFH_i(m)$ and $IFH_i(l)$ for medium and long histories can be calculated similarly. Among these three, we choose the one with the largest value and then compare it with

a threshold T_{IFH}. If it is above the threshold, the corresponding history range (short, medium, or long) is considered as user preferred. Otherwise, $u_i.PH$ will be "no preference". We could set the threshold as 50% to make sure this preference is considered only when a dominating pattern exists.

For other preferences in this group, the basic calculation steps are the same. However, the number of fuzzy values and their ranges might be different. We define *PH*, *PP*, *SH*, *SF*, *SP*, *SC*, *SD* to have 3 fuzzy values, and *SR*, *SA*, *SRT* to have 5.

After the implicit user modeling step, a user profile on his non-functional preferences is set up for every user.

4 Personalized Service Ranking

In our system, when a user submits a query, after the service discovery and matchmaking step, the personalized ranking component ranks all the matching services based on the user profile. The ranking step is essentially the similarity calculation between user's non-functional preferences and services' non-functional property values. Services with higher matching scores with the user's profile are ranked higher in the result list. To calculate the overall similarity score for the complete profile, we need to first define how to measure the similarity for individual properties. Again we divide them into two groups based on their data types.

In the first group, to calculate the similarity between a matching service s_k and a user u_i's non-functional preference on provider names, we use the following formula:

$$sim_{PN}(u_i, s_k) = \begin{cases} 1 & if\ u_i.PN = \emptyset \\ 1 & if\ s_k.pd.n \in u_i.PN \\ 0 & if\ s_k.pd.n \notin u_i.PN \end{cases} \quad (3)$$

When $u_i.PN$ is empty, it means the user has no preference on providers, any provider is considered as a match, and thus the similarity score is 1. When $u_i.PN$ is not empty, if $s_k.pd$ is in the user preferred list, the similarity score is 1, and otherwise, the score is 0. Similar steps are used for *SL*. In our current implementation, we only get Boolean similarity values. Later, if want to get a numeric similarity score, we could consider different degrees of preference on different providers.

To calculate the similarity on provider locations, because the location property has two sub-properties: continent and country, there are different scenarios we need to consider. When the service provider's country is one of user preferred countries, it is considered as a perfect match and the similarity score is 1. If only the continent matches, it is a partial match and the score is 0.5. The formula is as below:

$$sim_{PL}(u_i, s_k) = \begin{cases} 1 & if\ u_i.PL = \emptyset \\ 1 & if\ s_k.pd.cy \in u_i.PCY \\ 0.5 & if\ s_k.pd.ct \in u_i.PCT\ and\ s_k.pd.cy \notin u_i.PCY \\ 0 & if\ s_k.pd.ct \notin u_i.PCT\ and\ s_k.pd.cy \notin u_i.PCY \end{cases} \quad (4)$$

In the second group, if a user has no preference on the property, the similarity score is always 1. Otherwise, the following steps are taken. First, the user preferred fuzzy

value on the property is converted to a scale number. For instance, if the fuzzy values are short, medium, long, their corresponding scale numbers are 0, 1, and 2. Then, each matching service is assigned with a scale number on the property based on the range definition of the scales. For instance, if a service provider has been established for just one year, the provider history falls into the range for the fuzzy value "short", and its scale number is 0 (for short history). We adopt the method proposed in [14] to calculate the similarity between two scale numbers. For 3-scale preferences, we use the matrix below to get the similarity score. The left column represents user preferences and the top row represents service property values.

$$\begin{array}{cc} & s_k \\ & \begin{array}{ccc} 2 & 1 & 0 \end{array} \\ u_i \begin{array}{c} 2 \\ 1 \\ 0 \end{array} & \begin{bmatrix} 1 & 0.7 & 0 \\ 0.7 & 1 & 0.7 \\ 0 & 0.7 & 1 \end{bmatrix} \end{array} \tag{5}$$

For 5-scale preferences, we use a different matrix.

$$\begin{array}{cc} & s_k \\ & \begin{array}{ccccc} 4 & 3 & 2 & 1 & 0 \end{array} \\ u_i \begin{array}{c} 4 \\ 3 \\ 2 \\ 1 \\ 0 \end{array} & \begin{bmatrix} 1 & 0.7 & 0.2 & 0 & 0 \\ 0.7 & 1 & 0.3 & 0 & 0 \\ 0.2 & 0.3 & 1 & 0.3 & 0.2 \\ 0 & 0 & 0.3 & 1 & 0.7 \\ 0 & 0 & 0.2 & 0.7 & 1 \end{bmatrix} \end{array} \tag{6}$$

After we get the similarity scores for all the properties, we could combine them to get an overall score for the matching service using the formula as shown below:

$$sim(u_i, s_k) = \sum_{h=1}^{n} \alpha_h * sim_{a_h}(u_i, s_k) \tag{7}$$

where n is the number of non-functional preferences, a_h refers to the h-th preference, α_h is the coefficient on a_h, which measures how important the similarity score on the h-th property is to the overall score, the sum of the coefficients should be 1, and $sim_{ah}(u_i, s_k)$ measures the similarity of user u_i's profile and service s_k on the h-th property. In our current implementation, we use equal weights for all properties. Finally all the matching services can be ranked on their overall scores.

5 Experiments

5.1 Experiment Design

To the best of our knowledge, there is no publicly available dataset on service invocation histories. It is also difficult to collect an adequate amount of user data if we don't run a service selection system which is used by a lot of users. Due to these constraints, in our experiment, we didn't test the implicit user modeling component, and we only focused on the personalized ranking component. We developed a Windows-based prototype system using Java 1.6 with Eclipse IDE. We used MySQL Workbench 5.2 to build the provider and service repositories as well as the user profiles.

We wanted to demonstrate that for a registered user who has entered his personal preferences on non-functional service properties explicitly through the user interface, our system could generate a personalized ranking result for him. We also ran a simulator to generate user profiles with different combinations of preferences to test how different factors would affect the ranking accuracy.

Seekda is a publicly available web service search engine. It contains a good number of web services published online. It also maintains useful information of each service, such as its origin country, the provider information, a link to its WSDL file, tags, its availability, a chart of its response time in the past, a user rating, its level of documentation, etc. For most of the non-functional properties we consider in our system, we could find their values from either Seekda or the original hosting sites, except the provider popularity, the service popularity and the service cost. In the experiment, we excluded them from the similarity calculation.

We built our dataset by collecting web service data from Seekda during a six-month period (December, 2010 to May, 2011). There were 7739 providers and 28606 services stored in Seekda (as of August 2, 2011). Since some data were manually collected, such as the established time of a provider or the average response time of a service, we were not able to get all the services in Seekda. We followed a few different ways to get services: 1) for each continent, find some representative countries, and then for each country, find some representative providers, and get all the services published by these providers; 2) based on the tag cloud, choose tags with a large/medium/small number of matching services, and get all of these services; 3) follow the links provided by Seekda, get the most used services and the recently found services. After removing the services with expired URLs, we finally got 1208 services from 537 providers, and each provider contains at least one service. Since Seekda started crawling and monitoring web services from 2006, the oldest service in our dataset was published in 2006. We extracted the information we need and saved them into the provider repository and the service repository.

There are 287 service tags saved in our dataset, of which we chose 30 of them together with their matching services for our experiment. These tags are categorized into three groups based on the number of matching services. Group 1 has 10 tags and each tag has less than 50 matching services, group 2 has 10 tags with the number of services per tag between 50 and 100, and group 3 has 10 tags with more than 100 services per tag. In the experiment, each tag was used as a searching keyword to be submitted to our selection system. After the functional matching step, the relevant services were retrieved. Then the personalized ranking result could be generated based on the user profile.

We simulated 60 users. They are divided into 6 groups. For users in each group, they have preferences on a same number of properties (not necessarily same properties): 1, 2, 4, 6, 8, and 10 respectively, and for the rest of the properties, they have no preference on their values. For instance, one group may have preferences on 2 non-functional properties, and no preferences on the others. Users in this group could have preferences on different non-functional properties. User *A* from this group has preferences on service rating and service languages, and user *B* from the same group has preferences on provider names and provider history.

5.2 User Preference Forms and An Illustrating Example

In order to solicit user preferences, after a user registers to our system, he is required to specify his preferences on various non-functional properties. He could also update his profile later when necessary. There are two forms a user needs to fill. Figure 2 shows the first one. QoS related properties such as *SA*, *SRT* are included in another form, which is not shown in the paper due to the page limitation.

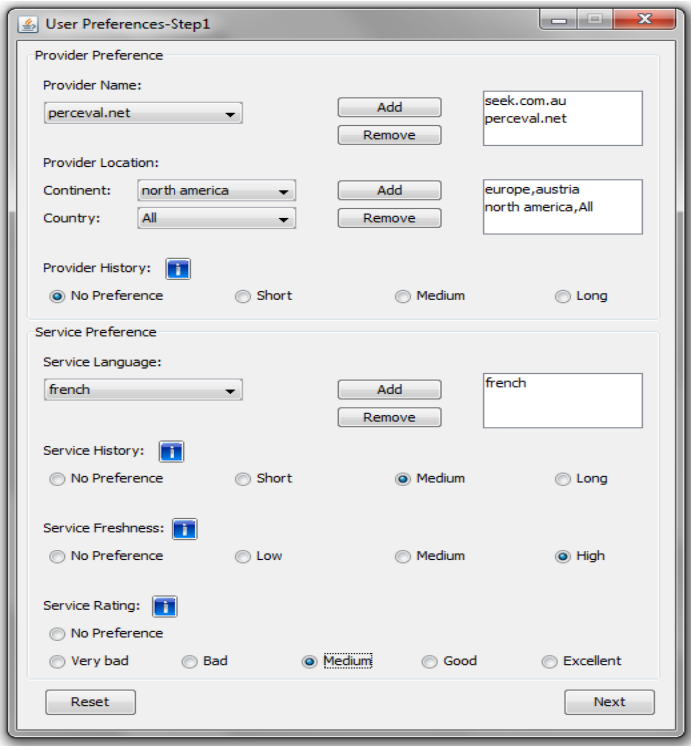

Fig. 2. User Preference Form 1

For properties with fuzzy values (e.g. provider history), there are information icons beside them. When a user clicks on an icon, the value range of the property will be displayed. Table 2 lists value ranges for all of these properties. Our system could handle multiple scales. But for the simplicity reason, we only use two scales here: 3 and 5, which are the common choices for fuzzy values. For history related properties, we use 3 scales because most services in our dataset are relatively new and the value span is small. For other properties, we use 5 scales.

After the user fills the forms, his profile will be built. Assume a user named *Dave* has the following profile: (*PH*: long, *SD*: good, *SA*: excellent). When *Dave* submits a query "Finance" to our system, the personalized ranking result is generated based on this profile. Table 3 shows the top 10 results, together with their original rankings and their related non-functional property values.

Table 2. Properties and their value ranges

Property	Ranges of its fuzzy values
PH (year)	short: <=1.5, medium: >1.5 and <=3, long: >3
SH (year)	short: <=1.5, medium: >1.5 and <=3, long: >3
SF (month)	low: <=6, medium: >6 and <=12, high: >12
SR (0~5)	very bad: <=1, bad: >1 and <=2.5, medium: >2.5 and <=3.5, good: >3.5 and <=4.5, excellent: >4.5
SA (%)	very bad: <=50, bad: >50 and <=70, medium: >70 and <=80, good: >80 and <=95, excellent: >95
SRT (ms)	very bad: >=790, bad: <790 and >=770, medium: <770 and >=750, good: <750 and >=700, excellent: <700

Table 3. Top 10 services with their original rankings and property values

Rank	Service Name	PH	SD	SA	Original Rank
1	DelayedStockQuote	4.30	good	99.50	3
2	ForeignExchangeRates	4.10	good	99.70	4
3	XigniteQuotes	4.30	good	99.81	8
4	XigniteHelp	4.09	good	99.73	17
5	LeadStatusQuery	3.41	good	98.63	18
6	AffiliateServices	3.17	good	99.25	22
7	XigniteGlobalQuotes	2.83	good	99.67	6
8	XigniteMoneyMarkets	2.83	good	99.38	12
9	XigniteFundHoldings	2.83	good	99.38	15
10	FinanceService	2.41	good	97.26	23

By checking the results, we can see that services satisfying user preferred non-functional values are promoted to higher positions. For instance, the original rankings for the 3rd and 4th services are 8 and 17, but since they have a long provider history, a good documentation and an excellent availability, their positions are promoted.

5.3 Factors Influencing the Result Accuracy

Since to the best of our knowledge, no similar work has been done before, in this experiment, we focus on evaluating the degree of improvement from the personalized ranking, as well as the impacts of different factors on this improvement. We mainly tested two factors: the number of functionally matching services, and the number of non-functional properties on which users have defined their preferences. According to a study on web users' searching behavior [7], the ranking order is really important because users normally only check the top *N* results (a common value for *N* is 10). So in this paper we use the ranking improvement to measure the system benefit. We take the top 10 results from the personalized ranking list, check their non-functional property values making sure they satisfy user preferred values (if not, remove them), and then check the ranking positions of these services in the original ranking list. If they are ranked very high in the original list, it means our algorithm does not improve the ranking very much. If they are originally ranked very low, it means our algorithm can promote these services to better positions, and therefore our personalized ranking

is beneficial. Here we measure the benefit using the Mean Average Precision (MAP) [7]. We take the personalized ranking results as the reference and check the precision of the original results. A lower value means a bigger loss in accuracy and thus a higher benefit. The following formula is used for each query q:

$$MAP_q = \frac{\sum_{i=1}^{R}(P(i)*rel(i))}{\min(R,10)} \tag{8}$$

where R is the number of functionally matching services, $P(i)$ measures the precision in the i-th position [7], $rel(i)$ is 1 if the i-th result is one of the top 10 services in the personalized result and 0 otherwise, $min(.)$ is to get a lower value between R and 10.

For each of the 60 users we simulated, we also generated some random service requests. Each user submitted 30 queries using 30 selected tags. Then we measured the MAP value for each query. Here we chose 3 tags from each group to discuss the results. The 9 selected tags are *charter flight*, *telecommunication*, *traffic*, *travel*, *government*, *finance*, *bioinformatics*, *tourism*, and *university*. Their corresponding numbers of services are 7, 21, 38, 70, 76, 97, 120, 140 and 169 respectively.

Figure 3 shows the average MAP values for these 9 keywords. The result is averaged on all 60 users. Since we listed the keywords in the ascending order of their numbers of matching services, from this figure, we can tell the relationship between the number of services and the MAP value. Initially when there are only 7 services (for *charter flight*), its MAP value is over 0.8, which shows the personalized ranking gets similar results as the original one and the benefit is not very obvious. When the number of services gets bigger, the benefit becomes higher. The slope is sharp when the number of services is below 50, and becomes fairly even after 50. The average MAP value when the number of services is above 50 is 0.11, which means that many of the top results from the personalized ranking are not in the original top list. It clearly shows the importance of the user modeling and the personalization, especially when there are many functionally matching services.

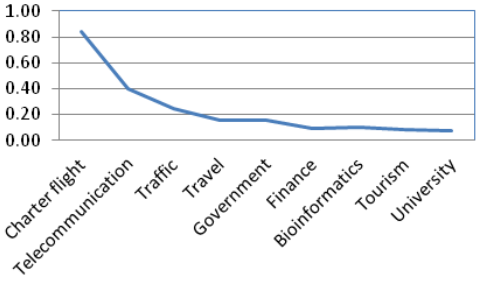

Fig. 3. Average MAP for 9 queries

Figure 4 shows the average MAP values for users with different numbers of preferences. The result is averaged on 10 users in each group and on all the 9 queries. In this figure, we could see that when users have preferences on more non-functional properties, the MAP value is getting smaller. So the personalized ranking could help more when users have more preferences. The curve here is not very sharp compared

to Figure 3. It shows that the impact from the number of preferences is less than that from the number of services. When the number of preferences is above 4, there is no big difference in the result.

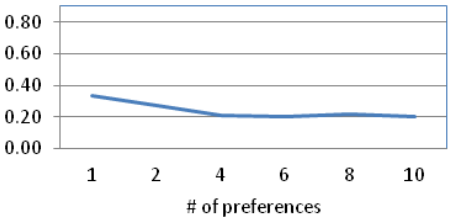

Fig. 4. Average MAP for different number of preferences

In general, based on these experiment results, we could conclude that the personalized ranking could benefit the service selection system to satisfy individual requirements, especially when there are a lot of functionally similar services and users have preferences on more than one non-functional property.

6 Related Work

Personalized service ranking or service recommendation based on user preferences has attracted research attention in recent years. In [11], previous interactions between service providers and requestors were modeled as a social network, and then results from the social network analysis were fed into a Bayesian classifier to rank services. In [9], personal profiles were built using the collaborative filtering technique to find similar users based on their invocation histories and the association rule mining to identify service dependencies based on the past composition transactions of these similar users. User similarity in [16] was measured by the similarity between the rankings of their observed QoS values on commonly invoked services, and then the personalization was implemented using the past experiences from similar users. The work in [2] collected the service invocation data and used it for the later discovery. However it only focused on the functional query part. In the above work, QoS data was included in the ranking process in [11] [16], but not the other two. All of them used the past usage data, mainly invocation histories to get the personalized results.

There are also systems relying on the explicit user ratings for the service ranking. Item-based collaborative filtering was used in [6] to predict the user rating on a service and the services were then ranked based on the predicted ratings. In [1], the score of a service was calculated based on ratings on this service from similar users and ratings on similar queries. In [14], service selection was based on both objective and subjective QoS values. A fuzzy inference system can handle the objective QoS factors (e.g. latency), and the collaborative filtering was used on subjective QoS factors (e.g. user ratings). The approach proposed in [5] was based on user experiences on given services, as well as their preferences, which provided contexts for the experiences and

made the personalization more accurate. Personalization in [13] was achieved by considering different user factors, such as user preferences on different QoS properties, user-defined comparisons on QoS values, and user's confidence level on the reputation mechanism used in the selection process.

Compared to these papers, our paper is unique because it emphasizes on building user models to capture their long-term non-functional preferences. Some papers also built user profiles. However, their profiles only considered users' functional interests [2], or the invocation frequencies on services [9]. The profile in [13] was a little similar to ours. But it puts too much workload on users by asking them to compare values for each QoS property. In our work, users only need to pick a fuzzy value or choose from a list of possible values, and we also have a more thorough definition on non-functional preferences and an implicit user data collection mechanism.

Non-functional preferences have been used for service selection and ranking in some work, e.g. [3] [10]. In [3], an expressive and user-friendly preference model was proposed, which considers both qualitative and quantitative preferences, as well as atomic and composite preferences. In [10], qualitative preferences can be defined using the TCP network, which can handle the conditional preferences and relative importance of various non-functional properties, and quantitative preferences can be defined as utilities using the UCP network. Currently our paper only considers 2 types of user preferences and does not consider the relative importance of different properties. In the future, we could extend our work based on [3] and [10].

7 Conclusions

In this paper, we proposed a user modeling approach to build user profiles on their non-functional preferences. Then a personalized ranking algorithm could be applied to the service selection process to identify user desired services. In the user modeling stage, the system could collect the user data explicitly through user forms, or implicitly by analyzing the past usage data. In the selection stage, after the functionally matching services are identified for the user query, they are ranked based on how well they could match with the user preferred values on the specified non-functional properties. Experiment results showed that the personalized ranking has largely improved the result accuracy for individual users.

There are a few directions we would like to work on in the future. First, we could set up a service community such as the one in [2] to collect the real usage data in order to evaluate our implicit user modeling algorithm. We would also like to conduct a user study to evaluate our system. Second, we would like to implement and test other more complex profiling and similarity calculation algorithms to see whether we could further improve the system performance.

Acknowledgements. This work is partially sponsored by Natural Science and Engineering Research Council of Canada (grant 299021-2010).

References

1. Averbakh, A., Krause, D., Skoutas, D.: Recommend me a Service: Personalized Semantic Web Service Matchmaking. In: Proceedings of the 17th Workshop on Adaptivity and User Modeling in Interactive Systems (2009)
2. Birukou, A., Blanzieri, E., D'Andrea, V., Giorgini, P., Kokash, N.: Improving Web Service Discovery with Usage Data. IEEE Software 24(6), 47–54 (2007)
3. García, J.M., Ruiz, D., Ruiz-Cortés, A.: A Model of User Preferences for Semantic Services Discovery and Ranking. In: Aroyo, L., Antoniou, G., Hyvönen, E., ten Teije, A., Stuckenschmidt, H., Cabral, L., Tudorache, T. (eds.) ESWC 2010, Part II. LNCS, vol. 6089, pp. 1–14. Springer, Heidelberg (2010)
4. Howard, J.A., Sheth, J.N.: The Theory of Buyer Behavior. John Wiley & Sons (1969)
5. Klan, F., König-Ries, B.: A Personalized Approach to Experience-Aware Service Ranking and Selection. In: Greco, S., Lukasiewicz, T. (eds.) SUM 2008. LNCS (LNAI), vol. 5291, pp. 270–283. Springer, Heidelberg (2008)
6. Manikrao, U.S., Prabhakar, T.V.: Dynamic Selection of Web Services with Recommendation System. In: Proceedings of the International Conference on Next Generation Web Services Practices, pp. 117–121 (2005)
7. Manning, C.D., Raghavan, P., Schütze, H.: Introduction to Information Retrieval. Cambridge University Press (2008)
8. Qiu, F., Cho, J.: Automatic Identification of User Interest for Personalized Search. In: Proceedings of the 15th International Conference on World Wide Web, pp. 727–736 (2006)
9. Rong, W., Liu, K., Liang, L.: Personalized Web Service Ranking via User Group combining Association Rule. In: Proceedings of the 7th International Conference on Web Services, pp. 445–452 (2009)
10. Schröpfer, C., Binshtok, M., Shimony, S.E., Dayan, A., Brafman, R., Offermann, P., Holschke, O.: Introducing Preferences over NFPs into Service Selection in SOA. In: Di Nitto, E., Ripeanu, M. (eds.) ICSOC 2007. LNCS, vol. 4907, pp. 68–79. Springer, Heidelberg (2009)
11. Shafiq, M.O., Alhajj, R., Rokne, J.: On the Social Aspects of Personalized Ranking for Web Services. In: Proceedings of the IEEE International Conference on High Performance Computing and Communications, pp. 86–93 (2011)
12. Tran, V.X., Tsuji, H., Masuda, R.: A New QoS Ontology and its QoS-based Ranking Algorithm for Web Services. Simulation Modeling Practice and Theory 17(8), 1378–1398 (2009)
13. Vu, L.H., Proto, F., Aberer, K., Hauswirth, M.: An Extensible and Personalzied Approach to QoS-enabled Service Discovery. In: Proceedings of the 11th International Database Engineering and Applications Symposium, pp. 37–45 (2007)
14. Wang, H.C., Lee, C.S., Ho, T.H.: Combining Subjective and Objective QoS Factors for Personalized Web Service Selection. Expert Systems with Applications 32(2), 571–584 (2007)
15. Zhang, Q., Ding, C., Chi, C.H.: Collaborative Filtering Based Service Ranking using Invocation Histories. In: Proceedings of the IEEE Internation Conference on Web Services, pp. 195–202 (2011)
16. Zheng, Z., Zhang, Y., Lyu, M.R.: CloudRank: A QoS-Driven Component Ranking Framework for Cloud Computing. In: Proceedings of the 29th IEEE International Symposium on Reliable Distributed Systems, pp. 184–193 (2010)