A Personalized Software Assistant Framework To Achieve User Goals

by

Pradeep Kumar Venkatesh

A thesis submitted to the
Department of Electrical and Computer Engineering
in conformity with the requirements for
the degree of Master of Applied Science

Queen's University
Kingston, Ontario, Canada
September 2017

Copyright © Pradeep Kumar Venkatesh, 2017

Abstract

The growing trend of devices participation in Internet of Things (IoTs) platforms has created billions of IoT devices for users. The rapid trend has made users to install IoT devices at homes to achieve their goals, such as to reduce electricity cost. Moreover, the increasing 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 to book flight tickets. Existing research with personalized software assistants has been conducted to assist users majorly in e-commerce sites for customized search recommendations. However, the potential of personalized software assistant systems usage in a user-centric model is highly unrealized in assisting users to achieve their personal goals through personalized context-aware interactions with users based on behavioural habits, such as to smartly recommend IoT devices usage in smart-homes to reduce electricity expenses, and engage users more during the process of service selection to achieve their goals. In this thesis, First, we propose an engine that identifies the behavioural patterns of IoT device users to make smart recommendations to reduce users cognitive overload. Then we propose an intellectually cognitive personalized assistant framework which helps users to achieve their personal goals through personalized context-aware interactions for selection of

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. 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 to achieve their goals. Further, we conduct a user study to obtain users' perception of using our framework to achieve their personal goals. Our user study results show that our framework is helpful in achieving user's goals and saves users time in finding their personalized services faster.

Co-Authorship

1. A Framework To Extract Personalized Behavioural Patterns of User's IoT Devices Data.

   Pradeep Kumar Venkatesh, Daniel Alencar da Costa, Ying Zou, and Joanna W. Ng. Proceedings of the 27th Annual International Conference on Computer Science and Software Engineering (IBM CASCON 2017).

   My contributions: Drafting the research plan, collecting the data, analyzing the data, writing and polishing the paper drafts.

2. A Personalized Assistant Framework for Service Recommendation.

   Pradeep Kumar Venkatesh, Shaohua Wang, Ying Zou, and Joanna W. Ng. Proceedings of the 14th IEEE International Conference on Services Computing (SCC 2017).

   My contributions: Drafting the research plan, collecting the data, analyzing the data, writing and polishing the paper drafts.

Acknowledgments

First and foremost, I would like to express my deep gratitude to my advisor and mentor, Professor. Dr. Ying (Jenny) Zou, for providing me the opportunity to study and do research under her invaluable guidance. In addition to her strong motivation, patience, and immense knowledge. I would like to thank her for being an encouraging and very supportive supervisor throughout my master studies.

I am very thankful for the opportunity to work and collaborate with Dr. Ahmed E. Hassan, Dr. Feng Zhang, Dr. Shaohua Wang, Dr. Daniel Alencar da Costa, and the collaborator at IBM Toronto Lab, Ms. Joanna W. Ng and for their invaluable insights and constructive inputs to my research. It gives me an immense pleasure to express my thanks to Professor. Dr. Mohammad Zulkernine and Professor. Dr. Scott Yam for taking their valuable time to sit in my thesis examination committee.

I am very special and lucky to know stunningly amazing people at Software Re-engineering Lab and at the Department of Electrical and Computer Engineering, Queen's University. I would like to express my special thanks to my labmates for their support throughout my endeavor: Mr. Yonghui Huang, Mr. Yu Zhao, Ms. Mariam El Mezouar, Mr. Ehsan Noei, and Mr. Yongjian Yang. Special thanks to Debra Fraser, the graduate program assistant for her moral support and encouragement.

Further, I would also like to thank my friends for their constant support.

Last but not the least, I would like to thank my parents, Venkatesh (Father), Krishnaveni (Mother) for their unconditional love, patience, and support. Without their encouragement, this thesis would not have been possible. Finally, I will be eternally grateful to my beloved wife Vinethaa for her devotion and sacrifice, and to my parent-in-laws, Govindaraj (Father-in-law) and late Padmavathy (Mother-in-law) for their support.

Specially, I would like to dedicate this thesis to my beloved goddess Arulmigu Masani Amman and god Venkateswara.

Sincerely,

Pradeep Kumar Venkatesh

Kingston, Ontario

Canada

Contents

List of Tables

List of Figures

Chapter 1

Introduction

The advancements in hardware and software, computers have no doubt changed the way humans live. The exponential growth in hardware engineering has created a lot of interests among industries and academic researchers to create more energy efficient and internet enabled devices for users. For example, the participation of the device on the Internet of Things (IoTs) platform has created more than 2 billion IoT devices since 2006 and it is expected to reach 50 billion connected devices by 2020 [79]. On the other hand, the rapid development in software engineering, especially in service-oriented computing has put a pressure on computer science researchers to design, build, and deliver highly complex personalized software assistant systems. The personalized software assistant systems are built to help users achieve their personal goals, such as workflow management [68]. Due to the vast advancement and popularity of both the hardware and software engineering disciplines, more and more companies are starting to invest billions of dollars building energy-efficient, user-friendly, Internet-enabled (*i.e.*, IoT) devices and knowledge-based personalized assistant systems to reduce users cognitive overload.

1.1 Background

1.1.1 Personalized Software Assistants

With the vast spread of advanced personal computing, personal software assistants systems have been developed to assist users in achieving their personal tasks, such as work-flow management, information and mail organization, and calendar scheduling [68, 45, 24]. In particular, knowledge-based personal software assistants are popular among the users as it acts kind of a software secretary which perform various tasks, such as making travel arrangements, paying bills, and locating accurate information on artifacts in on-line libraries. The knowledge-based personal software assistant's success rate among the users depends as much on the knowledge of the particular user's habits and goals as on the outlined set of operations the software assistants can perform. For example, a software assistant is designed to correctly model the user's criteria to sort the incoming mail, such as urgent, and others category. Whether the assistant will succeed or fail the mail sorting will heavily depend on the accuracy of modelling the user's criteria by learning through experience, such as using users' historical information to model the criteria. The personal software assistants are distinguished into three distinct classes [40]: 1) Gopher Software Assistant, 2) Service Performing Software Assistant, and 3) Predictive Software Assistant.

Gopher Software Assistant

The personal gopher software assistants are designed to perform straightforward tasks based on the pre-defined criteria's and assumptions. For example, the gopher

assistant will notify the owner when the stock items need to be re-ordered, or alert the stack-holders when the share price drops 10% below the current price.

Service Performing Software Assistant

The personal service performing software assistants are designed to execute well-defined tasks at the request of the users. For example, the service performing assistant will arrange a meeting with team members sometime next week, or to find a cheapest business class in a star alliance flight to Vancouver.

Predictive Software Assistant

The predictive personal software assistants are developed to provide information or services to users without an explicit request from users (*i.e.*, whenever it is considered appropriate). For example, a travel assistant, knowing the user is interested in visiting parks, may monitor the online websites and inform the user when the discount is available on parks visiting tickets. Moreover, a monitoring personal software assistant which monitors online documents, artifacts and newsgroups to return the snippets that it believes to be an interest to the user.

1.1.2 Internet of Things

The Internet of Things (IoTs) are a network of internet-connected devices that collect and exchange data using embedded sensors. Due to the increasing growth of devices participation in IoTs platform, have created more than 2 billion devices since 2006 [79]. Many companies have started investing more to build IoT devices

for users, such as Google Nest Cam IQ[1]. On the other hand, users use devices in their favour to achieve their personal goals, such as to monitor their homes. Typically, the IoT devices perform the following three tasks for users: 1) Monitor the environment (*e.g.*, monitor the room temperature); 2) Perform actions (*e.g.*, turn on/off fans); or 3) both 1 and 2 (*e.g.*, if room temperature reaches below 19°C then turn on the heater). In general, users use cloud-based platforms to store their IoT devices data, such as dweet.io.

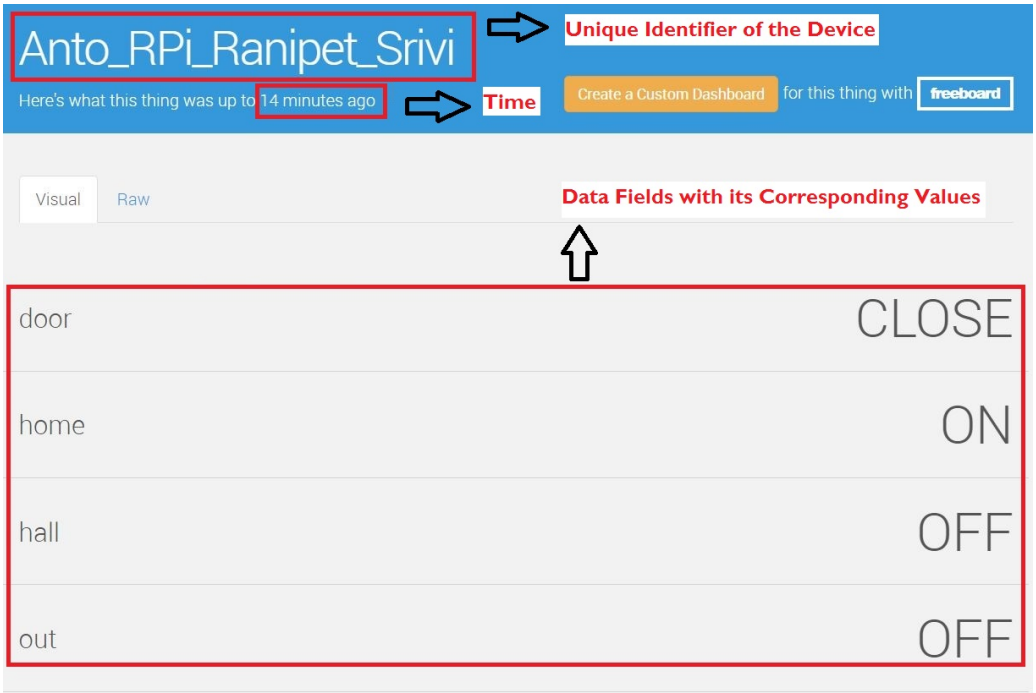

Figure 1.1: An annotated screen shot of the dweet.io, a published message of a device.

dweet.io is a simple publishing and subscribing site for users to store and share their device data in real-time. The device refers to machines, sensors, devices, robots, and

[1]Nest Cam IQ is an advanced intelligent IoT device build by Google which can tell the difference between a person and a thing to alert the user.

gadgets. The publishing data messages to the platform are referred as "dweets". The dweet.io site allows dweets to be up to 2,000 characters payload. The dweets can be easily assessed through a web based RESTful API.[2] Typically, users publish their device data for simple sharing, storage, and alerts purposes.

Figure 1.1 illustrates a dweet message posted on the dweet.io site. There are mainly four pieces of information, such as a unique identifier of the device, the data fields of the device and its corresponding values, and the time it was posted.

1.1.3 Service Oriented Architecture

The Service Oriented Architecture (SOA) is an Information Technology architectural paradigm to assist the integration process of repeatable tasks. A lot of definitions are available to define the SOA around the concept of services [80, 6, 25]. In general, the SOA has three main components: 1) service registry; 2) service provider; and 3) service consumer. The service provider builds the services and advertises the services in the service registry for the service consumers to access. The service consumer is an application or an another service which looks for service in the service registry to consume the service. Figure 1.2 shows an overview of the service oriented architecture with the elaborated demonstration of interactions among those three components.

Service discovery is the process of finding the suitable service to meet the requirements of fulfilling the user objectives (*e.g.*, checking current stock price). Lately, search engines have become popular among users for searching for web services, such

[2]https://dweet.io/get/latest/dweet/for/<my-thing-name>, where <my-thing-name> should be replaced with the assigned unique name of the IoT device.

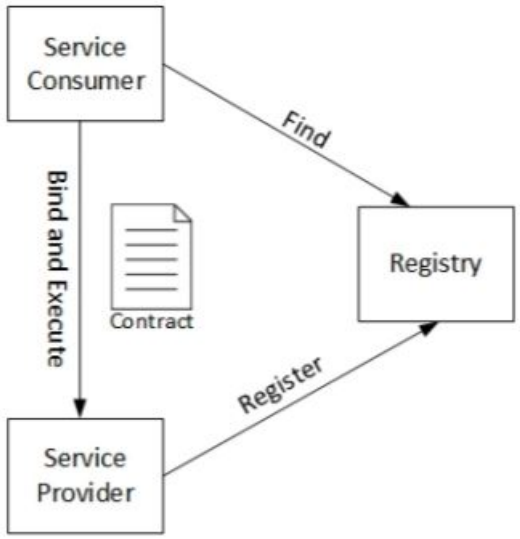

Figure 1.2: An overview of service oriented architecture.

as Google Search[3] and Bing[4] search engines. Moreover, some search engines are particularly designed to retrieve services, such as Vector Space [65] and Woogle [1]. The service consumer can use service search keywords or some specific query strings to identify services for users using these particular search engines.

1.2 Research Problems

The personalized software assistant systems are being used in an increasingly wide range of software applications from comparatively small systems, such as e-mail organizing and filtering in mailboxes (*e.g.*, [45]) to a very large-scale complex mission critical systems such as aircraft traffic control systems (*e.g.*, [41]). In general, the software assistant systems are designed with several characteristics, such as goal-oriented,

[3]https://www.google.ca
[4]https://www.bing.com

intelligent, and autonomous. The characteristics make the software assistant systems well suited for the role of personalized recommender to users. Nowadays, there is an increasing demand for applying the capabilities of personalized software assistants systems to help users achieve their personal goals, such as to intelligently control user's devices in smart-home environments and to purchase movie tickets online.

The growing trend of devices participation on the Internet of Things (IoTs) platforms have created billions of IoT devices in both consumer and industrial environments. (*e.g.*, [74, 63]). Due to the increasing growth of IoT devices, more and more companies begin to invest billions of dollars in producing IoT devices for consumers, such as Amazon, Google, and Cisco. In addition, consumers (*i.e.*, users) use IoT devices in their favour to achieve their personal goals, such as to save electricity cost at their homes. Typically, users install multiple IoT devices in their smart-home environments, such as smart garage door, kitchen lights, and bathroom lights and fan. However, the process of controlling each IoT device individually can be overwhelming and tiresome, since the manual effort from a user causes frustration (*e.g.*, [35]). To reduce the user's cognitive overload, it is crucial to understand the IoT device usage in users' smart-home environments and propose approaches to make smart recommendations of the IoT device usage in their smart-home environments.

The popularity of service-oriented computing makes more and more services available on the Web. For example, the ProgrammableWeb[5], a popular online services repository, alone indexes 16,521 services as of Jan 31st, 2017. Users make use of these services to achieve their personal goals, such as purchasing movie tickets online and

[5]https://www.programmableweb.com/

booking flights. Existing research has proposed various techniques to help users to select services to achieve their goals [76, 5, 15]. Typically, the user choice of services changes 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. It often leads to not finding the most appropriate services for users. Therefore, it is critical to analyze and have more personalized context-aware interactions with users to recommend personalized services based on the users' history and contextual information to save their time.

Existing research with personalized software assistants has been conducted to assist users majorly in e-commerce sites (*e.g.*, [62]). The key characteristic of an e-commerce site is that they will unavoidably move more towards a user-centric model in order to be more competitive in the market. Therefore, caused an increasing demand for software assistant systems in e-commerce sites. However, the potential of personalized software assistant systems usage in a user-centric model is highly unrealized when it comes to assisting users to achieve their personal goals through personalized recommendations and context-aware interactions.

1.3 Thesis Statement

The main objectives of my research work are to realize the potential of personalized software assistant in a user-centric model by studying the user behaviour using machine learning techniques to make smart recommendations for automatically controlling users' IoT devices in smart-home environments, and to have personalized context-aware interactions with users to recommend services which help achieve their

personal goals, such as to find a budget accommodation in downtown Vancouver.

- **Extract Personalized Behavioural Patterns of User's IoT Devices Data (Chapter 3)**

  We propose an engine that identifies the behavioural patterns of IoT device users. First, our engine captures the IoT devices data in a smart-home environment and uses a relational database to store the IoT devices usage data. Second, our engine prepares the stored data in the database in a suitable model for further data analysis. Finally, our engine analyses the represented data to extract user behavioural patterns. We aim to identify the behavioural patterns of users' IoT device usage, such as identifying the most used devices at specific times and studying the relationships between the most used devices and other devices in the environment. The learned users IoT devices behavioural patterns can be used to communicate with users to notify when the abnormality of the device behaviour happens and to make smart recommendations to propagate actions across devices in the environment automatically, without any user intervention to reduce users cognitive overload.

- **A Personalized Assistant Framework for Service Recommendation (Chapter 4)**

  We propose an intellectually cognitive personalized assistant framework to help users achieve their personal goals by engaging them more through personalized context-aware interactions during the process of service selection. In particular, our framework uses learning-to-rank machine learning algorithm, to improve

the user's service selection experience by asking necessary and the most relevant questions of their intended goals to help narrow down the search space of identifying the most preferred services. We have designed and developed a prototype as a proof of concept. We conduct 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.

1.4 Organization of Thesis

The remaining chapters of this thesis are organized as follows:

- **Chapter 2. Background and Related Work.** We present the background and the existing work on personalized behavioural knowledge extraction of IoT device users, process knowledge creation from online sources, service oriented architecture, and discovery of web services.

- **Chapter 3. Extract Personalized Behavioural Patterns of User's IoT Devices Data.** We propose a behavioural extraction engine that identifies the personalized behavioural patterns of IoT device users.

- **Chapter 4. A Personalized Assistant Framework for Service Recommendation.** We propose a personalized assistance framework using AdaRank, a learning-to-rank machine learning algorithm, which improves the user service

selection experience by asking them the most relevant and necessary questions of their intended goals from the built process knowledge to help narrow down the search space of identifying the most preferred services.

- **Chapter 5. Conclusion.** This chapter concludes the thesis by highlighting the contributions of the work and offering potential directions for future work.

Chapter 2

Background and Related Work

In this chapter, we discuss the background and the existing work on personalized behavioural knowledge extraction of IoT device users, process knowledge creation from online sources, learning-to-rank machine learning algorithms, service oriented architecture, and discovery of web services.

2.1 Personalized Behavioural Knowledge Extraction

2.1.1 Knowledge Discovery

The various techniques from the machine learning, pattern recognition, and artificial intelligence areas can be applied to discover knowledge from machines, sensors, devices, robots, and gadgets data [36]. The research related to the three popular knowledge discovery techniques that are well established in the above disciplines are described below [82]:

(DT1.) Association Analysis is a process of uncovering the relationships that exists among the data [75]. The association rules are the models that identify how the data items in a dataset are associated to each other. The association analysis has been used in various research endeavours, such as market analysis (*e.g.*, [51, 18]) and gene classifications (*e.g.*, [69, 30, 4]). Regarding IoT devices data, the association analysis for knowledge discovery is very useful. However, the technique is not been thoroughly explored on IoT devices data for knowledge discovery. In our research, we have used the association rule mining technique to identify the relations among devices in the users' home environment.

(DT2.) Clustering Analysis is a process of partitioning a set of analyzed data into subsets [26]. Each subset is represented as a cluster. The data within the same cluster is similar to one another while different among in other clusters. The clustering analysis is a classic knowledge discovery technique. In clustering analysis, various clustering methods are available. In IoT devices data, different clustering methods may yield different varieties of clusters on the same set

of analyzed data. For example, clusters, formed on counting the persons inside the garage and the parked vehicles in the garage, are different from clusters formed when trying to count the vehicles parked in the garage by their parked direction. Ortiz *et al.* [60] used the clustering analysis to cluster between the IoT and Social networks to enable the connection of people to the ubiquitous computing devices. Sohn and Lee [72] applied the clustering analysis by ensembling the individual classifiers from two categories of severity, such as property damage and bodily injury in road accidents based on the data collected from devices installed in their city. The clustering analysis is not yet been widely studied among personalized IoT devices. In our research, due to very limited availability of IoT devices for each user the clustering analysis technique was not needed.

(DT3.) Outlier Analysis is a process of identifying the data point that is very different from most of the remaining data [2]. The clustering analysis determines the groups of data points that are identical and forms a cluster, whereas outlier analysis identifies the individual data point that are different from the remaining data. Outliers are also commonly referred as abnormalities or anomalies. In IoT devices, the data might not comply with the general actions (*i.e.*,may have abnormalities), such as falsifying the fire alarm at home. The outlier analysis is widely used in research studies, such as the one performed by Elio Masciari which runs an outlier analysis on Radio-Frequency identification (RFID) data streams to identify the tags that are abnormally attached to the objects [55]. Hromic *et al.* [39] used the statistical outlier analysis detection in the real-time

sensor data of Internet of Things for events processing using intelligent servers. Kantarci *et al.* [42] used outlier detection analysis for environmental safety by measuring the trustworthiness of data received from cloud-centric IoT. In general, the outliers are detected from the normal data sets so that they can be discarded to keep the study environment pure. The analysis is not yet been vastly studied due to the limited availability of users IoT devices data to public. However, in our research the outlier analysis was not necessary due to the availability of clean IoT devices data.

2.2 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 [20], OntoLearn [78], Mo'K Workbench [7] and OntoLT [11]). 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.*, [50, 83]). 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.

2.3 Learning-to-Rank Algorithms

Learning-to-Rank (LtR) algorithms have been widely studied and used in various software applications for Information Retrieval. In general, all LtR algorithms have two phases: 1) training phase and 2) testing phase. In the training phase, given a set of queries (Q) using Equation 2.1.

$$Q = \{q^1, q^2, q^3, ..., q^n\} \tag{2.1}$$

where n is the number of queries, and a collection of documents.

Each query q^i $(1 \leq \ i \geq \ n)$ is mapped to a collection of documents $D^i = \{d_1^i, d_2^i, d_3^i, ..., d_m^i\}$ where the d_l^i denotes the l^{th} document. Each list of documents (*i.e.*, D^i) corresponds to a list of relevance results $R^i = \{r_1^i, r_2^i, r_3^i, ..., r_m^i\}$ where the R_l^i denotes the relevance result on document d_l^i with respect to the query q^i. In general, the relevance results are represented in five levels, such as perfect-match, excellent match, good match, fair match, and bad match. The higher the document relevance match (*i.e.*, excellent match) is, the more relevant the document is to query.

A feature vector is created for each query-document pairs. The features are defined as functions of the query-document pair. Given a set of training data, the training phase will automatically learn the function $F(x)$. The feature vectors are ranked according to $F(x)$, where the top K results are evaluated according to their corresponding relevance results. The LtR algorithms perform better than the conventional ranking algorithms (*e.g.*, Bayesian belief networks based ranking algorithm [23]), by automatically learning and combining the different factors affecting the ranking to

suggest an ideal ranked list of a given query. The LtR algorithms are categorized into three different types: 1) Pointwise Algorithms, 2) Pairwise Algorithms, and 3) Listwise Algorithms.

2.3.1 Pointwise Algorithms

Pointwise ranking algorithms transform the ranking problem into a regression classification problem. Each query-document pair in the training data has a numerical or ordinal value. The training setup is viewed as a regression classification problem. For example, the PRank model employs the Perceptron algorithm to simultaneously learn linear models [22]. Given a set of training data, the PRank model iteratively learns a number of parallel Perceptron models and the build each model separates the two neighbouring grades.

2.3.2 Pairwise Algorithms

Pairwise ranking algorithms perform the ranking based on three categories of learning models, such as Support vector machine, Boosting techniques, and Neural networks.

The Support Vector Machine (SVM) [21] is a supervised machine learning model for classification and regression based analysis. A few LtR algorithms use SVM as their learning model (*e.g.*, Ranking SVM). For example, the Ranking SVM [38] formalizes the ranking problem as a pairwise classification problem and uses the SVM technique to perform the ranking task of document retrieval.

Some other pairwise algorithms use Boosting techniques (*e.g.*, [28, 88]). It is a learning model which uses the supervised machine learning algorithm to reduce

bias and variance [9]. For example, the RankBoost LtR algorithm [28] is based on gradient boosting technique which aims to select a set of weak rankers to build a strong ranking function. The GB Rank LtR algorithm [88] uses regression ranking by employing a pairwise regression loss function. The pairwise regression loss function iteratively minimizes the loss function based on Gradient Tree Boosting algorithm.

A family of statistical learning models, such as Neural networks have been used by pairwise algorithms. For example, RankNet LtR algorithm [12] is a pairwise classification ranking algorithm. RankNet uses the neural network as a ranking model and cross entropy as loss function to measure the loss. Gradient Descent is used by the RankNet algorithm to learn the optimal neural network model.

2.3.3 Listwise Algorithms

Listwise LtR algorithms take the ranking list as instances in both learning and prediction while maintaining the group structure of ranking. The evaluation measures of ranking are more directly incorporated into the loss function of the ranker. The Listwise algorithms can be grouped into two categories based on optimization. First, based on the performance measure of Information Retrieval (IR) calculated using metrics, such as Mean Average Precision (MAP) and Normalized Discounted Cumulative Gain (NDCG). One set of algorithms can optimize the objective functions based on the bounds of the IR measures. For example, AdaRank [84] minimizes the exponential loss of the ranker function directly based on the evaluation measure with the use of boosting techniques.

Some other Listwise algorithms do not create replacement functions based on IR

measures (*e.g.*, [14, 13, 12]). LambdaRank [13] uses the implicit listwise loss function optimized with Gradient Descent to learn the optimal ranking model. LambdaRank uses the neural network model for training the ranker. ListNet [14] uses the neural network model for training and makes uses of Kullback-Leibler divergence [46] for loss function. LambdaMART [12] employs the Gradient Tree [29] boosting technique to learn a boosted regression tree as a ranking function. The efficiency and accuracy of LambdaMART are significantly better than the LambdaRank [81].

2.4 Service Oriented Architecture

The Service-oriented architecture (SOA) is an approach used to create an software architecture based upon the usage of services. The SOA have two most popular types of services: 1) SOAP-based services; and 2) REpresentational State Transfer architectural (RESTful services).

2.4.1 SOAP-based Web Services

The web services are commonly used to implement SOAs. Web Service Description Language (WSDL) is used to define service interfaces [19]. WSDL describes the web service with the description of the services abstract functionality and the complete service description details, such as how and where the service is offered. SOAP protocol is used to communicate between the service provider and the consumer [34].

SOAP web services is a stateless one-way message exchange model between two parties (*i.e.*, from a SOAP sender to a SOAP receiver). The SOAP message is in a standard XML document. Figure 2.1 shows the basic format of a SOAP message.

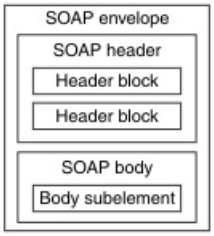

Figure 2.1: The basic structure of a SOAP message.

The SOAP message have three major components: 1). SOAP Envelope; 2). SOAP Header; and 3). SOAP Body. The SOAP envelope is the root tag which holds both the SOAP Header and Body tags. The SOAP Header tag stores transactional information, such as session Id, state and transaction Id. The SOAP body tag have the XML document which contains elements, such as the data, arguments, and reporting information for the recipient (*i.e.*, service consumer).

2.4.2 RESTful Services

RESTful services eliminates the complexity of processing the overhead of the SOAP-based web services [27]. RESTful services relies on the HTTP protocol for the interaction between the service provider and the consumer. The HTTP protocol messages has four basic operations: GET, PUT, POST, and DELETE. The GET requests the service provider to retrieve the data. The PUT requests the service provider to update the old data with new information sent in from the consumer. The POST request is used to create a new resource. The DELETE request indicates that a service consumer wants the service provider to delete the data.

2.5 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.*, [17, 52, 8]). The service discovery process can be performed in either of three popular ways: 1). Information retrieval approach (*e.g.*, keyword based and classification types); 2). Semantic based approach (*e.g.*, OWL-S and hREST [57, 44]); and 3). Contextual data based approach (*e.g.*,OWL model context [10]).

2.5.1 Information Retrieval Approaches

Many research works use Information Retrieval techniques to find services for end-users (*e.g.*, [31, 32]). The two most-common form of service search is keyword based service matching and classification type service matching. The keyword based service matching type retrieves the service by matching the keywords provided by the service provider in the service description data field and keywords provided by the end-users [31]. For example, given a service search string (*i.e.*, user input), the service discovery process starts to search the textual services descriptions and returns the discovered candidate services set. End-users can then select the appropriate one from the discovered candidate service set. The classification type service matching automatically classify the services based on two main approaches [47]: heuristic and non-heuristic methods. The heuristic method uses the information in non-semantic service descriptions to identify the service category by using various data mining techniques, such as Natural Language Processing (NLP), Machine Learning (ML), and

Text-based classifications. The non-heuristic method uses the same service descriptions to classify services in to service categories dynamically using various clustering techniques, such as K-means and hierarchical clustering.

2.5.2 Semantic Based Approaches

The traditional information retrieval approaches lacks to identify the services based on their semantic service description matching [61]. In general, the service providers and the consumers have their unique knowledge and difference in perspective about the same service. Further, the service providers might not describe the service exactly like the service consumer needs. For example, the service provider may describe their service as a financial service, while the service consumer may look for services to have an updated stock price. The preception of semantic based web services is to offer high interoperability among the deployed web services, where the designed software application can dynamically identify and bind services without any prior knowledge of service discovery and invocation. A specific OWL ontology, OWL-S is designed to offer a framework structure of semantically describing services from many perspectives [54], such as service discovery and invocation. The main drawback with the semantic based service searching approach is that is not been widely used in the community, as it is difficult to convince service provider to use a particular ontology. Therefore, the adoption rate of semantic based approach is low.

2.5.3 Contextual Data Based Approaches

Broens *et al.* proposed a new context based methodology to model contextual and service description information using ontologies [10]. The service provider, service consumer, and context providers share the common knowledge platform as defined in ontologies. The ontologies defines four service properties for each service to have a service matching, such as service types, contextual information, inputs, and outputs. The service type is used to demonstrate the classification of a service. The contextual information represent the contextual information collected from both the service consumers and the service providers. The inputs and outputs are data attributes defined in ontology to specify the meaning of inputs and outputs of a services. The algorithm use the service type, inputs, and outputs service properties defined in ontology to search for services, and then use the contextual property to sort services. Balke *et al.* propose an algorithm to select personalized web services based on user profile information [5]. The algorithm takes the service search query string (*i.e.*, search text) from user as an input and if there are too many services, then it expands the service search query to include user's preferences. The algorithm narrows down the search space of services by adding constraints one at a time.

Chen *et al.* [17] propose a collaborative filtering technique to recommend services to users based on the Quality of services. Balke and Nolan [8] 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.* [52] 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 thesis work, we take into consideration both the user's service selection history and contextual information to help recommend services to users.

2.6 Summary

In this chapter, we introduce the background and existing studies on personalized behavioural knowledge extraction of IoT device users, process knowledge creation from online sources, learning-to-rank machine learning algorithms, service oriented architecture, and discovery of web services.

Chapter 3

Extract Personalized Behavioural Patterns of User's IoT Devices Data

*In this chapter, we propose an engine that identifies the personalized behavioural patterns of IoT device users. First, our engine captures the IoT devices data in a smart-home environment and uses a relational database to store the IoT devices usage data. Second, our engine prepares the stored data in the database in a suitable model for further data analysis. Finally, our engine analyses the represented data to extract user behavioural patterns. We perform an empirical study to evaluate our engine using 4 users and 31 IoT devices usage data collected from a well-known IoT data publishing-subscription site (*i.e.,*dweet.io*[1]*). Our results shows that users have, on average, use 2 devices at specific times and have a relatively small impact across other devices in the environment.*

[1]https://dweet.io/play/

3.1 Introduction

The exponential growth of smart devices participation in the Internet of Things (IoTs) platform created more then 2 billion smart devices since inception and is expected to reach about 50 billion connected smart devices in 2020 (*e.g.*, [79, 74, 63]). The IoT devices are a network of internet-connected devices that collect and exchange data using embedded sensors. In particular, the interconnection via the Internet of computing devices embedded in everyday objects, enabling them to send and receive data[2]. Due to the increasing growth of IoT devices, many companies have started investing to produce IoT devices for consumers, such as Amazon, Google, and Cisco. In addition, consumers (*i.e.*, users) use devices in their favour to achieve their personal goals, such as to save electricity cost at their homes.

Usually, IoT devices perform the following three tasks for users: 1) Sense and Monitor the environment (*e.g.*, monitor the room temperature); 2) Perform certain actions (*e.g.*, turn on/off lights); or 3) both 1 and 2 (*e.g.*, if room temperature reaches above 24°C then turn on the air conditioner). The IoT devices have their own user interface to let users take control of that particular device in a particular environment. Typically, users install multiple IoT devices in the desired environments, such as smart garage door, kitchen lights, and a foyer fan. However, the process of controlling each IoT device individually is very tiresome, since the manual effort from a user causes frustration (*e.g.*, [35]).

Research has been invested to help users achieve their personal goals using IoT devices (*e.g.*, [43, 33, 89, 71]). For example, existing research helps the identification

[2]https://en.oxforddictionaries.com/definition/us/Internet_of_things

of different interconnection mechanisms among the IoT devices to save energy in a smart home (*e.g.*, [89, 71]). Other line of research helps to better manage the resources of resource constraint IoT devices (*e.g.*, [33]). Kelly *et al.* [43] proposed a new implementation mechanism for IoT devices to monitor domestic conditions by means of low cost ubiquitous sensing systems. However, the existing prior work does not investigate the historical data of IoT devices usage to assist users in achieving their goals. For example, learning users' behavioural patterns from their IoT devices usage to automatically help users to achieve their personal goals.

In this chapter, we propose a personalized behavioural extraction engine using the Apriori algorithm, a rule mining learning technique. Our engine infers behavioural patterns from IoT devices usage data. The extracted behavioural patterns can be used to help users to intelligently control their IoT devices with minimal user involvement. An example of a behavioural pattern is: *during evening hours, the kitchen lights are ON, while the garage doors are CLOSED*. Our approach learns these behavioural patterns and use them to control the IoT devices. Our engine works in three steps. First, the engine uses a database to store the IoT devices usage data. Second, our engine prepares the data in a suitable representational model for analysis. Finally, the engine analyses the represented data to extract user behavioural patterns. To evaluate our approach, we perform an empirical study using 4 users and 31 IoT devices usage data collected from a well-known IoT data publishing-subscription site (*i.e.*,dweet.io[3]). We derive the behavioural patterns for users'. In this study, we investigate the following research questions:

[3]https://dweet.io/play/

(RQ1.) ***What are the most used devices at a given time?***

To understand which IoT devices are most used by users at a given time helps us to identify behavioural patterns of IoT device usage. (*e.g.*, garage door always open between 8am to 9am on weekdays). In total, we mine 35 behavioural rules of all the devices from 4 users. We observe that, on average, only 2 IoT devices are being used by users at certain time intervals (*i.e.*, IoT devices used more than 80% of times). Hence, users can concentrate on a limited number of devices when trying to control their environment.

(RQ2.) ***What is the relationship between the most used devices and the other devices in the environment?***

Studying the relationships between the most used devices and the other devices of the environment may be useful to give additional behavioural patterns (*e.g.*, one relationship shows that whenever the garage door is CLOSED the kitchen lights are ON). We observe, a small proportion of relationships among devices with a confidence interval between >50% to <80%. Therefore, users can use these identified relationships to intelligently take actions across other devices in the environment.

3.2 Our Proposed Engine

In this section, we present our proposed behavioural extraction engine. We describe the components of our engine as well as which inputs and outputs are consumed and generated by them. We also discuss the necessary steps that our engine performs

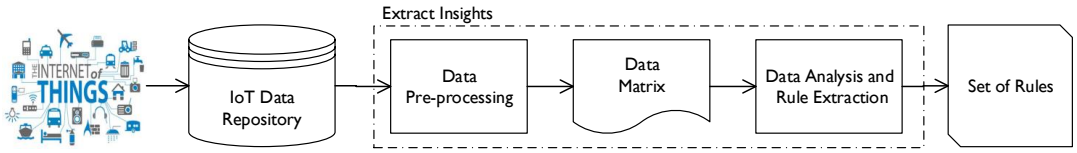

Figure 3.1: An overview of our behavioural extraction engine.

to extract behavioural patterns. Figure 3.1 provides an overview of how our engine works.

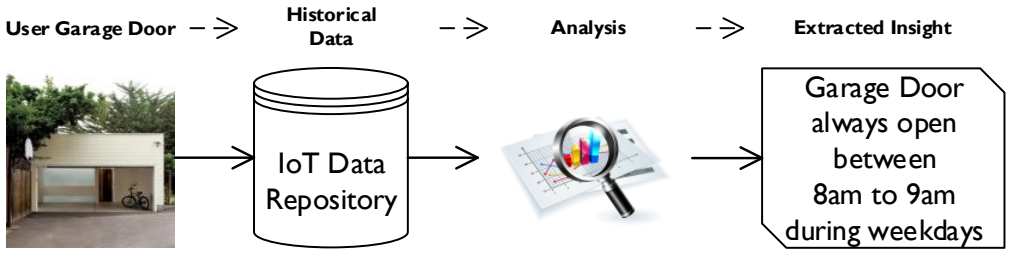

Figure 3.2: An example of how our engine works.

Our engine works in three simple steps. First, our engine collects the usage data for each IoT device. For example, the engine records the status of the device and its respective time-stamp (*e.g.*,{Device: Garage Door, Status: OPEN, Date: 01/01/2017, Time: 8am}). Next, in Step 2, our engine prepares the data in a suitable representational model for analysis (see *e.g.*, Equation 3.2). Finally, our engine analyses the representational data model to extract user behavioural patterns that are used to assist users in achieving their personal household goals. Figure 3.2 briefly illustrates the overall steps of how our behavioural extraction engine works.

3.2.1 IoT Data Collection

We developed a python crawler to access and download the web pages that have dweet posts from users. We parse the DOM tree[4] of a web page to extract the information of the dweet posts. The extracted dweet posts are cleaned and processed for data analysis with the following steps:

- Remove any code snippets other than raw data field containers from each dweet posts (*i.e.*, statements not enclosed with tag "<raw>...</raw>"). We filter out such an information because it is not necessary to extract an user's behaviour.
- Remove any data fields whose values are not in a form to convert to a binary form from each dweet posts, as these fields are not informative to our study. For example, temperature data field (*i.e.*, 21°C).
- Apply the conversion mechanism that maps data field values to their binary form. For instance, OPEN and CLOSE are converted to "1" and "0".

3.2.2 IoT Data Repository Design

Each IoT smart device is associated with a user, environment (*e.g.*, Home), and a set of states (*e.g.*,OPEN or CLOSE). To capture all these elements, we model the Entity-Relationship (ER) model [16] that is shown in Figure 3.3. The IoT data repository stores the usage of each IoT device in the format of transactions.

[4]https://www.w3.org/TR/DOM-Level-2-Core/introduction.html

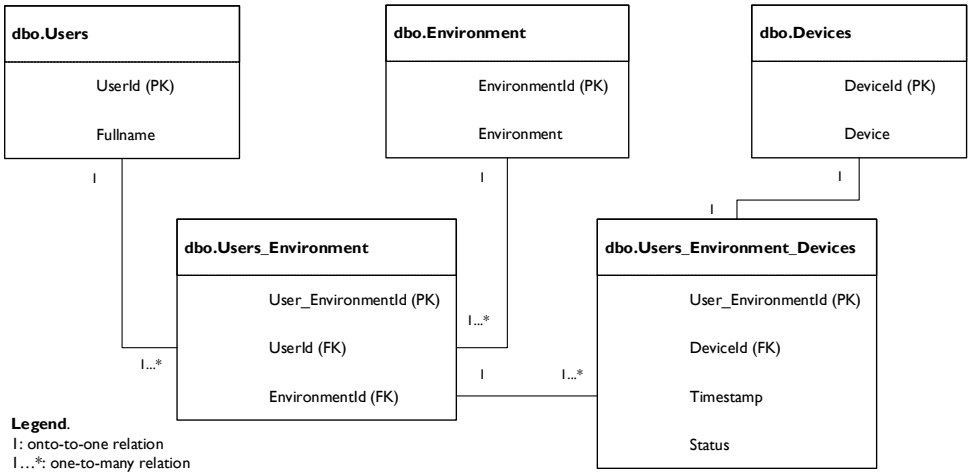

Figure 3.3: The entity-relationship diagram of IoT data repository.

Each transaction (T) in the repository is represented in the form of:

$$T = \{DeviceId, UserId, User\_EnvironmentId, Status, Time\_Stamp\} \tag{3.1}$$

where DeviceId is a unique identifier for an IoT device. UserId is the name of the user who owns the IoT device while User_EnvironmentId is the environment where a device resides. Status is the current status of the device and Time_Stamp denotes the recorded date and time of the transaction.

3.2.3 Data Representation

To prepare the data for analysis, we present the data in the form of matrix using Equation 3.2, where each row denotes the time (*i.e.*, 9am, 10am, *etc.*) and each column represents a day. The vector in each cell represents the list of all devices in the environment with its corresponding status at the given day and time, such as Monday at 9am: {Garage Door: OPEN, Foyer Light: ON, Basement Light: ON}.

Data Matrix Model =

$$\begin{bmatrix} & \mathbf{Day}_1 & \mathbf{Day}_2 & .. & \mathbf{Day}_n \\ \mathbf{T}_1 & (d1:ON, d2:ON, .., dn:OFF) & (d1:OFF, d2:ON, .., dn:OFF) & & (d1:ON, d2:ON, .., dn:OFF) \\ \mathbf{T}_2 & (d1:OFF, d2:OFF, .., dn:OFF) & (d1:ON, d2:ON, .., dn:OFF) & & (d1:OFF, d2:OFF, .., dn:OFF) \\ \mathbf{T}_3 & (d1:ON, d2:ON, .., dn:OFF) & (d1:ON, d2:ON, .., dn:OFF) & & (d1:ON, d2:OFF, .., dn:ON) \\ \vdots & \vdots & \vdots & & \vdots \\ \mathbf{T}_n & (d1:OFF, d2:ON, .., dn:ON) & (d1:ON, d2:ON, .., dn:OFF) & & (d1:ON, d2:OFF, .., dn:OFF) \end{bmatrix} \tag{3.2}$$

3.2.4 Behavioural Patterns Extraction

Our engine identifies the behavioural patterns of users' IoT device usage in a two-fold process. First, our engine measures each device usage by summarizing the frequency of possible device states at specific times in the environment. For example, Foyer lights are turned On 90% of times between 8am till noon. Second, our engine identifies the impact (*i.e.*, corresponding actions and relationships) among other IoT devices in the environment using Apriori algorithm. For example, the foyer lights are On between 8am till noon then the front door is Closed.

Apriori Algorithm. It is an association rule mining technique which mines if-then rules given a set of transactions [3]. For instance, given a set of transactions (Trans.) the rules are mined to form implication expressions in the form of if-then rules (*e.g.*, X1 $\implies$ Y1, where X1 and Y1 are disjoint itemsets on devices in our case). The formed if-then rules are extracted using the following two metrics: how often a rule is applicable in the given data and how frequently items in Y1 appears in transactions that contain X1 (*i.e.*, X1 and Y1 are IoT devices in the house-hold in our case).

Our engine's analysis steps are explained in depth in the upcoming research questions.

3.3 Case Study

In this section, we present our case study setup and results. We describe the motivation, the approach and the findings of our two research questions.

3.3.1 Case Study Setup

IoT Data Repository. We create the IoT data repository with users' IoT devices usage data. We use a well-known publishing and subscribing site for IoT devices (*i.e.*,dweet.io) to collect the IoT devices real-time usage data for 4 users. The collected IoT devices of all users are from smart home environments. Table 3.1 shows the detailed summarizes of the IoT devices data collected for each user. The table describes the list of devices for each user with its corresponding description and possible device states (*e.g.*, Open or Close).

3.3.2 Research Question

***RQ1.* What are the most used devices at a given time?**

Motivation. Controlling the devices of an environment requires a great effort from users because controlling each device may be very tiresome for users. Therefore, it is important to identify which devices are used the most by users at a given time. This analysis allows us to learn their habits of devices usage. Users can use this knowledge to minimize their effort on controlling the devices in their environment.

Table 3.1: List of studied IoT devices for personalized behavioural extraction.

IoT Devices	States
(User 1): Ranipet House	
(D1). Front Door: provides the status of the front door.	(OPEN|CLS)
(D2). Outside Light: shows the current status of the outside light.	(ON|OFF)
(D3). Hall Light: shows the current status of the hall room light.	(ON|OFF)
(D4). Home Fan: shows the running status of the fan.	(ON|OFF)
(User 2): Alastair House	
(D1). Front Door: provides the status of the house's front door.	(OPEN|CLS)
(D2). Back Door: shows whether the back door is open or not.	(OPEN|CLS)
(D3). Garage Door: shows whether the garage door is open or not.	(OPEN|CLS)
(User 3): Azer House	
(D1). Family Room: provides the status of the family room light.	(ON|OFF)
(D2). Kitchen: indicates the status of the kitchen lights.	(ON|OFF)
(D3). Bedroom: indicates the status of the master bedroom light.	(ON|OFF)
(D4). Foyer: provides the status of the foyer room light.	(ON|OFF)
(User 4): Laabs House	
(D1). Bedroom L: shows the status of the bedroom light.	(ON — OFF)
(D2). Track Light Front: shows the status of the front track lights.	(ON — OFF)
(D3). Track Light Rear: shows the status of the back track lights.	(ON — OFF)
(D4). Downstairs L: shows the status of the downstairs hall light.	(ON — OFF)
(D5). Upstairs Light: shows the status of the upstairs hall light.	(ON — OFF)
(D6). Shower: shows the status of the master bathroom shower.	(ON — OFF)
(D7). Vanity L: the master bathroom vanity light status.	(ON — OFF)
(D8). Bathroom: shows whether the bathroom light is on or not.	(ON — OFF)
(D9). Washer: indicate whether the garage washer is active or not.	(ON — OFF)
(D10). Nursery Floor: the nursery lamp status.	(ON — OFF)
(D11). Porch: shows whether the porch front light is on or not.	(ON — OFF)
(D12). Entryway: indicates the status of the entry way hall light.	(ON — OFF)
(D13). Bedroom CL: shows the state of the bedroom corner lamp.	(ON — OFF)
(D14). Floor Lamp: indicates the hall floor lamp status.	(ON — OFF)
(D15). Kitchen: current status of the kitchen room light.	(ON — OFF)
(D16). Entry Table: hallway entry table lamp status.	(ON — OFF)
(D17). Downstairs Fan: downstairs bathroom fan status.	(ON — OFF)
(D18). Bathroom Fan: shows the master bathroom fan status.	(ON — OFF)
(D19). Bed LED: the master bedroom underbed LED strip status.	(ON — OFF)
(D20). Garage L: the garage fluorescent light status.	(ON — OFF)

Note:- CLS denotes the device state Close.

Analysis Approach. To identify which devices are used the most on specific times, we perform the following steps.

First, for each user, we arrange the IoT devices data in the IoT repository in a form of data matrix model as outlined in Section 3.2.3. In the data matrix model, each row represents the devices state in the environment at a specific time (*e.g.*, 9am) and each column represent the day of the week (*e.g.*, Sunday). Second, to measure the devices usage, we introduce a new metric DFreq (see Equation 3.3) to calculate the number of times the device remains in a specific state at given times.

$$DFreq = \frac{\sum_{Time=1}^{n}(\sum_{Day=1}^{m} DeviceStatus_{Day} : State)}{Total\ number\ of\ days} \tag{3.3}$$

*where the State represents the status of a device (*i.e.*, either On or Off, or Open or Close). In this research question, we aim to study the devices that are the most used (*i.e.*, device state either On or Open). Therefore, by default we set the State variable of the Equation 3.3 to be 1 (*i.e.*, device state either ON or Open).*

Third, to identify the devices that are used by users at specific times, we filter the data matrix by applying a 80/20 Pareto's principle [67] (*i.e.*, DFreq>=80% confidence interval threshold), where the 80% of data represents the remaining 20% of data population. In this way, we derive the most used devices by users on specific times.

Results. For each user, only 2 devices are being used the most in the environment at specific times. The most used devices are shown in Table 3.2. For instance, there are 2 out of 4 devices that are majorly used by User 1. Those

Table 3.2: List of the most used IoT devices per users for rule extraction.

IoT devices	**Time-frames**	
	Weekdays	**Weekends**
(User 1)		
(D3). Hall Light	8:30 to 12:15pm	8:45am to 11:45am
(D4). Home Fan	8:30 to 11:30am	9:45am to 10:45am
(User 2)	No rule can be concluded.	
(User3)		
(D1). Family Room	1:15 to 1:30am	1 to 1:15am; 7:45 to 8am;
(D2). Kitchen	No rule can be concluded.	1:15pm to 1:45pm; 3:15pm to 4:30pm; 10 to 10:15pm
(D3). Bedroom	6:45pm to 7pm	3 to 3:15pm
(D4). Foyer	7:45pm to 8pm	3 to 3:15am; 12:15 t0 12:45pm
(User 4)		
(D1). Bedroom Light	10:45 to 11:30am	1:15am to 2:30am
(D2). Track Light Front	12pm to 12:15pm; 4pm to 4:15pm; 8:30pm to 3am	12am to 3am; 8:45pm to 12am
(D3). Track Light Rear	12pm to 12:30pm; 8:15 to 3am	12am to 3am; 8:45pm to 12am
(D11). Porch	10:15pm to 2:30am;	12 to 3am; 10:15pm to 12am
(D13). Bedroom CL	1:45 to 2am; 11am to 11:45am	12am to 2:30am; 10:30pm to 12am
(D15). Kitchen	12am to 11:45pm	12am to 11:45pm

devices are the Hall Light and Home Fan which are being used during Weekends from 8:45am to 11:45am and 9:45am to 10:45am, respectively.

In particular, we observe that for User 2 we cannot identify which devices that are

mostly used, since those devices do not achieve DFreq>=80% in the studied period. Therefore, we eliminate those devices from further analysis.

The results presented in Table 3.2 shows the users' behavioural usage pattern for the most used devices only, we show the usage patterns for Weekdays and Weekends. However, if desired, the same technique could be applied across different time-frames to learn the behavioural patterns of users, such as each day of the week (*e.g.*, Monday), monthly (*e.g.*, April), and seasonally (*i.e.*, summer, fall, winter, and spring). The behavioural patterns can be used to help users by automatically taking actions on their most used devices. For example, our behavioural pattern for User1 shows that his/her Hall light is ON during Weekdays from 8:30 to 12:15pm. We can use this learned behavioural pattern to automatically turn ON the Hall light during this time-frame to assist User1.

On average, users' use less than 50% of their IoT devices at specific times in the environment.

RQ2. ***What is the relationship between the most used devices and the other devices in the environment?***

Motivation. In a given environment, users tend to use multiple IoT devices to achieve their personal goals. From RQ1, we observe the IoT devices that are mostly used by each user at specific times. However, these most used devices may impact the usage of other devices. For example, whenever the garage door closes the kitchen lights are turned ON. Therefore, in RQ2 we aim to study the impact (*i.e.*, corresponding actions and relationships) on the other devices in the environment.

Users can use learned impact insights to automatically take actions of the devices in the environment.

Analysis Approach. To identify the impact that the most used IoT devices have on the other devices in the environment, we execute the Apriori algorithm on our IoT repository data for each user. We identify the implications of the most used devices on the other devices in the environment as follows:

- We arrange the data in the IoT data repository for each user as shown in Equation 3.2. In particular, the data matrix rows represent the time-frame and its corresponding device states and columns represent the day of the week.

- We use the data matrix as input to the Apriori algorithm and the algorithm produces the set of implication rules (*i.e.*, patterns). For example, the produced pattern is {Kitchen Light: ON $\implies$ Front Door: Closed}. Right-hand side and Left-hand side of the pattern is known as consequents and antecedents, respectively.

- Based on the most used devices that we identify in RQ1, we filter the produced implication patterns where the antecedents of the expression belong to the most used devices.

- Further, we compute two metrics to measure the strength of the rules, the metrics are Support and Confidence. The Support (S) and Confidence (C) metrics are measured using the Equations 3.4 and 3.5, respectively.

$$S\{X1 \implies Y1\} = \frac{\{\#\ of\ times\ a\ pattern\ exists\ in\ the\ given\ dataset\}}{\{Total\ \#\ of\ transactions\ in\ the\ given\ dataset\}} \tag{3.4}$$

$$C\{X1 \implies Y1\} = \frac{\{\#\ of\ times\ a\ pattern\ exists\ in\ the\ given\ dataset\}}{\{Total\ \#\ of\ transactions\ where\ X1\ exists\ in\ the\ given\ dataset\}} \tag{3.5}$$

- Finally, we extract the patterns for which the support and confidence values are greater than a half of the transactions using the Confidence Interval threshold (*i.e.*, S{X1 $\implies$ Y1}>=50% and C{X1 $\implies$ Y1}>=50%).

Results. A strong implications patterns are inferred for each user. The identified implication patterns are shown in Table 3.3. For example, during weekends, User 1 use his/her Hall Light and Home Fan between 8:30 to 11:30am when the Front Door is closed on the majority of the time (*i.e.*, 70%).

Table 3.3: The results of the implication rules produced by Apriori algorithm.

List of devices	**Weekdays**			**Weekends**		
	Implication	**(S)**	**(C)**	**Implication**	**(S)**	**(C)**
(User 1)						
Hall Light:ON, Home Fan:ON	8:30 to 11:30am Outside Light:OFF	58.33%	82.35%	8:30 to 11:30am Front Door:Closed	70.00%	87.50%
(User 2)	No strong impliction rules can be infered.					
(User 3)						
Family Room:ON	1 to 1:15am			7:45 to 8am		

	Kitchen: ON	50.00%	83.40%	Foyer: ON	80.00%	80.00%
				1 to 1:15am		
				Kitchen:OFF	80.00%	80.00%
Foyer:ON	7:45pm to 8pm			3 to 3:15am		
	Family Room:OFF	50.00%	85.71%	Bedroom:OFF	60.00%	100.00%
Kitchen: ON				1:15pm to 1:45pm		
				Foyer: ON	60.00%	100.00%
				3:15pm to 4:30pm		
				Foyer: OFF	80.00%	81.00%
(User 4)						
Kitchen:ON, Bedroom L:ON, Bedroom CL: ON	10:45 am to 11:30 am			No strong rules can be inferred.		
	Garage L: ON	73.08%	90.48%			
	Washer: ON	73.08%	90.48%			
	Shower: OFF	73.08%	90.48%			
	Downstairs L: ON	80.77%	100.00%			
	Bathroom: OFF	76.92%	95.24%			
	Bathroom Fan: OFF	76.92%	95.24%			
	Bed LED: OFF	80.77%	100.00%			
	Downstairs L:OFF	80.77%	100.00%			
	Entry Table: OFF	80.77%	100.00%			
	Floor Lamp: OFF	80.77%	100.00%			
	Entryway: OFF	80.77%	100.00%			
	Porch: OFF	80.77%	100.00%			
	Upstairs Light: OFF	80.77%	100.00%			
Track Light Front:ON, Track Light Rear: ON, Kitchen:ON	12pm to 12:15pm			No strong rules can be inferred.		
	Downstairs L: OFF	61.54%	88.89%			
	Bed LED: OFF	69.23%	100.00%			
	Bathroom Fan: OFF	69.23%	100.00%			
	Entry Table: OFF	69.23%	100.00%			
	Floor Lamp: OFF	69.23%	100.00%			

Track Light Front:ON, Kitchen: ON	Entryway: OFF	69.23%	100.00%	
	4pm to 4:15pm			No strong rules can be inferred.
	Track Light Rear: ON	69.23%	90.00%	
	Nursery Floor: OFF	65.38%	85.00%	
	Garage L: OFF	73.08%	95.00%	
	Shower: OFF	69.23%	90.00%	
	Washer: OFF	73.08%	95.00%	
	Downstairs L: OFF	73.08%	95.00%	
	Bathroom: OFF	76.92%	100.00%	
	Porch: OFF	73.08%	95.00%	
	Bed LED: OFF	76.92%	100.00%	
	Bathroom Fan: OFF	76.92%	100.00%	
	Downstairs Fan: OFF	76.92%	100.00%	
	Entry Table: OFF	76.92%	100.00%	
	Floor Lamp: OFF	76.92%	100.00%	
	Entryway: OFF	76.92%	100.00%	
	Bathroom L: OFF	76.92%	100.00%	
	Upstairs Light: OFF	76.92%	100.00%	

Notes:- (S) and (C) denotes the Support and Confidence of the rule.

We observe that implication patterns with a strong support and confidence metrics can be used in two ways: 1) alert the user about the abnormalities of device behavior in the environment (*e.g.*, intimate the user to shut off the shower when bathroom lights are off to conserve water consumption), or 2) to make smart propagation of certain actions across the devices in the environment (*e.g.*, when the Hall Light and Fan are being used then the system should automatically close the Front door.).

*A strong user's behavioural impacts exists among devices which can be used to make smart recommendations (*i.e.,*actions) across the other devices.*

3.4 Threats To Validity

In this section, we discuss the threats to validity of our study through a common guideline [86]:

3.4.1 Internal Validity

An internal threat to validity is that we only focus on IoT devices data were posted on the dweet.io publisher-subscription site. The quality of IoT devices data being published might affect the experimental results. To deal with the possible bias, the first author of this work manually verified all the data collected from IoT smart devices to make sure it has the appropriately described data fields.

3.4.2 External Validity

An external threat to validity is that we only studied 31 devices from 4 unique users. There are over 2,500 devices available on dweet.io site. Since, our study is focused on discovering knowledge of personalized behaviours in home settings, we analyzed all the available devices and eliminated the devices belonging to industrial settings. Nevertheless, further studies using more IoT devices are welcome.

3.4.3 Construct Validity

A construct threat to validity is that the IoT devices data that is used in our study is based on the collection of 30 consecutive days. To remove any possible bias, a longer period of data collection should be performed.

3.5 Summary

The increasing popularity of the smart devices and its connectivity to the Internet of Things (IoTs) platform created millions of IoT devices. Users use those IoT devices to achieve their personal goals. In this chapter, we provide a behavioural extraction engine using Apriori, an association rule mining algorithm, to identify the most used devices by users and to find their relationships with other devices in the environment. In particular, our engine can be used to identify the personalized user behaviour with respect to their devices usage patterns that can be used to alert or make smart recommendations to users in achieving their personal goals.

We conduct a case study from the users' IoT devices usage data collected from dweet.io (*i.e.*, a popular site for publishing and subscribing data of IoT devices) for 4 users. Our results show that, users have on average, 2 IoT devices that they use at specific times and have a relatively small impact across other devices in the environment. Hence, the assimilated users IoT devices behavioural patterns can be used to communicate with users to notify when the abnormality of the device behaviour happens and/or to make smart recommendations; or to propagate actions across devices in the environment automatically, without any user intervention.

Chapter 4

A Personalized Assistant Framework for Service Recommendation

In this chapter, we propose an intellectually cognitive personalized assistance framework to achieve user goals. Our framework using a learning-to-rank machine learning algorithm improves the user service selection experience by asking them the most relevant and necessary questions of their intended goals to help narrow down the search space of identifying the most preferred services. We perform a case study to evaluate the effectiveness of our framework. On average, our framework, utilizing the learning-to-rank algorithm improves the nine baseline approaches 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 services faster.

4.1 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 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 [77, 87].

Extensive research has been conducted to help users select desired services (*e.g.*, [76, 5, 15, 85]). For example, some research (*e.g.*, [5, 76]) proposes filtering-based approaches to help users navigate through a relatively huge volume of services to select relevant services. Some other research (*e.g.*, [15, 85]) 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

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

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 the previous chapter, we showed how to extract the behavioural patterns of IoT device users using machine learning techniques to reduce their cognitive overload. However, the space of machine learned context-aware interactions with the user was not explored. In this chapter, 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

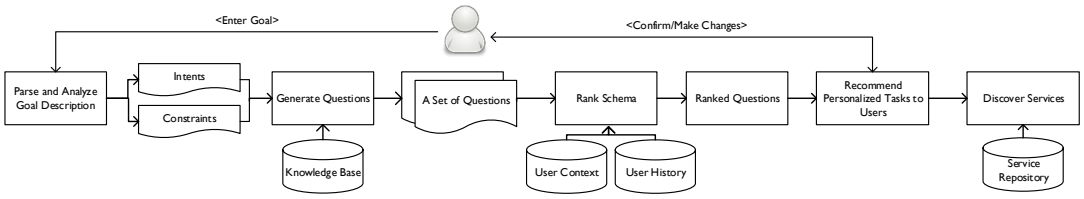

Figure 4.1: An overview of our personalized assistance framework.

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 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.

4.2 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 4.1 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 4.2 briefly illustrates the overall steps of our framework for achieving the goal "I want to find accommodation".

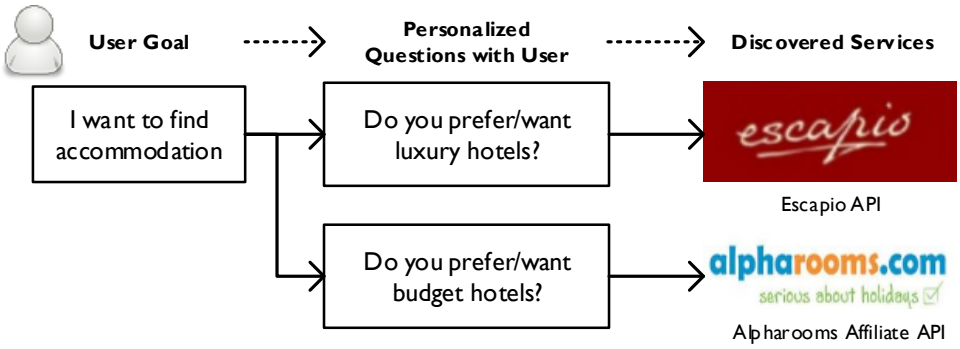

Figure 4.2: An example overview of our framework.

4.2.1 Creating Process Knowledge Bases

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 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 publicly available websites.

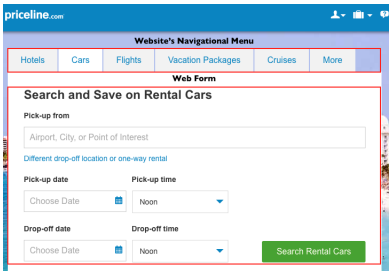

Figure 4.3: An annotated website structure from priceline.com.

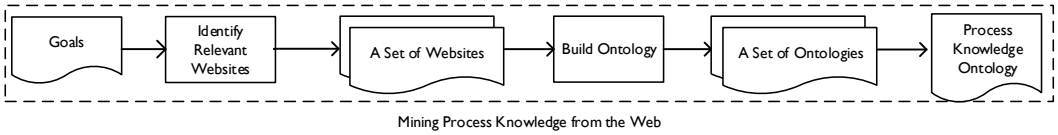

Figure 4.4: An overall process of creating knowledge base.

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 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 4.3 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 [83]. We use the following steps as shown in Figure 4.4 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 [83]. Second, to capture the concrete process knowledge for the given goal, we utilize the approach in [83]. The approach automatically retrieves the process knowledge

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

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.

Table 4.1: 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 Context (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 Context (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.

4.2.2 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.* [87]. The approach utilizes the start-of-the-art Stanford natural language processing techniques[3] to analyze the user goal and extract 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.

4.2.3 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 work, 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.

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

Initially, we search and identify the required process knowledge ontology for a given user goal from a set of our stored ontologies in Section 4.2.1. 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 [64] If any questions have the similarity scores higher than 90%, we

rank them in a descending order 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 4.1 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 [37]. Later, Lee et al. [48] 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 [70].

Therefore, we build 15 learning features to capture the changes in individual contexts.

[C3.] Service Context. 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 service 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 [84].

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 4.5 shows an example of the transitions in the Logical Graph.

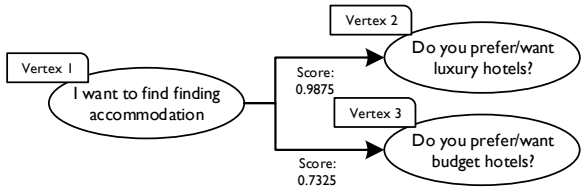

Figure 4.5: An example representation in Logical Graph.

4.2.4 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 [53].

2. Use WordNet[4] to remove all non-English words.

3. Apply the Porter stemming to map words to their base form [66]. *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 [56]. 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 [49] is a popular semantic database which

[4]https://wordnet.princeton.edu/
[5]http://lucene.apache.org/

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" [73]. 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.

4.3 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.

4.3.1 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.*,programmable web[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 4.2 shows the statistics of our collected services.

Table 4.2: 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

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

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.

4.3.2 *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 most preferred services. It is critical to evaluate the performance of our framework with regards to recommend questions which aid users to discover their most preferred 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.

2. *Logical Graph Ranking*: We utilize the Logical Graph in Section 4.2.3 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 4.1.

$$Score_{(V_i,V_j)} = \sum_{u=1}^{n} SC(S_{u...n}) \tag{4.1}$$

   We normalize the obtained score into the range of [0 – 1]. In Equation 4.1, 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 the 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 [59].

4. *ListNet*: is a listwise LtR technique using a neural network approach with gradient descent to minimize a loss in the ranking function [14].

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 [12]. 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 [29].

7. *LambdaMART*: is an ensemble LtR method consisting of boosted regression trees in a combination with LamdaRank [81]. LambaRank is a neural network algorithm similar to RankNet algorithm [12].

8. *RankBoost*: is a pairwise LtR technique that aims to select a set of weak rankers to build a strong ranker function [28].

9. *Coordinate Ascent*: is a listwise LtR approach optimizing rankers by minimizing a specific loss function [58].

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 (4.2) and Equation (4.3), respectively.

$$Precision@k = \frac{\{\#\ of\ answered\ questions\}}{K} \tag{4.2}$$

$$Recall@k = \frac{\{\#\ of\ answered\ questions\}}{\{\#\ of\ questions\ should\ be\ answered\}} \tag{4.3}$$

where K is the number of recommended questions. In this work, 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.

Table 4.3: Evaluation results of the ten Learning-to-Rank algorithms for recommending personalized questions to users help achieve their personal goals.

	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

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.

AdaRank algorithm performs better than the other outlined baselines. In particular, performs 24.42% better than the popular graph-based algorithm (*i.e.*, PageRank) at precision@1. Intuitively, the common question raised by authors is that how our AdaRank performs better than the well-known and widely used PageRank algorithm. We perform the investigation and it turns out that PageRank algorithm computes score at each node based on its number of subsequent child nodes. However, AdaRank measures score at each node based on its immediate child nodes. In our case, since the number of nodes (*i.e.*, questions) available in any given scenario is two-levels deep so the AdaRank performs better than the PageRank algorithm. Contrastively, if we build our knowledge base on a real-time and if there exist more levels of questions, then PageRank algorithm may perform better at recommending personalized questions to users to achieve their personal goals.

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.

Table 4.4: 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.

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

4.3.3 *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 (4.2) 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 4.4 to calculate the importance of a learning feature.

$$Importance = (P@1\ in\ RQ1)\ \ minus\ \ (P@1\ in\ RQ2) \tag{4.4}$$

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.

Table 4.5: Results of feature importance in AdaRank learning technique performance. 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%

Results.

Our results in Table 4.5 show that among three main categories, "User Context" is the most important one, 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 4.4 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 Context is the most important learning feature category that affects the performance of our framework the most.

4.3.4 *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 4.6 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 4.6 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 4.6: 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.

4.4 Summary

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 work, 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.

Chapter 5

Summary and Future Work

In this chapter, we describe the summary and contributions of the thesis in Section 5.1. Finally, the possible directions for future work are presented in Section 5.2.

5.1 Summary and Contributions

The overall goal of this thesis is to realize the potential of personalized software assistant system in a user-centric model by studying the user behaviour using machine learning techniques. First, we propose an engine that analyzes and identifies the behavioural patterns of IoT device users to make smart recommendations to automatically control users' IoT devices in smart-home environments. Then, we create an intellectually cognitive personalized assistant framework, using the learning-to-rank algorithm, namely Adarank, to recommend service through personalized context-aware interactions with users to help them achieve their personal goals. We summarize the main contributions of this thesis as follows:

1. We proposed an approach to reduce IoT device users' cognitive overload by learning personalized behavioural patterns. So that, the personalized software systems can make intelligent recommendations to automatically control users' devices in smart-home environments.

2. We proposed an intelligently cognitive personalized assistant approach to engage in personalized context-aware interactions with users to help them narrow down the search space of service selections to achieve their personal goals.

5.2 Future Work

We believe that this thesis makes a major contribution towards realizing the potential of personalized software assistant in a user-centric model to make smart recommendations on IoT device usage and to have personalized context-aware interactions with

users. However, there are many other open challenges and opportunities for further improvements. In this section, we discuss the possible future directions.

5.2.1 Combining IoT Devices of Different Environments

In this thesis, the studied IoT devices are purely of consumer devices in the smart home environment. Our results may not generalize for IoT devices in different environmental settings, such as manufacturing. Our study can be extended to include more types of IoT resources, such as industrial IoT devices.

5.2.2 Creation of Rich Process Knowledge

In this thesis, we create the process knowledge from online sites only. However, a large variety of other knowledge sources exist for process knowledge creations, such as How-to instruction pages. For example, wikiHow[1], the How-to instructions site currently store millions of articles on how to do things step by step, which covers almost every domain of our lives, such as business, travel, health, finance, food, entertainment and education. In the future work, we plan to create larger process knowledge with more types of on-line resources.

5.2.3 Integration of Our Approaches in Real-world Systems

We plan to apply our approach to develop a more robust user-friendly application to help users achieve their personal goals.

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

Bibliography

[1] Woogle–web-service search engine. `http://db.cs.washington.edu/projects/woogle.html`. (Accessed on 07/18/2017).

[2] Charu C Aggarwal. Outlier analysis. In *Data mining*, pages 237–263. Springer, 2015.

[3] Rakesh Agrawal, Ramakrishnan Srikant, et al. Fast algorithms for mining association rules. In *Proc. 20th int. conf. very large data bases, VLDB*, volume 1215, pages 487–499, 1994.

[4] Frank C Arnett, Steven M Edworthy, Daniel A Bloch, Dennis J Mcshane, James F Fries, Norman S Cooper, Louis A Healey, Stephen R Kaplan, Matthew H Liang, Harvinder S Luthra, et al. The american rheumatism association 1987 revised criteria for the classification of rheumatoid arthritis. *Arthritis & Rheumatology*, 31(3):315–324, 1988.

[5] Wolf-Tilo Balke and Matthias Wagner. Towards personalized selection of web services. In *WWW (Alternate Paper Tracks)*, pages 20–24, 2003.

[6] Philip Bianco, Rick Kotermanski, and Paulo F Merson. Evaluating a service-oriented architecture. 2007.

[7] Gilles Bisson, Claire Nédellec, and Dolores Canamero. Designing clustering methods for ontology building-the mo'k workbench. In *ECAI workshop on ontology learning*, volume 31. Citeseer, 2000.

[8] M Brian Blake and Michael F Nowlan. A web service recommender system using enhanced syntactical matching. In *IEEE ICWS*, pages 575–582, 2007.

[9] Leo Breiman. Bias, variance, and arcing classifiers. 1996.

[10] Tom Broens, Stanislav Pokraev, Marten Van Sinderen, Johan Koolwaaij, and Patricia Dockhorn Costa. Context-aware, ontology-based service discovery. In *European Symposium on Ambient Intelligence*, pages 72–83. Springer, 2004.

[11] Paul Buitelaar, Daniel Olejnik, and Michael 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.

[12] Chris Burges, Tal Shaked, Erin Renshaw, Ari Lazier, Matt Deeds, Nicole Hamilton, and Greg Hullender. Learning to rank using gradient descent. In *Proceedings of the 22nd ICML*, pages 89–96.

[13] Christopher J Burges, Robert Ragno, and Quoc V Le. Learning to rank with nonsmooth cost functions. In *Advances in neural information processing systems*, pages 193–200, 2007.

[14] Zhe Cao, Tao Qin, Tie-Yan Liu, Ming-Feng Tsai, and Hang Li. Learning to rank: from pairwise approach to listwise approach. In *Proceedings of the 24th ICML*, pages 129–136.

[15] Irene YL Chen, Stephen JH Yang, and Jia Zhang. Ubiquitous provision of context aware web services. In *IEEE SCC*, pages 60–68, 2006.

[16] Peter Pin-Shan Chen. The entity-relationship modeltoward a unified view of data. *ACM Transactions on Database Systems (TODS)*, 1(1):9–36, 1976.

[17] Xi Chen, Xudong Liu, Zicheng Huang, and Hailong Sun. Regionknn: A scalable hybrid collaborative filtering algorithm for personalized web service recommendation. In *IEEE ICWS*, pages 9–16, 2010.

[18] Yen-Liang Chen, Kwei Tang, Ren-Jie Shen, and Ya-Han Hu. Market basket analysis in a multiple store environment. *Decision support systems*, 40(2):339–354, 2005.

[19] Erik Christensen, Francisco Curbera, Greg Meredith, Sanjiva Weerawarana, et al. Web services description language (wsdl) 1.1, 2001.

[20] Philipp Cimiano and Johanna Völker. text2onto. In *International Conference on Appl. of Natural Language to Information Sys.*, pages 227–238, 2005.

[21] Corinna Cortes and Vladimir Vapnik. Support vector machine. *Machine learning*, 20(3):273–297, 1995.

[22] Koby Crammer and Yoram Singer. Pranking with ranking. In *Advances in neural information processing systems*, pages 641–647, 2002.

[23] Marco Antônio Pinheiro de Cristo, Pavel Pereira Calado, Maria de Lourdes Da Silveira, Ilmério Silva, Richard Muntz, and Berthier Ribeiro-Neto. Bayesian belief networks for ir. *International Journal of Approximate Reasoning*, 34(2-3):163–179, 2003.

[24] Lisa Dent, Jesus Boticario, John P McDermott, Tom M Mitchell, and David Zabowski. A personal learning apprentice. In *AAAI*, pages 96–103, 1992.

[25] Thomas Erl. Service-oriented architecture (soa): concepts, technology, and design, 2005.

[26] Usama M Fayyad, Gregory Piatetsky-Shapiro, Padhraic Smyth, and Ramasamy Uthurusamy. *Advances in knowledge discovery and data mining*, volume 21. AAAI press Menlo Park, 1996.

[27] Roy T Fielding and Richard N Taylor. *Architectural styles and the design of network-based software architectures*. University of California, Irvine Doctoral dissertation, 2000.

[28] Yoav Freund, Raj Iyer, Robert E Schapire, and Yoram Singer. An efficient boosting algorithm for combining preferences. *Journal of machine learning research*, 4(Nov):933–969, 2003.

[29] Jerome H Friedman. Greedy function approximation: a gradient boosting machine. *Annals of statistics*, pages 1189–1232, 2001.

[30] W James Gauderman. Candidate gene association analysis for a quantitative trait, using parent-offspring trios. *Genetic epidemiology*, 25(4):327–338, 2003.

[31] John Gekas. Web service ranking in service networks. In *the 3rd European Semantic Web Conference (ESWC 2006), Budva*, 2006.

[32] John Gekas and Maria Fasli. Automatic web service composition using web connectivity analysis techniques. In *W3C Workshop on Frameworks for Semantics in Web Services*, pages 9–10, 2005.

[33] Alexander Gluhak, Srdjan Krco, Michele Nati, Dennis Pfisterer, Nathalie Mitton, and Tahiry Razafindralambo. A survey on facilities for experimental internet of things research. *IEEE Communications Magazine*, 49(11), 2011.

[34] Martin Gudgin. Soap version 1.2 part 1: Messaging framework. *http://www.w3. org/TR/soap12-part1/*, 2003.

[35] Lynne Hamill. Controlling smart devices in the home. *The Information Society*, 22(4):241–249, 2006.

[36] Jiawei Han, Jian Pei, and Micheline Kamber. *Data mining: concepts and techniques*. Elsevier, 2011.

[37] Stephen José Hanson, Robert E Kraut, and James M Farber. Interface design and multivariate analysis of unix command use. *ACM Transactions on Information Systems*, 2(1):42–57, 1984.

[38] Ralf Herbrich, Thore Graepel, and Klaus Obermayer. Large margin rank boundaries for ordinal regression. 2000.

[39] H. Hromic, D. Le Phuoc, M. Serrano, A. Antoni, I. P. arko, C. Hayes, and S. Decker. Real time analysis of sensor data for the internet of things by means

of clustering and event processing. In *2015 IEEE International Conference on Communications (ICC)*, pages 685–691, June 2015.

[40] N. Jennings and M. Wooldridge. Software agents. *IEE Review*, 42(1):17–20, Jan 1996.

[41] Nicholas R Jennings, Katia Sycara, and Michael Wooldridge. A roadmap of agent research and development. *Autonomous agents and multi-agent systems*, 1(1):7–38, 1998.

[42] B. Kantarci and H. T. Mouftah. Trustworthy sensing for public safety in cloud-centric internet of things. *IEEE Internet of Things Journal*, 1(4):360–368, Aug 2014.

[43] Sean Dieter Tebje Kelly, Nagender Kumar Suryadevara, and Subhas Chandra Mukhopadhyay. Towards the implementation of iot for environmental condition monitoring in homes. *IEEE Sensors Journal*, 13(10):3846–3853, 2013.

[44] Jacek Kopeckỳ, Karthik Gomadam, and Tomas Vitvar. hrests: An html microformat for describing restful web services. In *Web Intelligence and Intelligent Agent Technology, 2008. WI-IAT'08. IEEE/WIC/ACM International Conference on*, volume 1, pages 619–625. IEEE, 2008.

[45] R Kozierok and P Maes. Intelligent groupware for scheduling meetings. *Submitted to CSCW-92*, 1992.

[46] Solomon Kullback. *Information theory and statistics*. Courier Corporation, 1997.

[47] Andreas Heß Nicholas Kushmerick. Automatically attaching semantic metadata to web services. 2003.

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

[49] Hugo Liu and Push Singh. Conceptneta practical commonsense reasoning toolkit. *BT technology journal*, 22(4):211–226, 2004.

[50] Yaling Liu and Arvin Agah. Crawling and extracting process data from the web. In *Conf. on Adv. Data Mining and App.*, pages 545–552, 2009.

[51] Bing Liu Wynne Hsu Yiming Ma and Bing Liu. Integrating classification and association rule mining. In *Proceedings of the 4th*, 1998.

[52] Zakaria Maamar, Soraya Kouadri Mostefaoui, and Qusay H Mahmoud. Context for personalized web services. In *System Sciences,Proceedings of the IEEE International Conference on*, pages 166b–166b, 2005.

[53] Christopher D Manning, Prabhakar Raghavan, Hinrich Schütze, et al. *Introduction to information retrieval*, volume 1. Cambridge university press Cambridge, 2008.

[54] David Martin, Mark Burstein, Jerry Hobbs, Ora Lassila, Drew McDermott, Sheila McIlraith, Srini Narayanan, Massimo Paolucci, Bijan Parsia, Terry Payne, et al. Owl-s: Semantic markup for web services. *W3C member submission*, 22:2007–04, 2004.

[55] Elio Masciari. A framework for outlier mining in rfid data. In *Database Engineering and Applications Symposium, 2007. IDEAS 2007. 11th International*, pages 263–267. IEEE, 2007.

[56] Michael McCandless, Erik Hatcher, and Otis Gospodnetic. *Lucene in Action: Covers Apache Lucene 3.0*. Manning Publications Co., 2010.

[57] Sheila A McIlraith, Tran Cao Son, and Honglei Zeng. Semantic web services. *IEEE intelligent systems*, 16(2):46–53, 2001.

[58] Donald Metzler and W Bruce Croft. Linear feature-based models for information retrieval. *Information Retrieval '07*, 10-3:257–274.

[59] Rada Mihalcea. Graph-based ranking algorithms for sentence extraction, applied to text summarization. In *ACL 2004*, page 20.

[60] A. M. Ortiz, D. Hussein, S. Park, S. N. Han, and N. Crespi. The cluster between internet of things and social networks: Review and research challenges. *IEEE Internet of Things Journal*, 1(3):206–215, June 2014.

[61] Massimo Paolucci, Takahiro Kawamura, Terry R Payne, and Katia Sycara. Semantic matching of web services capabilities. In *International Semantic Web Conference*, pages 333–347. Springer, 2002.

[62] Mike P Papazoglou. Agent-oriented technology in support of e-business. *Communications of the ACM*, 44(4):71–77, 2001.

[63] Charith Perera, Arkady Zaslavsky, Peter Christen, and Dimitrios Georgakopoulos. Context aware computing for the internet of things: A survey. *IEEE Communications Surveys & Tutorials*, 16(1):414–454, 2014.

[64] Mohammad Taher Pilehvar and Roberto Navigli. From senses to texts: An all-in-one graph-based approach for measuring semantic similarity. *Artificial Intelligence*, 228:95–128, 2015.

[65] Christian Platzer and Schahram Dustdar. A vector space search engine for web services. In *Web Services, 2005. ECOWS 2005. Third IEEE European Conference on*, pages 9–pp. IEEE, 2005.

[66] Martin F Porter. An algorithm for suffix stripping. *Program*, 14(3):130–137, 1980.

[67] F John Reh. Pareto's principle-the 80-20 rule. *BUSINESS CREDIT-NEW YORK THEN COLUMBIA MD-*, 107(7):76, 2005.

[68] Norman Sadeh. Look-ahead techniques for micro-opportunistic job shop scheduling. Technical report, CARNEGIE-MELLON UNIV PITTSBURGH PA SCHOOL OF COMPUTER SCIENCE, 1991.

[69] Richa Saxena, Benjamin F Voight, Valeriya Lyssenko, Noël P Burtt, Paul IW de Bakker, Hong Chen, Jeffrey J Roix, Sekar Kathiresan, Joel N Hirschhorn, Mark J Daly, et al. Genome-wide association analysis identifies loci for type 2 diabetes and triglyceride levels. *Science*, 316(5829):1331–1336, 2007.

[70] Hinrich Schütze. Introduction to information retrieval. In *Proceedings of the ACM international conference on computing machinery*, 2008.

[71] Anuj Sehgal, Vladislav Perelman, Siarhei Kuryla, and Jurgen Schonwalder. Management of resource constrained devices in the internet of things. *IEEE Communications Magazine*, 50(12), 2012.

[72] So Young Sohn and Sung Ho Lee. Data fusion, ensemble and clustering to improve the classification accuracy for the severity of road traffic accidents in korea. *Safety Science*, 41(1):1–14, 2003.

[73] Robert Speer and Catherine Havasi. Representing general relational knowledge in conceptnet 5. In *LREC*, pages 3679–3686, 2012.

[74] Melanie Swan. Sensor mania! the internet of things, wearable computing, objective metrics, and the quantified self 2.0. *Journal of Sensor and Actuator Networks*, 1(3):217–253, 2012.

[75] Pang-Ning Tan, Michael Steinbach, and Vipin Kumar. Association analysis: basic concepts and algorithms. *Introduction to data mining*, pages 327–414, 2005.

[76] Bipin Upadhyaya, Foutse Khomh, Ying Zou, Alex Lau, and Joanna Ng. A concept analysis approach for guiding users in service discovery. In *5th IEEE International Conference on SOCA*, pages 1–8.

[77] Bipin Upadhyaya, Ying Zou, Shaohua Wang, and Joanna Ng. Automatically composing services by mining process knowledge from the web. In *International Conference on Service-Oriented Computing*, pages 267–282, 2013.

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

[79] H Vestberg. Ceo to shareholders: 50 billion connections 2020, 2010.

[80] Mathias Weske. Business process management architectures. In *Business Process Management*, pages 333–371. Springer, 2012.

[81] Qiang Wu, Burges, et al. Adapting boosting for information retrieval measures. *Information Retrieval*, 13(3):254–270, 2010.

[82] Qihui Wu, Guoru Ding, Yuhua Xu, Shuo Feng, Zhiyong Du, Jinlong Wang, and Keping Long. Cognitive internet of things: a new paradigm beyond connection. *IEEE Internet of Things Journal*, 1(2):129–143, 2014.

[83] Hua Xiao, Bipin Upadhyaya, Foutse Khomh, Ying Zou, Joanna Ng, and Alex Lau. An automatic approach for extracting process knowledge from the web. In *IEEE ICWS*, pages 315–322, 2011.

[84] Jun Xu and Hang Li. Adarank: a boosting algorithm for information retrieval. In *Proceedings of the 30th annual international conference on Research and development in information retrieval*, pages 391–398.

[85] Stephen JH Yang, Jia Zhang, and Irene YL Chen. A jess-enabled context elicitation system for providing context-aware web services. *Expert Systems with Applications*, 34(4):2254–2266, 2008.

[86] Robert K Yin. *Case study research: Design and methods*. Sage publications, 2013.

[87] Yu Zhao, Shaohua Wang, Ying Zou, Joanna Ng, and Tinny Ng. Mining user intents to compose services for end-users. In *IEEE ICWS*, pages 348–355, 2016.

[88] Zhaohui Zheng, Hongyuan Zha, Tong Zhang, Olivier Chapelle, Keke Chen, and Gordon Sun. A general boosting method and its application to learning ranking functions for web search. In *Advances in neural information processing systems*, pages 1697–1704, 2008.

[89] Michele Zorzi, Alexander Gluhak, Sebastian Lange, and Alessandro Bassi. From today's intranet of things to a future internet of things: a wireless-and mobility-related view. *IEEE Wireless Communications*, 17(6), 2010.