A Personalized Assistant Framework for Service Recommendation

Pradeep K. Venkatesh*, Shaohua Wang*, Ying Zou*, and Joanna W. Ng†
*Department of Electrical and Computer Engineering, Queen's University, Canada
†IBM Watson IoT Division, IBM Canada Ltd., Canada
15pkv@queensu.ca, shaohua@cs.queensu.ca, ying.zou@queensu.ca, jwng@ca.ibm.com

***Abstract*—The popularity of service-oriented computing makes more and more services available on the Web. Users make use of these services to achieve their personal goals, such as purchasing movie tickets on-line and booking flights. Existing research has proposed various techniques to assist users to select services for achieving user goals. Typically, user choice of services change under different contexts. However, these approaches cannot recommend the desired services based on the changes of user contexts, and are not able to learn from user service selection history. In this paper, we provide an intellectually cognitive personalized assistant framework to achieve user goals. In particular, considering user contexts and historical service selection, our framework interacts with users by asking relevant and necessary questions, and help users navigate through sets of services to identify the desired services. We have designed and developed a prototype as a proof of concept. We perform a case study to evaluate the effectiveness of our framework. On average, our framework, utilizing the learning-to-rank algorithm, namely AdaRank, improves the nine baseline approaches by 12.02%–31.52% in helping users find the desired services. Our user study results show that our framework is helpful in achieving user goals and useful in saving users' time in finding their personalized services faster.**

***Keywords*-knowledge creation; natural language processing; web tasking; user intent; personal assistance; service discovery.**

I. Introduction

Nowadays, web services are getting extremely popular among users with the maturation of service-oriented computing. For example, the ProgrammableWeb[1], a popular on-line web services repository, alone indexes 16,521 web services as of $Jan\ 31st$, 2017. More importantly, web services play a vital role in every aspect of our lives, such as checking the weather forecast, finding stock prices, and purchasing products on-line. To accomplish daily goals online, users need to perform the following steps in orders: breaking down a high-level goal into concrete tasks, identifying the service for accomplishing the concrete tasks from huge volume of web services available on the Web, and invoking the identified service. The process of accomplishing a daily goal can be very tedious, as the above steps, quite often, require the manual involvement from users, such as finding the suitable service to carry out the user goal [26, 32].

Extensive research has been conducted to help users select desired services (*e.g.*, [1, 7, 25, 31]). For example, some research (*e.g.*, [1, 25]) proposes filtering-based approaches to help users navigate through a relatively huge volume of services to select relevant services. Some other research (*e.g.*, [7, 31]) develop various rule-based service recommendation applications based on users contextual information. However, the aforementioned research primarily has the following two drawbacks:

- **Lack of context-aware interactions with users in the process of service selection.** User contexts and decisions can change very often in service selection. When selecting a set of services, a user's choice on a service (*i.e.*, select or not) can change under different contexts, and it can affect the users' choices on next subsequent services. Moreover, a user goal is usually abstract and lack of detailed information for achieving the goal. For example, a goal "find an accommodation in Toronto" has no information of types of hotels (*e.g.*, luxury or budget). Therefore, context-aware interactions with users are critical to identify users' needs. However, existing approaches limit the user involvement in service selection and cannot identify the swift changes of user contexts and decisions. It often leads to not finding the most appropriate services for users and causes frustration.
- **Lack of automatically learning from users' history of service selection.** To achieve specific goals, users often select and invoke particular services repeatedly over a period of time. For example, a user may always opt for services related to luxury or budget hotels when booking the accommodation. It is crucial to automatically analyze and have personalized service recommendation based on the user's history to save their time.

In this paper, we propose a personalized assistance framework using AdaRank, a learning-to-rank machine learning algorithm, to address the aforementioned limitations. Our framework interacts with users to improve user experience by asking them the most relevant and necessary questions about their intended goals. More specially, we analyze user contexts and historical service selection to propose 21 learning features and apply AdaRank with the proposed learning features to recommend relevant and necessary questions. The questions help narrow down the search space of identifying

[1]http://www.programmableweb.com/

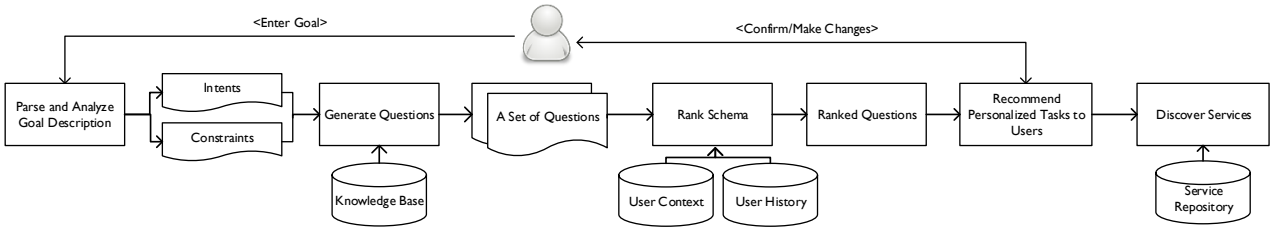

Figure 1: An overview of our personalized assistance framework.

the most preferred services. The number and sequences of the asked questions differ based on the user contexts and previous service selections.

We have implemented a prototype of our framework as a proof of concept. We evaluate the effectiveness of our personalized assistance framework through empirical experiments and a user case study. Our personalized assistance framework utilizing AdaRank achieves a precision at 1 of 61.22% and improves the other nine approaches by 12.02%–35.51% in finding the most relevant questions. Furthermore, the results of our user case study show that our framework is helpful in finding the most appropriate services for user goals and saves users time by recommending the personalized services to users.

Paper organization. Section II describes our proposed approach. Section III shows our empirical studies. Section IV summarizes the related work. Finally, we conclude and provide directions for future work in Section V.

II. Our Personalized Assistance Framework

In this section, we present our proposed personalized assistance framework. First, we briefly introduce the general steps of using our framework to find desired services. Second, we present the major components of our framework and their interactions. Figure 1 illustrates the overall approach of our framework and the interactions among its components.

More specifically, a user issues a user goal in natural languages to our personalized assistance framework.

Our framework analyzes the entered goal to understand the user intents and constraints. For example, given a user goal: "I want to find accommodation in Toronto", our framework can analyze the goal and extract the user intents, "find accommodation" and constraints "{location, Toronto}".

With the learned intents and constraints, our framework asks users the personalized, necessary, and relevant questions to guide them to find the desired services. Our framework uses AdaRank, a learning-to-rank (LtR) algorithm, to ask questions to users by taking into consideration *user contexts, historical user choices on service selection, and the process knowledge mined from the Web for achieving the goal*. For a question, the user can decide to answer it or not, and the user's choice affects the recommendation of next questions. Through interacting with the user by asking questions, our framework narrows down to the desired web services for achieving the entered user goal. Figure 2 briefly illustrates the overall steps of our framework for achieving the goal "I want to find accommodation".

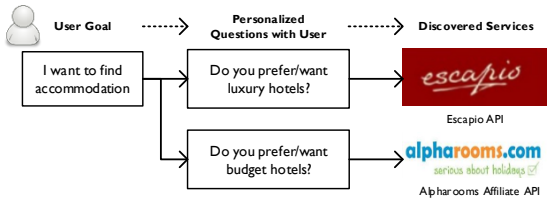

Figure 2: An example overview of our framework.

A. Creating Process Knowledge Base

No existing process knowledge base in public domains is available to help achieve their goals. However, the publicly available on-line resources act as a rich textual source of process knowledge for users. In general, a user needs to delve into details of a goal (*e.g.*, types of accommodation, such as luxury or budget hotel) to achieve a goal. To obtain the process knowledge for accomplishing the detailed steps of a goal, our framework creates machine-understandable process knowledge base containing detailed steps by analyzing the publicly available websites.

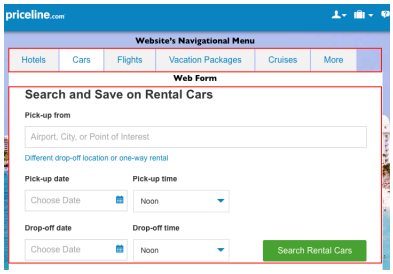

Figure 3: An annotated website structure from priceline.com.

On-line websites. contain domain knowledge of carrying out business operations that help users complete their goals. A website can contain a collection of web pages. A web page can have a web form to collect user inputs for performing a

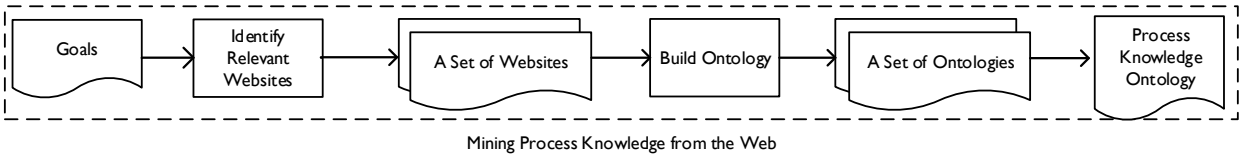

Figure 4: An overall process of creating knowledge base.

user goal, *e.g.*, reserving rental cars, and a menu to help users navigate through different web pages, *i.e.*, index of various user goals. For example, Figure 3 shows an annotated web page of priceline.com. The page contains a web form and a navigational menu. The navigational menu contains a group of menu items in which each menu item links to a different web page where a user can perform a goal. Each menu item contains a label showing the name of the menu item and a URL for linking the menu item to a web page. Basically, a group of menu items in navigational menu dictate the set of steps a user has to perform to complete a goal. Therefore, the process knowledge can be captured from the navigational menus and web forms of websites.

Mining process knowledge from the Web. Normally, a website contains web pages that help users accomplish goals, and acts as the main source of concrete process knowledge of a business process [29]. We use the following steps as shown in Figure 4 to create the process knowledge base: First, we automatically search for top 10 websites for a given goal using Google Custom Search API[2] as suggested by [29]. Second, to capture the concrete process knowledge for the given goal, we utilize the approach in [29]. The approach automatically retrieves the process knowledge from on-line websites and stores it in an ontology for easier machine readability. In particular, the approach builds a knowledge ontology from each website, and merge the built knowledge ontologies from multiple websites to one complete process knowledge ontology. Last, we store the process knowledge ontology into the repository for the given goal. Each user goal maps to its corresponding process knowledge ontology.

B. Analyzing and Parsing User Goals

Our personalized assistance framework analyzes the goal to identify user intents and constraints using the approach proposed by Zhao *et al.* [32]. The approach utilizes the start-of-the-art Stanford natural language processing techniques[3] to analyze the user goal and to extract the user intent and constraints. For example, given a user goal: I want to purchase a house in Toronto, the approach outputs the intents and constraint that are {purchase house} and {location, Toronto}, respectively.

C. Recommending Personalized Tasks to Users

When a user specifies a goal, the goal description can be high-level and lacks of detailed information that is needed for achieving the goal. For example, a goal "I want to find accommodation for my trip to Toronto" does not have any detailed information of types of accommodation, such as budget or luxury hotel. Even more, user's choices on selecting services can change under different contexts. Therefore, to help users identify the desired services accurately and efficiently, we need to obtain the detailed information of a user goal quickly and accurately. In this paper, to improve the user experience, we engage with users through dialogs to understand the user goals better and execute the required services. Each dialog has a question.

Next, we delve into the details of dialog establishment and question recommendation.

Interacting with users through dialogs. To quickly obtain the detailed information of a user goal, we need ask users the most relevant and necessary questions. Initially, we search and identify the required process knowledge ontology for a given user goal from a set of our stored ontologies in Section II-A. To accomplish a user goal, multiple paths of traversing nodes in the ontology can exist. We need identify a proper path and even switch to a different path based on the changes of user contexts and choices. Then, we traverse the selected path and convert each node in the path to a question. We convert the node to a question using a predefined template: Do you prefer/want <node name>? For example, finding accommodation path leads to two potential nodes (*i.e.*,luxury hotel and budget hotel). The converted questions is "Do you prefer/want luxury hotel?" and "Do you prefer/want budget hotel?".

Recommending Questions. We propose a learning-to-rank based approach to learn the changes of user contexts and choices on service selection, and more importantly recommend the relevant questions to users. If a user agrees to answer a question within a path, we continue to move on to the other questions within the path. Otherwise, we proceed to the next available path. We store each node of path visited by the user in the selected process knowledge ontology.

If a user is not satisfied with all the possible questions offered from our mined process knowledge, we let the user enter his or her own choice. We calculate the textual similarity between the user entered choice and all the possible questions available in the process knowledge ontologies using TextGraph, a technique for measuring the textual similarity between two short phrases taking into consideration of the sentence semantics [21]. If any questions have the similarity scores higher than 90%, we rank them in a descending order

[2]https://developers.google.com/custom-search/

[3]http://nlp.stanford.edu/software/

Table I: List of learning features available to be used in learning techniques.

Category	Sub Category	Description	List of 21 Learning Features
User Execution History (3)	1). Vertex Passes (1)	Number of times a user selects a certain action in the execution sequence.	P_{v_i}: Number of passes at vertexes
	2). Service Pick (1)	Number of services a user selects while performing the same goals.	P_s: Number of services
	3). Execution Sequence (1)	Number of questions answered by the user to attain a specific goal.	P_l: Length of the execution paths
User Contexts (15)	4). Time (4)	Users tend to have behavioral habits that they perform online tasks at a certain time of the day.	T_m: Morning; T_a: Afternoon; T_e: Evening; T_n: Night
	5). Date (2)	Record which day of the week to learn user behavioral habits.	D_{wd}: Weekdays; D_{we}: Weekends
	6). Location (6)	Places where users tries to achieve goals.	L_c: Country; L_p: Province; L_{ci}: City; L_w: Work; L_l: Lab; L_h: Home
	7). Devices (3)	Devices users use to perform their online activities, such as achieving goals.	D_l: Laptop; D_m: Mobile; D_t: Tablet
Service Contexts (3)	8). Usage (1)	Frequency of the same service execution.	S_u: Number of times service used
	9). Score (1)	Functionality score calculated based on service relevance to the goal.	S_{u+1}: Computed Functionality score
	10). Non-Functional (1)	Non-functionality attributes of the service, such as quality of service.	S_{u+2}: Non-functionality scores

(#): Denotes the number of learning features in each learning feature category.

based on the similarity scores and offer these questions as alternate questions. If the user accepts an alternate question, we proceed on the alternate question's path route. If the user is not satisfied with any alternate questions, our framework cannot help the user with the learned process knowledge.

Learning Features for our learning approach. Through the interactions with users, the sequences of user choices on selecting questions and services in achieving user goals can be obtained. Learning from previous user choices can help users achieve their goals with a minimal number of interactions with our approach. To capture user specific needs, we build three categories of 21 learning features as shown in Table I that can be used by our learning-to-rank approach for minimizing the interactions with users by recommending the most suitable questions:

[C1.] User Execution History. Early Human Computer Interaction research (HCI) shows that users use a small number of commands repeatedly (*i.e.*, history of reuse) in command line interfaces [12]. Later, Lee et al. [13] present approaches of different ways in which user interaction history could be used. We build 3 learning features to reuse users' history to learn user preferences.

[C2.] User Contexts. Users perform certain goals on the fly as a part of their daily activities, such as "checking weather for the day" when at home and "finding favorite restaurant" whenever visiting a new city. Information changes in every individual context over time. The context knowledge can provide us with some hints of conceivable information a user tries to perform at a particular context [23]. Therefore, we build 15 learning features to capture the changes in individual contexts.

[C3.] Service Contexts. Users typically select and execute services to finish goals. Each of the selected services have non-functional attributes, such as Quality of Service. We capture the non-functional attributes, and the service usage history to identify user preferences. We build 3 learning features based on service context.

Our learning approach using AdaRank, a learning-to-rank algorithm, use the built learning features to recommend users with the most relevant questions. AdaRank is a boosting technique which can repeatedly construct weak rankers by re-weighting the training data and finally linearly combines the weak rankers to form a strong ranking function to directly yield higher performance measures [30].

To capture users' choices on service selection, we build a *Logical Graph* (L_{Graph}) for modeling the transitions among questions. L_{Graph} is a directed acyclic graph where a vertex represents a question and an edge is directed from one question to another. Each edge records a transition from one question to another with user contexts. Our framework uses AdaRank to compute scores at each outgoing edge from a vertex to identify the highest scoring outgoing edge as the one to recommended question to the user. Figure 5 shows an example of the transitions in the Logical Graph.

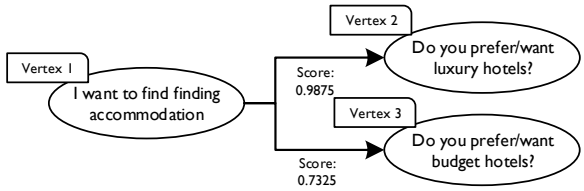

Figure 5: An example representation in Logical Graph.

D. Discovering Services

Once a user answers a path of questions for a user goal, we discover and select the required services for the user goal. In a service repository, a service has a service name and description, and a set of input and output parameters.

To make the service search meaningful, we normalize the words within service description in three steps:

1) Remove stop words in English, such as "a", "is", and "the", as these words do not facilitate meaning [17].
2) Use WordNet[4] to remove all non-English words.
3) Apply the Porter stemming to map words to their base form [22]. *e.g.*, "shopping", and "shopper" are reduced to their base form "shop".

We use Lucene[5] (*i.e.*,text-based search engine) to search for services from the service repository. Lucene is a scalable information retrieval library using two theoretical models (*i.e.*, Boolean Model and Vector Space Model) to perform the search for a given query [18]. We take the <node name> from all the answered questions in the Logical Graph and the user goal as a query to search for the desired services. For each service, Lucene produces a relevance score denoting the relevance of a service to the query. In general, a query can contain at least a few words (*e.g.*,find accommodation). To increase the chance of identifying relevant services for a given query, we expand the given query to include semantically related words. We use the ConceptNet to expand a query. ConceptNet [14] is a popular semantic database which connects concepts (*i.e.*,words) with one another using semantic network concepts. ConceptNet identifies 21 categories of semantic relations between concepts, such as "IsA", "UsedFor", and "PartOf" [24]. We choose the three popular categories of semantic relations between concepts to expand the search string:

(R1) Part denotes a "Part Of" relation among concepts (*i.e.*,words). *e.g.*,morning is a part of day.

(R2) HasA represents a concept that is superior to the other concept. *e.g.*,cars have four wheels.

(R3) IsA shows a concept that is a particular instance of another concept. *e.g.*,year with 366 days is a leap year.

We include all the retrieved concepts from the ConceptNet in a search query to discover services. The results are represented as a weighted vector based on similarity score for each concept. We compute an overall score to find the relevant services for user goals: $Overall_{Score} = G_{Score} + C_{Score}$, where the G_{Score} is the sum of similarity scores between all the answered questions combined of a goal and the overall description of the service, and C_{Score} is the similarity score between the user goal defined constraint and the overall description of the service. Finally, we rank the services based on the $Overall_{Score}$ and select the top ranked service for the user goals.

III. Case Study

In this section, we introduce our case study that evaluates our framework. We first describe the setup of our case study. Second, we present the evaluation results of our framework.

[4]https://wordnet.princeton.edu/

[5]http://lucene.apache.org/

Table II: The statistics of the collected services.

No.	Domain	SOAP-based Services	RESTful Services	Total
1.	E-commerce	186	165	351
2.	Travel	414	181	595

A. Case Study Setup

Creating Knowledge Base. We construct the process knowledge ontologies from various online websites for 12 user goals in two different domains *i.e.*, 1). E-commerce (*e.g.*, how to buy clothes online) and 2). Travel (*e.g.*, how to plan a vacation on a budget).

Creating Service Repository. We create a service repository with two types of services, SOAP-based services and RESTful services. We download 600 publicly available WSDL files for SOAP-based services. We use a popular online Web API repository (*i.e.*, ProgrammableWeb[6]) to manually collect 346 RESTful services. The collected services in our service repository are from two distinct domains: 1). E-commerce and 2). Travel. Table II shows the detailed statistics of our collected services.

User Study. We conduct a user study to evaluate the effectiveness of our framework. We sent out the invitation requests to 20 subjects, and 12 out of 20 subjects agreed to participate in our user study. Our participants are from different groups (*i.e.*,7 graduate students and 5 working professionals). All of the participants are familiar with performing online tasks and usually spend 8-10 hours online on a daily basis. For each participant, we provide a face-to-face tutorial on the background knowledge and the steps involved to complete the user study. Each participation tutorial session lasts about 20 minutes. As part of our user study, participants use our developed tool. Our tool lets users sign in, enters their goals using natural languages, selects their appropriate questions, and displays the desired service. Each participant used our tool for one month. At the end of the user study, each participant submitted a feedback questionnaire. All our user study participants were paid after completion.

B. Research Questions

RQ1. What is the performance of our personalized assistance framework?

Motivation. Our personalized assistance framework utilizes AdaRank, a learning-to-rank (LtR) machine learned ranking algorithm, to recommend questions which help users discover their desired services. It is critical to evaluate the performance of our framework with regards to recommend questions which aid users to discover their desired services.

Approach. To test the performance of our framework, we build 9 baselines that are listed as follows:

1. *Random Selection*: We randomly select the possible transitions for any given vertex using the Logical Graph.

[6]http://www.programmableweb.com/category/all/apis

2. *Logical Graph Ranking*: We utilize the Logical Graph in Section II-C to calculate the scores among vertexes using the computed features weights from the collected user study data. We compute scores at each outgoing edge from a vertex to identify the highest scoring outgoing edge using only the learning features of service context. The scores are calculated using the Equation 1.

$$Score_{(V_i,V_j)} = \sum_{u=1}^{n} SC(S_{u...n}) \quad (1)$$

We normalize the obtained score into the range of [0 – 1]. In Equation 1, where V_i *and* V_j are the origin and end vertexes of a given edge; and SC is service context values obtained from the user history at the given edge which help navigate the user from one vertex to the other (*i.e.*,V_i *to* V_j).

3. *PageRank*: a popular graph-based algorithm that integrates the impact of both incoming and outgoing edges into one single ranker model [20].

4. *ListNet*: is a listwise LtR technique using a neural network approach with gradient descent to minimize a loss in the ranking function [6].

5. *RankNet*: is a pairwise LtR approach using neural network with gradient descent steps by controlling the learning rate at each run of the ranker [5]. ListNet differs from RankNet by using a listwise technique instead of pairwise.

6. *MART*: is a LtR boosting technique using gradient descent to create a ranking function [11].

7. *LambdaMART*: is an ensemble LtR method consisting of boosted regression trees in a combination with LamdaRank [28]. LambaRank is a neural network algorithm similar to RankNet algorithm [5].

8. *RankBoost*: is a pairwise LtR technique that aims to select a set of weak rankers to build a strong ranker function [10].

9. *Coordinate Ascent*: is a listwise LtR approach optimizing rankers by minimizing a specific loss function [19].

We apply our approach and the above baselines on our collected user data. To evaluate the performance of the above algorithms, we compute the precision and recall using Equation (2) and Equation (3), respectively.

$$Precision@k = \frac{\{\#\ of\ answered\ questions\}}{K} \quad (2)$$

$$Recall@k = \frac{\{\#\ of\ answered\ questions\}}{\{\#\ of\ questions\ should\ be\ answered\}} \quad (3)$$

where K is the number of recommended questions. In this paper, we set K = 1 and 2, as we aim to recommend the right questions within a very short list. Each time, a user can only select one question. Therefore, "# of answered questions" and "# of questions should be answered" can only be 1.

Results. Our personalized assistance framework with AdaRank outperforms all the baselines in recommending the relevant questions for users. On average, AdaRank achieves a precision@1 of 61.22% and outperforms the 9 baselines by 12.02%–31.52% on the precision@1. In most cases, our approach can recommend the relevant question to users. Furthermore, our approach achieves a recall@2 of 77.51%, meaning that in most cases, one of the top two recommended by our approach is the relevant question to users.

Table III: Evaluation results of the ten algorithms.

	Algorithm	P@1	P@2	R@1	R@2
	Our Framework				
	AdaRank	61.22%	38.75%	61.22%	77.51%
	Baseline Approaches				
1	Random Selection	29.70%	28.80%	29.70%	57.59%
2	Logical Graph	34.56%	31.66%	34.56%	63.31%
3	PageRank	36.80%	35.31%	36.80%	70.63%
4	ListNet	39.48%	30.47%	39.48%	60.94%
5	RankNet	40.40%	31.27%	40.40%	62.55%
6	MART	43.19%	27.57%	43.19%	55.13%
7	LambdaMART	45.52%	31.92%	45.52%	63.83%
8	RankBoost	47.89%	34.75%	47.89%	69.50%
9	Coordinate Ascent	49.20%	33.31%	49.20%	66.62%

P@k and R@k denote the precision and recall when k=1 and 2

Among the baselines, generally, the learning-to-rank algorithms outperform the graph-based algorithms and random selection at P@1. However, the graph-based algorithms, PageRank and Logical Graph, achieve a higher R@2 than other baselines. It means that within the 2 top recommended questions, one of them is relevant to users.

AdaRank, a learning-to-rank algorithm, performs the best in recommending relevant questions to users.

RQ2. What are the important learning features?

Motivation. To better optimize the performance of our approach, it is crucial to learn the most important learning features from the user execution sequences for helping increase the determining power of the AdaRank.

Approach. To learn the importance of a learning feature, we run the AdaRank algorithm by removing the learning feature and evaluate the performance using Equation (2) when K=1. We compare the obtained Precision@1 in this RQ with the one obtained using all the learning features in **RQ1**. We use Equation 4 to calculate the importance of a learning feature.

$$Importance = (P@1\,in\,RQ1)\ minus\ (P@1\,in\,RQ2) \quad (4)$$

We rank all the learning features based on the importance values in a descending order. We have 10 sub-categories of learning features in three main categories. We only use 8 sub-category features in this analysis, as our service repository has no information of non-functional service attributes, and most importantly, our prototype is designed to offer only the first top picked service. Therefore, we omit the two sub-category features (*i.e.*, Non-Functional and Service Pick) in identifying the most important learning features.

Results. Our results in Table V show that among three main categories, "User Contexts" is the most important one,

Table IV: Top 5 learning features for each subject.

Top	User 1	User 2	User 3	User 4	User 5	User 6	User 7	User 8	User 9	User 10	User 11	User 12
1	Vertex P.	Date	Time	Date	Location	Date	Usage	Usage	Devices	Exec. Seq.	Date	Exec. Seq.
2	Date	Location	Score	Score	Date	Exec. Seq.	Time	Date	Location	Date	Time	Time
3	Score	Devices	Location	Location	Devices	Score	Date	Location	Score	Location	Score	Devices
4	Location	Time	Devices	Devices	Time	Location	Location	Devices	Time	Devices	Location	Usage
5	Devices	Vertex P.	Date	Exec. Seq.	Exec. Seq.	Devices	Devices	Time	Date	Score	Devices	Score

Note: Vertex P. and Exec. Seq. denote vertexes passes and execution sequences features, respectively.

Table V: Results of feature importance. R: Removing.

Category/Features	P@1 After R features	Importance
User Context	51.32%	9.90%
Date	51.38%	9.84%
Devices	57.61%	3.61%
Location	58.67%	2.55%
Time	60.75%	0.47%
Service Context	59.07%	2.15%
Usage	59.07%	2.15%
Score	61.02%	0.20%
User Execution History	60.56%	0.66%
Execution Sequence	57.60%	3.62%
Vertex Passes	64.52%	-3.30%

as it decreases the performance of AdaRank by 9.90%. Furthermore, among 8 sub-categories, "Date", "Execution Sequence", and "Devices" are the 3 most important features, and decreases the performance of AdaRank by 9.84%, 3.62%, and 3.61%, respectively. Moreover, Table IV shows the top 5 learning features for each subject, and each subject has a similar set of important features. For example, "Date" ranks the top for 4 out of 12 subjects and "Execution sequences" ranks the top for 2 of 12 subjects.

User Contexts is the most important learning feature category that affects the performance of our framework the most.

RQ3. Are users satisfied with our personalized assistance framework for helping select services?

Motivation. In addition to the empirical evaluations, we are interested to obtain users' perception of using our framework to achieve goals.

Approach. We send out an on-line feedback survey as shown in Table VI to our 12 subjects to measure the user experience in terms of usefulness, importance, accuracy, and completeness of using our framework. Each question has 5 choices (*i.e.*, strongly disagree, disagree, neutral, agree, and strongly agree) for a subject to choose. We collect and summarize their responses to the survey.

Results. The results in Table VI show that the majority of our subjects do not posses a complete knowledge to achieve their goals. Therefore, it is important to automatically guide users through the process of achieving their goal. Overall, the subjects agree that our personalized assistance framework is helpful in achieving their goals and useful in saving their time in finding their personalized services.

Table VI: Evaluation results on users experience about our framework. Perc.: denotes percentage of acceptance; Total: denotes number of users who agree or strongly agree.

Summary	User Study Questions	Perc.	Total (#/12)
Usefulness	Is our system helpful in achieving your goals?	92%	11
	Does our system preserve your time by minimizing interactions to achieve your goals?	83%	10
Importance	Is it crucial to identify actions and constraints from the entered goal descriptions?	75%	9
	Is it essential to minimize interactions with the system to achieve your goals?	75%	9
Accuracy	Are the actions and constraints identified from the goal descriptions meaningful?	83%	10
	Are the interactions from our framework for the goals meaningful?	92%	11
Completeness	Do you have a complete knowledge to achieve the goals yourself?	58%	7
	Do the interactions with our system help find the most appropriate service to achieve your goals?	83%	10

Our personalized assistance framework is more efficient in helping users achieve their goals.

IV. Related work

In this section, we summarize the related work on knowledge creation and service discovery.

Process knowledge Creation. Creating machine readable knowledge is widely studied in various research communities, such as natural language processing, information retrieval, and web mining. Researchers have developed various frameworks for building ontologies from textual resources (*e.g.*, Text2Onto [9], OntoLearn [27], Mo'K Workbench [2] and OntoLT [4]). Some other research has extensively focused on building ontologies by crawling and extracting domain specific informations from web resources, such as instruction articles and online sites (*e.g.*, [15, 29]). However, the above studies do not use the built knowledge to find the most appropriate services for users. In our approach, we use the built knowledge from online websites to generate relevant questions and interact with users to help them in finding the appropriate services for their user goals.

Service Discovery. The main purpose of discovering services is to accomplish users' specific needs. In prior years, researchers have developed various techniques to identify and recommend services to users (*e.g.*, [3, 8, 16]). For example, Chen *et al.* [8] propose a collaborative filtering technique to recommend services to users based on the Quality of services. Balke and Nolan [3] propose an enhanced syntactical matching technique which computes service recommendation scores between user's operational sessions and the description of web services, and uses the score to decide whether the user is interested in the service or not. Maamar *et al.* [16] propose a contextual model for service recommendation based on web service and resource context. All the above studies focus on recommending services to users. However, these studies fail to learn from user historical service selection history. In our framework, we take into consideration both the user's service selection history and contextual information to help recommend services to users.

V. Conclusion and Future Work

Due to the increasing popularity of service computing, a large number of services are available on the Web. Users utilize those services to attain their personal goals. In this paper, we provide a personalized assistance framework using AdaRank, a learning-to-rank machine learning algorithm, to achieve user goals by asking users the most relevant and necessary questions for quickly identifying the desired services. In particular, our framework automatically analyzes and learns from the user contexts and historical service selection for generating the most relevant and necessary questions that can quickly narrow down the search space of identifying desired services. Our framework improves the nine baselines by 12.02%–31.52% in terms of Precision@1. Our user case study results show that our framework is helpful in attaining user goals and useful in saving users time by finding their personalized services.

In the future, we plan to create a larger process knowledge base with more types of on-line resources. Furthermore, we would like to perform our user study on a larger scale of user base and the resources in more domains to remove any possible bias.

References

[1] W.-T. Balke and M. Wagner, "Towards personalized selection of web services." in *WWW (Alternate Paper Tracks)*, 2003, pp. 20–24.

[2] G. Bisson, C. Nédellec, and D. Canamero, "Designing clustering methods for ontology building-the mo'k workbench." in *ECAI workshop on ontology learning*, vol. 31. Citeseer, 2000.

[3] M. B. Blake and M. F. Nowlan, "A web service recommender system using enhanced syntactical matching," in *IEEE ICWS*, 2007, pp. 575–582.

[4] P. Buitelaar, D. Olejnik, and M. Sintek, "Ontolt: A protégé plug-in for ontology extraction from text in: Proceedings of the demo session of the international semantic web conference," *Florida*, 2003.

[5] C. Burges, T. Shaked, E. Renshaw, A. Lazier, M. Deeds, N. Hamilton, and G. Hullender, "Learning to rank using gradient descent," in *Proceedings of the 22nd ICML*, pp. 89–96.

[6] Z. Cao, T. Qin, T.-Y. Liu, M.-F. Tsai, and H. Li, "Learning to rank: from pairwise approach to listwise approach," in *Proceedings of the 24th ICML*, pp. 129–136.

[7] I. Y. Chen, S. J. Yang, and J. Zhang, "Ubiquitous provision of context aware web services," in *IEEE SCC*, 2006, pp. 60–68.

[8] X. Chen, X. Liu, Z. Huang, and H. Sun, "Regionknn: A scalable hybrid collaborative filtering algorithm for personalized web service recommendation," in *IEEE ICWS*, 2010, pp. 9–16.

[9] P. Cimiano and J. Völker, "text2onto," in *International Conference on Appl. of Natural Language to Information Sys.*, 2005, pp. 227–238.

[10] Y. Freund, R. Iyer, R. E. Schapire, and Y. Singer, "An efficient boosting algorithm for combining preferences," *Journal of machine learning research*, vol. 4, no. Nov, pp. 933–969, 2003.

[11] J. H. Friedman, "Greedy function approximation: a gradient boosting machine," *Annals of statistics*, pp. 1189–1232, 2001.

[12] S. J. Hanson, R. E. Kraut, and J. M. Farber, "Interface design and multivariate analysis of unix command use," *ACM Transactions on Information Systems*, vol. 2, no. 1, pp. 42–57, 1984.

[13] A. Lee, *Investigations into history tools for user support*. University of Toronto, 1992.

[14] H. Liu and P. Singh, "Conceptneta practical commonsense reasoning tool-kit," *BT technology journal*, vol. 22, no. 4, pp. 211–226, 2004.

[15] Y. Liu and A. Agah, "Crawling and extracting process data from the web," in *Conf. on Adv. Data Mining and App.*, 2009, pp. 545–552.

[16] Z. Maamar, S. K. Mostefaoui, and Q. H. Mahmoud, "Context for personalized web services," in *System Sciences,Proceedings of the IEEE International Conference on*, 2005, pp. 166b–166b.

[17] C. D. Manning, P. Raghavan, H. Schütze *et al.*, *Introduction to information retrieval*. Cambridge university press, 2008, vol. 1.

[18] M. McCandless, E. Hatcher, and O. Gospodnetic, *Lucene in Action: Covers Apache Lucene 3.0*. Manning Publications Co., 2010.

[19] D. Metzler and W. B. Croft, "Linear feature-based models for information retrieval," *Information Retrieval '07*, vol. 10-3, pp. 257–274.

[20] R. Mihalcea, "Graph-based ranking algorithms for sentence extraction, applied to text summarization," in *ACL 2004*, p. 20.

[21] M. T. Pilehvar and R. Navigli, "From senses to texts: An all-in-one graph-based approach for measuring semantic similarity," *Artificial Intelligence*, vol. 228, pp. 95–128, 2015.

[22] M. F. Porter, "An algorithm for suffix stripping," *Program*, vol. 14, no. 3, pp. 130–137, 1980.

[23] H. Schütze, "Introduction to information retrieval," in *Proceedings of the ACM international conference on computing machinery*, 2008.

[24] R. Speer and C. Havasi, "Representing general relational knowledge in conceptnet 5." in *LREC*, 2012, pp. 3679–3686.

[25] B. Upadhyaya, F. Khomh, Y. Zou, A. Lau, and J. Ng, "A concept analysis approach for guiding users in service discovery," in *5th IEEE International Conference on SOCA*, pp. 1–8.

[26] B. Upadhyaya, Y. Zou, S. Wang, and J. Ng, "Automatically composing services by mining process knowledge from the web," in *International Conference on Service-Oriented Computing*, 2013, pp. 267–282.

[27] P. Velardi, Navigli *et al.*, "Evaluation of ontolearn, a methodology for automatic learning of domain ontologies," *Ontology Learning from Text: Methods, evaluation and applications*, vol. 123, p. 92, 2005.

[28] Q. Wu, Burges *et al.*, "Adapting boosting for information retrieval measures," *Information Retrieval*, vol. 13, no. 3, pp. 254–270, 2010.

[29] H. Xiao, B. Upadhyaya, F. Khomh, Y. Zou, J. Ng, and A. Lau, "An automatic approach for extracting process knowledge from the web," in *IEEE ICWS*, 2011, pp. 315–322.

[30] J. Xu and H. Li, "Adarank: a boosting algorithm for information retrieval," in *Proceedings of the 30th annual international conference on Research and development in information retrieval*, pp. 391–398.

[31] S. J. Yang, J. Zhang, and I. Y. Chen, "A jess-enabled context elicitation system for providing context-aware web services," *Expert Systems with Applications*, vol. 34, no. 4, pp. 2254–2266, 2008.

[32] Y. Zhao, S. Wang, Y. Zou, J. Ng, and T. Ng, "Mining user intents to compose services for end-users," in *IEEE ICWS*, 2016, pp. 348–355.