AN INTERACTIVE AND CONTEXT-DRIVEN APPROACH TO MOBILE DECISION SUPPORT SERVICES
AN INTERACTIVE AND CONTEXT-DRIVEN APPROACH TO MOBILE DECISION SUPPORT SERVICES

By
AHAD YARAZAVI, B.Sc.

A Thesis
Submitted to the School of Graduate Studies
in Partial Fulfilment of the Requirements
for the Degree
M.A.Sc.

University of Ontario Institute of Technology
Copyright © by Ahad Yarazavi, 2013
M.A.Sc. University of Ontario Institute of Technology, Oshawa, Ontario

TITLE: An Interactive and Context-driven Approach to Mobile Decision Support Services

AUTHOR: AHAD YARAZAVI, B.Sc.

SUPERVISOR: Dr. Kamran Sartipi

NUMBER OF PAGES: ix, 81
Abstract

This thesis introduces a new approach to service sophistication where the users with no prior knowledge about a public domain’s list of services can conveniently and effectively use those services in companion with complementary utility services. Such a decision support service utilizes techniques from semantic analysis that are orchestrated through a new concept namely "Smart Decision Service" that coaches the user, who is not familiar with an organization, to select the desired organization’s business services and seamlessly connect them with the proper third-party applications (e.g., map, search engine, calendar, email, voice, video) in the user’s mobile device (smart phone or tablet). Such smart decision services can be provided for a variety of strategic business domains such as: banking, insurance, government, healthcare, and on-line shopping. A prototype of the application has been developed using Xcode IDE which runs on Apple iPhone.

In the proposed approach the user installs a new type of agent in his/her mobile device and requests to be advised for services that a particular organization (e.g., City-bank) provides. The cloud provider sends the City-bank smart service to serve the user, which collects the context of the user and interacts with the cloud provider to select a specific business service (e.g., stock invest) for the customer. Also the agent in the local cellphone uses the tables of maximal associations of previous customers which share the same set of conditions to recommend the current user with the services that probably meet his/her circumstances.
Acknowledgment

I wish to acknowledge those who I feel have greatly aided me in completing this thesis.

I would like to thank Professor Kamran Sartipi for providing an open study and discussion environment, having faith in me throughout the project, constant encouragement and willingness to provide advice, and very clear guidance towards the success of this thesis project.

Furthermore I would like to express my deep gratitude to the committee members Dr. Makrehchi and Dr. Heydari for the precious time and effort they have spent on this thesis.

Special thanks to my classmates, without whose help I would not have been able to get through this difficult and emotional time. Thanks to all friends who ever helped me over the past two years.

Special thanks go to my family especially my mom and dad, for their constant supports.
# Contents

1 Introduction 1

1.1 Problem Description ........................................... 3
1.2 Proposed Solution ............................................. 3
1.3 Proposed Framework ............................................ 4
1.4 Extended Framework ............................................ 7
1.5 Thesis Contribution ............................................. 9
1.6 Thesis Overview ................................................ 10

2 Related Works 11

2.1 Semantic Matching .............................................. 11
2.2 Concept Lattice Analysis ....................................... 12
2.3 Decision Support Systems ...................................... 13
2.4 Cloud Computing ............................................... 14

3 Technologies 16

3.1 Xcode and iPhone .............................................. 16
3.2 PHP ............................................................ 17
3.3 SQL database ................................................... 18
3.4 Python .......................................................... 19
A Taxonomy

B Sample Code Snippets

B.1 Objective-C
B.2 PHP
B.3 Python
List of Tables

4.1 Parameters of Algorithm 2 ........................................ 31
4.2 Functions of Algorithm 2 .......................................... 31

5.1 Amazon ec2 prices based on instance size. .................. 47
5.2 Comparison of several famous IaaS providers. ............ 48
List of Figures

1.1 The client-server framework of the proposed Decision Support System. . 5
1.2 The extended architecture with the use of service representative on the client side of the application. . . . . . . . . . . . . . . . . . . . . . . . 8

3.1 An example of a lattice with 10 objects and 10 attributes where the indicated concept shows the association of objects 10, 4, 1 and 8 which share the attribute 10. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 22

4.1 Architecture for the proposed mobile decision service. . . . . . . . . . . 26

5.1 The complete process of using the application. . . . . . . . . . . . . . . 36
5.2 An example of a decision tree for TD bank where each node contains a unique context and the next question or the selected set of services. . . . 37
5.3 Categories which application may support. . . . . . . . . . . . . . . . . . 38
5.4 Shows the start of the questionnaire with TD bank as the selected organization. 39
5.5 User’s status during the questionnaire. . . . . . . . . . . . . . . . . . . . 40
5.6 Application retrieves services after filling out the user’s context. . . . . 41
5.7 Application redirects to TD website to apply for the service. . . . . . . 42
5.8 The lattice for the table of customers-attributes with 35 customers and 25 attributes. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 43
5.9 The lattice for the table of customers-attributes after selecting the targeted concept. ........................................... 44
5.10 The lattice for the table of recommendations-customers. ............... 44
5.11 Recommendations based on the user’s context. .......................... 46
Chapter 1

Introduction

Most large organizations world-wide are equipped with a variety of services for their clients, which are considered as effective means to offer their services to the current clients and to attract new clients. While extensive efforts and budgets have been invested to develop long lists of sophisticated business services, the effective and user-friendly offering of these services to the clients are still big challenges for the businesses, which require them to expand their marketing departments and hire information desk employees.

On the other hand, customers are not familiar with these services and their features. Due to these drawbacks, customers find it extremely difficult to choose a relevant service based on their needs. For instance, a customer intends to invest his money in a bank but he does not know what services that specific bank provides; or he may not be familiar with the circumstances or benefits of different services to decide on their appropriateness. In reality, customers either call or visit the information desks or the human consultants to get such strategic information. Also, to make an optimal decision they need to compare the pros and cons of similar services in competing organizations.

In the proposed approach the user installs a new type of customizable agent in his/her mobile device and requests to be advised for services that a particular organization (e.g.,
City-bank) provides. The cloud provider sends the City-bank decision support service to serve the user, which collects the context of the user locally to select a specific business service (e.g., stock investment). The cloud provider then sends specific instructions to the customizable agent to customize it at the mobile device to redirect the user to the web page where the user can apply for the chosen service.

Each service interface description in the cloud will be annotated with an attribute-tuple, namely "service-context" such as: <domain, role, operation, data, expertise, security, cost>. Each attribute is associated with a list of known attribute-values as well as a number of semantically related synonyms based on expert’s knowledge to characterize that attribute-value. The relationships between attributes, attribute-values, and synonyms are managed within the WordNet framework.

As a case study, we modelled the system in a Banking environment. The mobile decision support service performs a series of interactions with the user to identify a proper attribute-value for each individual attribute of the service-context. The proposed mobile decision support service performs the following tasks:

1. Prompts the text of a general question to the user, such as "Which bank do you need service from?".

2. Collects the user’s input, parses and tokenizes the input and performs a semantic search against the stored attribute-value synonyms within the WordNet database. The resulting standard synonyms of attribute-values trigger the next question for the user.

3. Repeats step 2 until the attribute-values for all attributes in the service-context are identified, and accordingly retrieves the identified service for the customer.

4. Uses map view to provide user with the closest locations of the organization (e.g.,
a branch of City-bank).

The above process guarantees that, given the available knowledge stored in the system and the context of the customer, the identified service is the most appropriate service for the customer.

1.1 Problem Description

Service selection has been crucial for large companies as they usually develop many services and several of these services have complicated features and descriptions, which make the selection process difficult for customers who should decide which one best matches with his/her needs. Moreover, organizations always attempt to assist their customers by providing brochures and personal consultations. However, the customer is required to understand the detailed requirements of the service before making an educated decision. This involves extensive discussions with the company’s consultant which in-turn consumes valuable resources in terms of time, while still the customer may not fully understand the service and finally make a wrong decision. Based on the above discussions we define the problem in this thesis as:

provisioning interactive and mobile decision support services in large organizations that guide the customers to identify their desired services through guided questions that are derived from the customer’s personal context.

1.2 Proposed Solution

In this thesis, we propose a mobile device application (as mobile decision support service) which assists customers in service selection process within an organization. The requirements of such a solution are as follows:
Chapter 1. Introduction

- Decision Service should manage to offer services suitable for the user based on the user’s context.
- It should administer the user in service selection with the least number of questions possible.
- It should not require the user to be familiar with the services that the organization offers.
- It should be able to assist the users in utilizing the service by providing utility service within the decision service.

The proposed solution involves techniques from mobile software development, web services, database semantic analyses and text processing. Such a framework can be implemented for a variety of business domains such as: Banking, Insurance, Government and Healthcare.

1.3 Proposed Framework

Figure 1.1 illustrates a framework for the proposed solution in the form of a process flow graph. The decision support service in the mobile device uses a set of carefully designed questions to identify the user’s circumstances and consequently provide the best matched service. Different modules in Figure 1.1 are defined below.

- User Interaction. We used Apple iPhone to provide a simple and interactive user interface which gathers data from the user and displays the results and recommendations to the user.

- Match Finding. User interface sends the user’s answer to this module to decide whether the answer is comprehensible by the system or not. In the proposed
framework, for each attribute in the user-context, a set of known attribute-values (words) are defined and the decision service will select the proper attribute-value to be placed in the user-context. For example, if there is an attribute with the name "occupation" in the user-context, the decision service fills this attribute with the words such as: employee, unemployed, self-employed and student. Hence, based on user’s answer and these known words, this module indicates whether the answer should be sent directly to “Context Updating” module or it should be directed to “Text Processing” module which helps the decision service to understand the user’s unknown answer. We have used the interface for the user to enter his answer rather than providing preselected answers since the number of known answers may vary based on the question and displaying all the possible answers may not be possible.
on smart devices with small displays.

- **Context Updating.** Each user in the proposed approach is associated with an attribute-tuple namely user-context. For every business domain this attribute-tuple should be designed differently based on the information and the prerequisites that each domain needs to identify a service. For example in banking domain the user context may include: *city of residency, age, annual income, gender, credit* and *history*. This module is responsible to maintain the user context and instantiate its attributes with the proper attribute-value that is given to it, and send the updated user-context to the “Database Searching” module to decide on the next operation of the decision service.

- **Text Processing.** Natural Language Toolkit (NLTK) and WordNet are used to process the unknown answer (received from the Match Finding module) which assists the decision service in understanding the user’s answer to a question. This module receives the unknown user answer and compares it with the known answers associated with the targeted attribute in the user’s context to find the best match. If a match is found, this module sends it to the “Context Updating” module; otherwise, the “User Interaction” module is informed to notify the user to change his answer. Since the algorithm to understand an answer is highly complex and needs powerful machines, the Text Processing module is placed in the server-side of the decision service.

- **Database Searching.** Decision trees containing detailed information about the services should be designed to be used by the decision service. Each “service” or “question” in the database is associated with a unique context. This is the main part of the server which provides the decision service with the next move at each step. After receiving the user’s context, this module returns a node of the decision
tree, which contains: i) the next question to ask from user to identify the value of the next attribute in the attribute-tuple; or ii) a service name and other information that are required to execute the identified service.

- **Service Execution.** After asking several questions, the attribute-tuple (context) would be specific enough for Database Searching module to retrieve a service or multiple services. This module provides means to display services with their information for the user to choose from. Then it redirects the user to the web page where they can apply for that specific service, and also provides them with the map view to the closest branch of that organization. The Decision Service will then enter to a “Standby” mode to serve the customer at the next attempt.

- **Promotions Updating.** After each update of the user-context, the Promotion Updating module provides the user with the list of relevant services and promotions based on the services that the current and past customers have been using in the same organization. To provide such information, we use algorithms from data mining and concept lattice analysis in databases of customers to explore the associations between group of services they have used and prompt those that are appropriate to the current customer.

- **Standby.** After executing the service for the user, the decision service will enter the standby mode where all the user information, which we gathered through the process, will be deleted and it will wait for the next use.

### 1.4 Extended Framework

In this thesis, we propose to use client-side service representative to perform business services locally which preserves private client’s data, i.e., the client application does not
send its private data through communication channels to the service provider. Figure 1.2 indicates the processing of the service locally. We assume that the service selection has been successful and the user has selected its desired service to be performed.

Each task service interface description is represented by a task message in the provider’s server. Task message is defined as a triple \(<Model, Knowledge, Data>\) which is required to perform the service. Task Model specifies what the agent is supposed to do. Task Knowledge lets the agent know how to do each step of the task model. Task Data are the task resources.

In Figure 1.2 there is a repository of task messages in the organization’s private servers. When the client application requests a service, the service provider returns a task message designed for that specific service. At this point, the service representative which is deployed at the client side performs the requested service based on the task message and the client’s local information.
Chapter 1. Introduction

This architecture is suitable where the service provider is providing a service either or not based on client’s data but the organization is not required to store the private information of client and the service can be performed on the client’s local representative based on the instructions received from the service provider.

1.5 Thesis Contribution

This thesis presents an interactive and context-driven approach for mobile decision support services which provides assisted knowledge in service selection for customers in different organizations. Contributions of this thesis to the field of mobile web services are categorized as follows:

- Proposing a new type of context-aware decision support service that acts as a coach for the user to identify the desired services; without prior user knowledge about the organization’s services.
- Applying semantic analysis using WordNet to perform natural language processing.
- Applying a data mining technique using concept lattice analysis to provide additional guide for the user to select services.
- Proposing a client side service processing technique which preserves client data privacy.
- Providing a prototype mobile decision support service running on Apple iPhone.

As a result of this study, we implemented the proposed decision support approach using Xcode IDE toolkit and the following techniques: i) PHP, server side scripting language; ii) MySQL, relational database management system; iii) WordNet, text processing and NLTK (Natural language Toolkit); and iv) Python, to use text processing toolkit.
1.6 Thesis Overview

The remaining chapters of this thesis are structured as follows:

Chapter 2: Related Work. Presents an overview of the related approaches to the technique proposed in this thesis.

Chapter 3: Technologies. Describes the technologies we have used for experimenting our proposed architecture.

Chapter 4: Approach: Mobile Decision Services. Presents the proposed approach which we used toward designing our architecture.

Chapter 5: Experimentation. Illustrates the case study we used to build the decision service using iOS as the smart phone environment.

Chapter 6: Restrictions of the Approach. Describes the challenges we encountered during the experimentation.

Chapter 7: Discussion and Conclusion. Concludes the discussions, and proposes future works to improve the approach of this thesis.
Chapter 2

Related Works

In this chapter, we concisely discuss related works and several approaches that we have used in our project. In Section 2.1 we present the new approaches for semantic matching which we have used in text processing part of the application. Section 2.2 presents the history of the concept lattice analysis and also the applications of which. Section 2.3 discusses Decision Support Systems and its variety and finally section 2.4 discusses cloud computing.

2.1 Semantic Matching

Match is defined as an important operator that is used in many areas of software engineering such as data mining, text processing and data warehouses and integration. Many approaches has been introduced to solve matching problems, surveys of which can be found in [24, 10].

In[13], Guinchiglia and Shvaiko describe an algorithm, which takes two graph-like structures and produces mappings between those nodes of graphs, which relate semantically to each other. Semantic matching as defined in [12], is to calculate mapping by
computing the semantic relations holding between the concepts of the nodes. Concept of a node is defined to understand the definition we mean by the label of the node. As a case study they have used graphs explaining different documents. Possible semantic relations which has been used in [13, 11] are: equivalence, more general, less general, mismatch and overlapping.

Semantic matching has been introduced in [13] by analyzing its usability on two tree like structures where each node in both trees are attached with a label. The algorithm explained by the authors, first starts by computing concept for each nodes in both trees. They have used WordNet database for this matter which is introduced in [4]. After computing concepts, they tried to match between each concepts from both trees to find the semantic relations between nodes. Likewise in this thesis, semantic matching has been used in text processing to find the similarity between user’s response and the known terms by the application.

2.2 Concept Lattice Analysis

Concept lattice founded by G. Birkhoff [5] in 1940. But more widely, works in application-wise of concept lattice in the area of reverse engineering instructed in 1993. Concept lattice analysis provides a way to identify groupings of objects that have common attributes.

Modularization of legacy codes is using concept lattice analyzing widely [18, 29, 30], where lattice is used to identify the relation between program functions and their attribute values. Besides, more recently concept lattice analysis has been used in implementation of certain features of the software system [9].

From the early discovery of concept lattice analysis, it has been used in several works of data mining [1, 31], where it is an essential tool for gaining knowledges from large databases. In [1] Anis Yousefi and others, built a database with diseases as objects and
symptoms as attributes. Then they have used concept lattice analysis on this table to get
the frequent item sets (concepts) to use in diagnosing the patient because using concepts
would make the process of searching much faster and more understandable. Similarly we
used concept lattice analysis to find best promotions/services which matches the best with
user circumstances. It reduces the searching time and complexity of search algorithms
since we build the concept database offline and then we use it in the application.

2.3 Decision Support Systems

From the mid 1970s, Decision Support Systems became an interesting and widely used
area of research. As defined in [19] and [16] Decision Support System (DSS) is an interactive
computer-based system that helps users in judgment and choice activities. Initially,
DSS consists of knowledge base, inference engine and user support. Knowledge base is
an intelligent database to maintain and retrieve knowledge from related domains to use
in inference engine [16]. The inference engine is the part of expert system which makes
logical decisions based on the knowledge about a specific situation. DSSs are defined
in various domains such as health care (Clinical DSS), Organizational decision support
system (ODSS), Group decision support system (GDSS) and etc.

Decision Support Systems are a major point of interest in several field of research
streams. Task interaction, architectures including Web and client-server approaches,
effectiveness of computer graphics for decision support and linking models and database
technologies are examples of which areas. User-computer interaction and decision making
based on user response are the topics which we mainly focus on this thesis. In [28] a
decision making model is presented to support several aspects in business rules lifecycle
and they described a method for extracting business rules from decision system.

Currently, Recommender services based on DSS theory are widely used in companies
web sites to suggest proper services based on individual interests and their navigational behaviour. Such ideas have been emerged since mid 90’s where user’s conditions and opinions have been used in order to identify products or services which most likely are interesting for the customer such as Amazon and eBay web sites. In [33], Quan Wen and Jianmin He introduce a recommender service with and service oriented architecture which uses data mining algorithms to analyze customers shopping history. Their approach to recommend products is content based that is to recommend based on associations among products.

In this thesis, we have used DSS theory and data mining algorithms to design a recommender application which suggest proper services for a user with no prior knowledge from the organization based on user’s conditions and limitations.

### 2.4 Cloud Computing

Cloud Computing is an expression which describes a powerful computing power with the use of a large number of computers connected through internet as the communication channel. This creates the ability to run a program at the same time on several connected computers without any change in usage speed.

A survey of recent technologies in Cloud Computing is provided in [27] where the authors present different kinds of services that Cloud providers offer to public, such as: Infrastructure as a Service (IaaS) and Hardware as a Service (HaaS). Besides, healthcare and specifically mobile healthcare use Cloud as a service provider for interconnection between patients and hospitals [21, 35, 8]. In [17], Li and Svard describe a web service which is used for communications between customers and employers of a text correction company. They have used Cloud as their provider and they refer to their web services as a new kind of Cloud services namely Human as a Service (HuaaS). In this thesis we
investigated to migrate our server side of the project to Amazon EC2 (which is an IaaS provider) as our application server provider since the time-efficiency is important for a mobile application where the number of users may increase rapidly.
Chapter 3

Technologies

In this chapter we explain the technologies we have used in the proposed architecture. In Section 3.1 we explain Xcode as the platform for writing the client-side of the application in the Apple iPhone platform. Then in Section 3.2 and Section 3.3 we introduce PHP as the server-side scripting language and MySQL as the relational database management system that is used throughout the server-side programming of the application. Furthermore, in Section 3.4 and Section 3.5 we explain Python programming language and WordNet database which are used in our text processing operation. Also concept lattice analysis technique is explained in Section 3.6. Finally Cloud technology is discussed in Section 3.7.

3.1 Xcode and iPhone

Xcode is the main integrated development environment (IDE) created by Apple Cooperation which is used to implement applications for Mac OS X and iOS operating systems. XCode is the Apple’s integrated development environment (IDE) which was first released in 2003 and the latest version is Xcode 4.6, and it is free for Mac users. However, for
uploading the application to the Apple application store and for installing the application on an Apple mobile device (iPhone, iPod and iPad) there is an annual charge for developers.

Xcode uses Objective C programming language which is a derivation of the C programming language. It was introduced in the 1980s as a programming language for developing the NeXTSTEP operating system from which the OS X\textsuperscript{1} and iOS\textsuperscript{2} are driven. Unlike C++, Objective C is a single inheritance language and uses interfaces to fake multiple inheritance (like Java) which makes it less complicated to program with.

The prototype application developed in this thesis was originally designed for smart phones. We had to choose one of the two popular smart phone operating systems, i.e., iOS for Apple iPhone or Android supported smart phones. We decided to adopt iPhone and Xcode IDE due to the following advantages:

- iOS has been around for longer time compared with Android, therefore it is more stable with less bugs.

- More resources are available for developing an iPhone application.

- Simulator for iPhone is faster than the simulator for Android. During debugging the code, each run of the application on Android takes more than 30 seconds whereas the on iOS takes less than 5 seconds.

### 3.2 PHP

PHP (Hypertext Pre-processor) is a server-side scripting language introduced by Rasmus Lerdorf in 1995. PHP is free of charge and is installed on more than 2 million web servers\textsuperscript{3}.

\textsuperscript{1}Apple mac operating system

\textsuperscript{2}Apple iPhone operating system

\textsuperscript{3}
PHP code can be embedded into HTML code and works with many relational database management systems such as MySQL. Also, PHP version 3 and later releases support object oriented programming. PHP is designed to work as an interpreter which takes PHP commands and converts them to HTML code which is compatible with almost every web server.

We adopted the PHP programming language since it had the desired features required for our project: i) PHP is used for server-side programming; ii) it supports MySQL database; and iii) it is able to run Python scripts since the text processing part of the server is written in Python.

### 3.3 SQL database

SQL (Structured Query Language) is a programming language which is designed for handling data in a Relational Database Management System (RDBMS). SQL was developed by IBM organization in 1970 and became an standard of ISO (International Standards Organization) in 1987. Almost all the RDBMSs use SQL for the database management language like MySQL.

MySQL is the most widely used relational database management system. Several applications use MySQL database, including: Joomla, phpBB, MyBB and XAMPP. MySQL also is used in popular websites such as: Wikipedia, Facebook and Google. MySQL also works on several cloud computing platforms such as Amazon EC2\(^1\).

In this thesis, we use XAMPP which is a web server platform providing Apache HTTP server, MySQL database and also interpreters for PHP and Perl scripts. XAMPP is an open source and free of charge server side management application that works on different

---

\(^1\)http://www.php.net/usage.php
platforms as Windows and Mac OS X.

3.4 Python

Python is a high level programming language whose syntax allows programmers to develop applications with fewer lines of code. Python is a scripting dynamic programming language with automatic memory management system which provides support for object-oriented programming. Python interpreters are available for many platforms such as Mac OS X, Windows and Linux and in several cases these interpreters are open source and free of charge like CPython.

Python is widely used in text processing and machine learning programmings because of the packages that are available to use in this language. Natural Language Toolkit (nltk\(^1\)) is supported by Python which allows the developers to work with plain English and provides interfaces for several lexical databases such as WordNet.

For the text processing part of the application we used Python and NLTK packages. We have used NLTK to process and tockenize the user’s input and understand the concept using WordNet in Python.

3.5 WordNet

WordNet [4] is the largest lexical database of English words created by George A. Miller in mid-80’s at The Cognitive Science Laboratory of Princeton University. Since then, WordNet has been growing with fundings from government because of the interesting field of machine translation.

\(^{1}\)EC2 is the Amazon Organization cloud computing Platform
\(^{1}\)http://nltk.org
WordNet consists of nouns, verbs, adverbs and adjectives organized into groups of synonyms (namely “Synsets”) each stating a distinct concept. The most frequent relation among words are the “Heyponyms” and “Hypernyms” (namely ISA relation). For example, the word ”waiter” is a subordinate (Hypernyms) of the word ”worker” and the word ”entity” is the superordinate (Heyponym) of the word ”worker”. WordNet contains several graphs of synsets based on ISA relation among them [14, 7].

WordNet has been used in various fields of machine translation and learning, semantic analysis and text processing as a powerful tool for finding relations between words [15, 6, 23, 22]. As it has been explained in [26], three kinds of similarity measurements can be reached through WordNet:

- **lch-similarity:** finds the similarity based on the shortest path between two concepts and then it scales the value by the maximum path length in the same graph.

- **wup-similarity:** finds the similarity based on the depth of the concepts.

- **path-similarity:** it is the path similarity percentage of the two concepts.

WUP similarity (Wu and Palmer [34]) first, finds the depth of the LCS\(^1\) and each concepts alone and then, with the use of following equation it generates the similarity score:

\[
\text{score} = 2 \times \frac{\text{depth}(\text{lcs})}{\text{depth}(\text{s1}) + \text{depth}(\text{s2})}
\]

The score value is always between zero and one. For text processing part of this thesis, where we try to understand the user response, we use wup-similarity as the main function.

\(^1\)LCS is the least common subsumer (shared parent) of two concepts
3.6 Concept lattice

Concept Lattice theory is a way to find Maximal associations from a binary related collection of objects and their attributes. It means that each object possesses several attributes shown by a binary relation and presented by a two-dimension table, where rows represent objects and columns represent attributes. Next, from the table we can generate a lattice (an example is shown in figure 3.1) with the following properties:

- Except for the first and the last node, all nodes in a lattice are labeled with objects and attributes.
- In a lattice each object in a node inherits all the attributes above it.
- Every attribute endures in all the objects under it.

Each node in concept lattice is a concept which contains a set of objects and a set of attributes. The set of objects in a concept is called extent and the set of attributes is called intent. [3, 1]

Figure 3.1 illustrates the lattice of a table consisting of 10 objects and 10 attributes. The node shown in the figure with the red pointer indicates a concept with obj1, obj4, obj7 and obj8 as the extent and the attr10 as the intent. We used The Concept Explorer toolkit [2] to generate different tables and corresponding lattices in this thesis.

3.7 Cloud technology

The notion of ”intergalactic computer network” which is recently known as cloud computing was first introduced in 1962 by J.C.R. Licklider (Developer of ARPANET\(^1\)). The concept he initiated was for everyone, anywhere around the globe to be able to access
Figure 3.1: An example of a lattice with 10 objects and 10 attributes where the indicated concept shows the association of objects 10, 4, 1 and 8 which share the attribute 10.

data, programs and processing power throughout the network [25]. These notions contained everything we use as internet today including cloud computing. Although the idea of cloud computing introduced in 1962, cloud computing developed from being just an idea into a technology since 1990s when the Internet bandwidth expanded massively.

The term Cloud Computing has been used widely because most large companies such as Google and Amazon provide this service for public use at low cost and high efficiency. Google runs approximately 500,000 servers which are clustered into more than ten physical locations and created a new type of centralized computing power by creating a network that is spread wide and thin rather than narrow and deep. Cloud computing is fast and robust in terms of computational architecture. The cloud network can recover from regular servers weaknesses, such as hardware failures [32].

In [32] Weiss describes that several operating instances can run simultaneously using virtualization which is one of the advantageous of cloud. Cloud computing can break tasks into their smallest threads, and each thread could be completed simultaneously.

\footnote{Advanced Research Projects Agency Network, 1969}
with the use of difference processors. Specifically, for client-server applications, it is best to deploy the server side of the application in the Cloud since the response time would not slow down when the number of requests increases. A practical approach to move a business server to Cloud should cover the costs-benefits and migration to cloud computing. However similar to other Internet-based technologies Cloud Computing has its own limitations, e.g., security issues.

As it is described in [27] Cloud Computing mainly falls into three categories:

- **IaaS** (Infrastructure as a Service). IaaS provides virtual machines (virtual infrastructure) for the users, using a set of physical computers, and the users deploy their applications on the virtual machines instead of the physical machines. Examples of IaaS providers are Google Compute Engine, Amazon EC2, Rackspace and Joyenet.

- **PaaS** (Platform as a Service). PaaS provides a computing platform for the users including the operating system, database and web server. Example of popular PaaS providers are: Google App Engine, AppScale and Cloud Foundry.

- **SaaS** (Software as a Service). SaaS providers a range of software and databases, and sometimes is is referred to as “on-demand software”. Examples of such services are: Microsoft Office 365, GT Nexus and Google Apps.

### 3.8 Client-Side Service Computing

According to [20], “Data Service” refers to a web service where a client application (for simplicity we refer to “client application” as “client”) asks for a service on its own data from a service provider, and the client should send its local data to the server. The service provider processes the client data based on the requested service and sends the result back to the client. However, in a “Task Service” [20] the client doesn’t need to
send its local (and private) data to the server side; the client data is processed at the client side and hence the data privacy is preserved. The steps for task service processing are as follows: i) client asks for a service by sending a request message to the service provider; ii) service provider performs the required server side processing and defines a “task message” to be performed by client at the client side; iii) service provider sends this task message to the client; iv) client uses a local agent, namely “service representative” to perform the task for the client (see below for more details); v) service representative receives client data (locally), performs the task, and returns the result (locally) to the client.

A “generic” service representative is a piece of program that is written to act as an agent on-behalf of the service provider to serve the client. This generic agent should be able to receive commands from the provider to be customize for a specific task and then process the client data locally. After performing the task, the service representative sends the result back to the client. Thus, from client’s view, the server provided a service, but the difference here is that the client’s data has been processed locally to preserve confidentiality of client data. In [20] the task message is defined as a triple: task model, task knowledge and task data, as follows:

- **Task model**: is a set of steps to do the task based on the client request.

- **Task knowledge**: is a set of conditions/actions to do each steps in the task model.

- **Task data**: is the data needed for performing the task message within the service representative that parts into client-side data needed from the client and server-side data which as a part of task message received from the server.
Chapter 4

Approach: Mobile Decision Services

In this thesis, we propose an approach to mobile decision support service to gather the user’s attributes locally and communicate with the provider to propose the best services to the customer. Such a decision service application provides appropriate means for the user to apply for the desired services within the application.

A mobile decision service provider, namely “mService-Firm” provides registration facilities for the organizations that will participate in the proposed service. To offer this service, the mService-Firm provides the following facilities for two groups of clients: i) Registered organizations. It provides web-enabled applications for registering, billing, and customer service purposes; and ii) Mobile users. It provides an application to be installed on their mobile devices (smart phones, tablets) which allows the users to select their “target organization” among the registered organizations, and to invoke the decision service of the target organization to interact with the users. Each registered organization has its own cloud-based server that provides: i) a specific Decision Support Service that is installed on the user’s mobile device, and ii) a set of Decision Support Trees to be used for identifying a service of the organization that is closest to the user’s desired task.

The whole process is divided into four steps as follows.
Chapter 4. Approach: Mobile Decision Services

Figure 4.1: Architecture for the proposed mobile decision service.

- **Step 1 (Target Organization).** The user asks the installed application on the mobile device (from mService-Firm) for a specific target organization from the list of registered organizations, e.g., *Central Hospital* or *TD Bank*.

- **Step 2 (Decision Service).** The application invokes the specific decision service of the target organization. The decision service starts interacting with the user by asking questions to complete the attribute-tuples of the user context. At each iteration, updated user’s context will be sent to the organization’s server asking for the next question available in decision trees of that organization. At some point the user context will be detailed enough to allow the decision service to identify
services which are similar enough to the user’s desired task. The decision service will then prompt with a list of service names along with short descriptions about each service (service info). The user will then select one or more services from the list to be performed.

- **Step 3** (*Desired Service*). The provider sends back a message consisting of the required information to administer the decision service to provide the needed interface for the customer to use the service locally, or redirect the customer to a web page (or another application) where he can simply apply or use the business service.

- **Step 4** (*Utility applications*). The decision service can provide additional utility applications that are required to effectively perform the service for the user. For example, a map is needed for showing the nearest hospital to the client.

Figure 4.1 illustrates the architecture of the proposed decision support service. In the following subsections, the main four units of the architecture are discussed. These units are: *User Interface Unit, Data Unit, Processing Unit* and *Action Unit*.

### 4.1 User Interface

User Interface (UI) unit interacts with the user to obtain the input data and also displays the result of the Action unit to the user. At the end, the UI will display the service (or

---

1The following (Step 3) uses “Task Services” to implement the “Desired Service”, and is considered as the future work for this research.

**Step 3 (Desired service).** The decision service asks the organization’s server (service provider) to perform the selected service (which is a “task service”) for the user. Each task service sends a “task message” (a triple `<Task model, Task knowledge, Task data>`) to customize a client-side generic agent (a service representative installed on the user’s mobile device) to work closely with the user. Therefore, the service representative after customization by the task message will process the organization’s business service for the user locally on the mobile device. Hence, the user’s confidential information will remain in the mobile device to preserve data privacy.
services) that match with the customer’s needs. A pseudo-code implementing the high
level behaviour of the system is provided in algorithm 1.

Algorithm 1 : User Interface Procedure

1: while (True) do
2:     action = Receive input data;
3:     if (action implys a question) then
4:         Extract and display the question;
5:         Customize the UI with respect to the type of question;
6:         Wait for user to respond;
7:         Send the answer to Data Unit;
8:     else if (action implys a service) then
9:         Display the service link.
10:        Provide information for the user to locate the service;
11:        Update and display the new advice, promotion, or service;
12:     else // If action contains error message
13:        Display message: “Answer is not Understandable”;
14:     end if
15: end while

As it is shown in Algorithm 1, this unit receives data from the Action unit and sends
the user’s response to the Data unit. The received data can be either the next question or
the services which are matched with the user’s needs. Otherwise the action unit prompts
a message “Answer is not understandable” meaning that the system did not comprehend
the user’s response to the question and consequently the decision support service will ask
the user to modify his/her answer.

Attached with the question, Action unit sends the type of question and a few known
inputs to the user interface. This information is needed in this stage to customize the
UI regarding the type of the question we are asking from the user. For example, if the
question is "How old are you?", the system expects the user to enter a number. So the
application customizes the UI to show a decimal keypad for the user, or if the question
requires inputs which is not specific, the UI would show a few known inputs which user
can pick from or he can write his own answer in the provided text field. Another reason
for displaying a few known inputs is to give the user some ideas about what the question
means exactly.

4.2 Data Unit

Data unit collects data from the user interface based on the question that has been
asked. We have two types of user inputs, which are known inputs and unknown inputs.
For example, if the element we are trying to fill in the context-tuple is occupation, the
question we ask from the user can be: "what is your occupation?" and we have to define
the occupations we want our system to understand. In this case the known inputs that
the application recognizes would be: employee, employer, self-employed, student and
unemployed. These known inputs are attached to the question that we receive from the
server. If the user answers something else like "waitress", it would be tagged as an
unknown input.

If the user’s answer has been tagged as known it would be sent directly to the pro-
cessing unit where the context of the user will be updated with the exact same input.
Otherwise, this unit will package the user’s answer with the known inputs and sends
them to the python script at the server side for text processing which tries to map the
user’s unknown input to one of the known elements.
4.3 Processing Unit

The Processing unit processes the input data and fills the context accordingly. For the known input data the process will be easy as we know exactly what the user meant. We can simply fill the specific part in the user context with the same input. But the process will get more complicated when the input is unknown for the system and we do not have a definition for it. In this case we use WordNet database to map the input into a known element for the system. The pseudo-code for the above operation is shown in Algorithm 2.

Each question is typically targeted for a specific attribute in the user’s context. When receiving a question, the attribute name and the answers which are known for the targeted attribute are attached to the question so the application would know what to expect from the user’s response. Name of the attribute is needed by the application to apprehend which element in the user’s context is targeted by the question.

The following information will be sent to the server to be processed: unknown user response, and known inputs for the targeted attribute. In the server, using NLTK (Natural Language Toolkit) we first tokenize the input to extract the important part of the unknown input (it can be a “word” or a “sentence”) like nouns and verbs. Then, the WordNet database and WUP (Wu and Palmer [34]) similarity theory are applied to identify the best match between the extracted part of unknown input and the known inputs. The best similarity found will be sent back to the application. Otherwise, the server notifies the application with an error message meaning that an acceptable similarity percentage was not found.
Chapter 4. Approach: Mobile Decision Services

Parameters of Algorithm 2

<table>
<thead>
<tr>
<th>Input Parameter</th>
<th>Definition</th>
<th>Local Variable</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>ans</td>
<td>User’s answer (String)</td>
<td>synArrayAns,</td>
<td>Arrays of synonyms</td>
</tr>
<tr>
<td>$U_i$</td>
<td>The set of known inputs for the $i$th element of the context</td>
<td>synArrayKnown</td>
<td>Arrays of synonyms</td>
</tr>
<tr>
<td>$rel$</td>
<td>Semantic relation$(=,\sqsupseteq,\sqsubseteq,\perp,\sqcap)$</td>
<td>bestMatch</td>
<td>The best of ’match’</td>
</tr>
</tbody>
</table>

Table 4.1: Parameters of Algorithm 2

Functions of Algorithm 2

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RelWordNet($a, b$)</td>
<td>Returns semantic relation between $a$ and $b$</td>
</tr>
<tr>
<td>wn.synsets($a$)</td>
<td>Returns all the synonym sets of the word $a$</td>
</tr>
<tr>
<td>wn.similarity($array_i, array_j$)</td>
<td>Returns the average similarity percentage between two arrays of synsets</td>
</tr>
<tr>
<td>Tokenize($a$)</td>
<td>Returns an array of all the words in string $a$</td>
</tr>
<tr>
<td>Reduce($array$)</td>
<td>Omits all the oppositions and Conjunctions of $array$</td>
</tr>
<tr>
<td>FindMatch($array_i, array_j$)</td>
<td>returns the success percentage of semantic matching of all the elements in the two arrays</td>
</tr>
</tbody>
</table>

Table 4.2: Functions of Algorithm 2
Algorithm 2: Processing Unit Mechanism

1: if \((\text{ans} \in U_i)\) then
2:   Update Context with \text{ans};
3:   goto \text{exit};
4: end if
5: if \((\text{ans} \text{ contains one word})\) then
6:   for \((r \in U_i)\) do
7:     \text{rel} := \text{RelWordNet}(r, \text{ans});
8:     if \((\text{rel} = ”\sqsubseteq” \text{ or ”} = ”)\) then
9:       Update Context with \text{r};
10:      goto \text{exit};
11:     end if
12:   end for
13: end if
14: if \((\text{ans} \text{ contains multiple words})\) then
15: \text{synArrayAns} := \text{wn.synsets(ans)};
16: end if
17: if \((\text{ans} \text{ contains multiple words})\) then
18: \text{ansArray} := \text{Tokenize(ans)};
19: \text{ansArray} := \text{Reduce(ansArray)};
20: for \((r \in \text{ansArray})\) do
21:   \text{synArrayAns} := \text{synArrayAns} + \text{wn.synsets(r)};
22: end for
23: end if
24: \text{bestMatch} := \text{null};
25: for \((r \in U_i)\) do
26: \text{synArrayKnown} := \text{wn.synsets(r)};
27: \text{match} := \text{wn.similarity(synArrayAns, synArrayKnown)};
Algorithm 2: Processing Unit Mechanism (continued)

26: if (match is better than bestMatch) then
27:    bestMatch := match;
28:    bestMatchAns := r;
29: end if
30: end for
31: if (bestMatch) then
32:    Update Context with bestMatchAns;
33: goto exit;
34: else
35:    send ContextUpdateFailure message to Action Unit;
36: end if
37: exit: Send Context to Action Unit;

4.4 Action Unit

Action unit operates based on the current context. If we could not update the context in the Processing Unit we will have to ask the question in an altered way so that the user would respond something that the system can comprehend. Otherwise, the context has been updated; hence with the new context we can search the decision-tree database for the next move. If the context is not complex enough the system will receive another question to fill another attribute in the context-tuple. Subsequently after a few times we will get a context that is informative enough to be matched with a service context or a set of services from the service database. In this way, the application will retrieve the service or services that are also annotated with short descriptions for the user to select from.

Every time the user-context is updated, the Action Unit uses the database of concepts
to find relevant services (or promotions) to our current user based on his/her context.

Almost all large organizations have logs and history databases from their current and past customers that are considered as valuable assets with rich information about the decisions made by their customers over the years. In the proposed approach, we use a data mining technique using concept lattice analysis to extract useful knowledge from hidden information in the attributes of users with similar situations in order to assist the new user to make a more knowledgeable decision.

In this approach, in an off-line operation we run concept lattice analysis on the existing and past customer’s attributes and their selected services to create a knowledgebase of “highly associated groups of attributes and the selected services”. Such a knowledgebase will allow the decision service to assist new users in their service selections.

At each step during interaction with the user for collecting his/her attributes, the decision service performs a comparison between the collected user attributes with the attributes in the highly associated groups in the knowledgebase to come up with suggestions for the user. This allows the decision service to provide a summary information about current/past customers’ situations, choices, opinions, etc. to guide the new user.

In particular, we run concept lattice analysis on the table of previous customers and their attributes consisting of 35 customers and 28 attributes which is shown in Figure 5.9. The generated concepts (highly associated groups of customers and attributes) are stored in a database. We also generated a second concept lattice based on customers and their selected services and stored the concepts (highly associated groups of customers and services) in the second database. By using these databases, we could come up with a relation between the user attributes and the selected services and used them as described above. More detailed discussion has been provided in the Chapter Experimentation.
Chapter 5

Experimentation

For implementation of the proposed system we have used an Apple MacBook Pro computer with 2.3 GHz Intel Core i5 processor and 4GB memory capacity running a Mac OS X 10.8.4 operating system. We used Xcode 4.6 IDE and Objective C programming language to write the client-side of the application running on iPhone, and used PHP for server-side programming. Also we used Python 2.7.2 for text processing part of the application.

5.1 Case Study

To demonstrate the functionality of the system we implemented a banking application based on the services provided by the TD Bank of Canada. The Whole functionality of the application is shown concisely in figure 5.1. We have investigated several services offered by the TD Bank and the circumstances to apply for each service. Based on the gained knowledge we designed the attribute-tuple for TD Bank as: <Status, Occupation, TypeOfService, Age, AmountOfMoney, UseOfMoney, CreditHistory, Degree>. This attribute-tuple is the user’s context that we try to identify its attribute values, and
Figure 5.1: The complete process of using the application.
Figure 5.2: An example of a decision tree for TD bank where each node contains a unique context and the next question or the selected set of services.

based on the evaluated context we will be able to suggest the most relevant services for the user. Each element in this tuple has an attached question along with possible answers which the system should understand (i.e., known answers). Decision trees are designed with respect to the attribute-tuple and the next proper question to ask.

Figure 5.2 shows a simple example of decision trees that we have designed based on the TD Bank services. Each tree node is annotated with a context as well as a question to ask (or a service name to suggest). The context for the root node in each tree is designed with three common elements: Status in Canada, Occupation, and Type of the desired service. In our investigation with the TD bank we discovered that these elements are the most important among others, hence the decision service would ask these questions first and then based on the received answers it will match a decision tree which would be most suitable to proceed with user context exploration.
The above described technique is applicable for different domains, including: Banking, Hospital, Insurance, Transportation, etc. Figure 5.3 shows different categories of organizations that the user can select after running the application on the mobile device. For example when the user selects banking, the application will show the list of all banks in Canada and the user can search for specific bank he intends to do business with. In this case, when the user selects TD Bank from the list, the application searches the database of decision support services in the server and retrieves the TD bank decision service to run. This decision service only holds the first few questions and their known answers for that specific organization. Also the application would receive the name of the table of the decision trees specified for that organization. After receiving these information from the server, the application starts asking questions from the user and filling out the context required for the organization. Figure 5.4 displays the decision service for TD Bank.
5.2 Processing User’s Response

When the user answers a question, the application checks whether the answer is “known” or “unknown”. If the answer is known, the corresponding attribute in the attribute-tuple will be filled with the answer. Otherwise, the user’s answer will be tagged as “unknown” and the application will send the user answer and the set of “pre-defined answers”, namely, “known answers” to the Python script for text processing. These are all possible answers for that specific attribute. At this point, the application attempts to map the user’s unknown answer to one of the pre-defined known answers, to fill the corresponding attribute. In Python script: i) unknown answer will be tokenized and spell checked; ii) tokenized array will be reduced by excluding all parts of the sentence except the “nouns”, “verbs” and “adjectives”; iii) using NLTK (Natural Language Tool Kit) and WordNet, the NLTK searches to find a semantic match between the resulting
answer-array (Tokenized, spell checked and reduced) and each known answer; and iv) identified best match (i.e., a match with more than 80% similarity) will be returned to the application as the “matched answer”. If the Python script could find the matched answer, the user’s context will be filled out and the updated context will be sent to the server to process the next step (i.e., asking the next question based on the desired decision tree with regard to the updated context). Otherwise, if the application couldn’t find a proper match with the user’s answer the user will be informed and the question will be asked again.

The main task of the Python script is to use the WordNet to calculate a matching-percentage between two concepts (or synsets). To achieve this, we obtain the wup-similarity value that is defined in the WordNet as follows. If two concepts are in the same graph in the WordNet database, the wup similarity function finds the path lengths of the two concepts to the nearest common root, and returns a value between 0 and 1 as
the wup-similarity value with the use of following equation.

\[
\text{score} = 2 \times \frac{\text{depth}(lcs)}{\text{depth}(s1) + \text{depth}(s2)}
\]

For instance, \textit{wup} returns 0.9 as a result of searching for two words "employee" and "waiter". However, if the two concepts belong to different graphs the similarity function returns -1 as they have no common root [26]. In summary the overall process is as follows. The user answers to a question, e.g., "what is your occupation?" as "I am a waiter in MacDonald’s". The user’s answer will be tokenized and trimmed such that only the nouns in the answer will remain. In this case, the words "waiter" and "MacDonald’s" will be kept. Then the similarity function checks to find whether these two words have any similarity to our known inputs or not. Assume for this particular question the known answers are: "employee", "employer", "self-employed" and "student". In this

\footnote{LCS is the least common subsumer (shared parent) of two concepts}
case, the highest similarity exists between the words "waiter" and "employee" which is 90% (according to the above equation). Therefore, the application will update the user’s context with "employee" for the attribute “occupation”. As shown in Figure 5.5, at each step in the questionnaire the user can view his/her context to know if the application could understand his/her situation correctly or not.

5.3 Discovered Services

After asking several questions, the user’s context would be refined enough to identify a service or several services which match user’s circumstances. The application receives the name of the service or services which are appropriate for the user. Now, the user can select his preferred service to apply for and the application would redirect the user to the website for that service. The Map operation is also integrated with the application
Figure 5.8: The lattice for the table of customers-attributes with 35 customers and 25 attributes.

to show the closest TD Banks to the user location (Figure 5.7).

5.4 Proposing Relevant Services

At each step of refining the user’s context, the user receives different guiding information such as: recommended services, and promotions, which are highly relevant to the status of the user. The user receives these recommendations based on application of data mining operations on the existing information from the current customers who have similar attributes to this user and have applied for those recommended services.

In Figure 5.4, the plus button (“+”) at the bottom of the screen is intended to provide “recommendation” for the user at every step of the context refinement process. The application uses the “concept lattice analysis” technique that is applied on two tables in the database to identify the proper recommendations for the user. The first table
Chapter 5. Experimentation

Figure 5.9: The lattice for the table of customers-attributes after selecting the targeted concept.

Figure 5.10: The lattice for the table of recommendations-customers.
contains concepts from “(Costumer, Attribute) relation” and the second table contains concepts from “(Recommendation, Customer) relation” that allow us to find the appropriate recommendations for the current user during the interaction with the decision support service.

As an example, consider the lattice that we built for a table of (Customer, Attribute) relation with 35 customers and 28 attributes in Figure 5.9. The concept lattice analysis of this table creates a total of 504 concepts \{\{customers\}, \{attributes\}\} which are stored in a table to be used in the application. As an example, for a “male” user with the age below 25 years we find the concept \{\{male, ”age < 25”\}, \{C_{14}, C_{16}, C_{18}, C_{26}, C_{32}, C_{34}\}\} in the lattice of Figure 5.9, where the customers with ID numbers: 14, 16, 18, 26, 32 and 34 all share the same attributes (male and age<25).

This concept can be seen in the lattice of Figure 5.9 and it shows that these customers (14, 16, 18, 26, 32, 34) have the same attributes (male and age<25). We also use another table which includes all 35 customers and indicates which services and promotions they have used. From this table we create another lattice (Figure 5.10) which creates a table of concepts to be used in the application. With regard to this table of concepts we can find the best concept which contains maximal association of the customers we have found in the previous step. Figure 5.10 shows the best concept we found (indicated as node 14 in the figure) which includes customers 14, 16, 18, 32 and 34 as intent and these customers all use the “tax-free saving account” and “promotion-2” as indicated in the extent of the concept \{\{C_{14}, C_{16}, C_{18}, C_{32}, C_{34}\}, \{tax-free saving account, ”promotion-2”\}\}. Hence, these are the best options to display for the user based on what we know from him/her till now. Figure 5.11 shows a recommendation which application provides for the user.
5.5 Cloud as the server provider

We have used Amazon EC2 to test our server on Amazon public cloud provider. The AWS (Amazon Web Services) Free Tier includes 750 hours of Linux Micro Instances with 32 or 64 bits processor and 615MB memory. We installed Apache web server and MySQL RDBMS (Relational Database Management System) on the Linux virtual machine and deployed our server. Amazon EC2 IaaS pricing is listed in table 5.1.

We created an instance of Ubuntu 12.04 on amazon ec2. Apache web server, MySQL as relational database management system, PHP as server side scripting language and Python for implementing the algorithms of text processing have been installed on this virtual machine. The server is available to use on IP address: http://54.213.64.65/.

Also a comparison of security features and reliability of IaaS providers has been shown in table 5.2.
## Amazon ec2 prices based on instance size

<table>
<thead>
<tr>
<th>Linux Instance</th>
<th>Price/Hour</th>
</tr>
</thead>
<tbody>
<tr>
<td>Small Standard</td>
<td>$0.06</td>
</tr>
<tr>
<td>Medium Standard</td>
<td>$0.12</td>
</tr>
<tr>
<td>Large Standard</td>
<td>$0.24</td>
</tr>
<tr>
<td>Extra Large Standard</td>
<td>$0.48</td>
</tr>
<tr>
<td>Extra Large Second Generation</td>
<td>$0.5</td>
</tr>
<tr>
<td>Double Extra Large Second Generation</td>
<td>$1</td>
</tr>
<tr>
<td>Micro</td>
<td>$0.02</td>
</tr>
<tr>
<td>Extra large High Memory</td>
<td>$0.41</td>
</tr>
<tr>
<td>Double Extra Large High Memory</td>
<td>$0.82</td>
</tr>
<tr>
<td>Quadruple Extra Large High Memory</td>
<td>$1.64</td>
</tr>
<tr>
<td>Medium High CPU</td>
<td>$0.145</td>
</tr>
<tr>
<td>Extra Large High CPU</td>
<td>$0.58</td>
</tr>
<tr>
<td>Quadruple Extra Large Cluster Compute</td>
<td>$1.3</td>
</tr>
<tr>
<td>Eight Extra Large Cluster Compute</td>
<td>$2.4</td>
</tr>
<tr>
<td>Eight Extra High Memory</td>
<td>$3.5</td>
</tr>
<tr>
<td>Quadruple Extra Large Cluster GPU</td>
<td>$2.1</td>
</tr>
<tr>
<td>Quadruple Extra Large High I/O</td>
<td>$3.1</td>
</tr>
<tr>
<td>Eight Extra Large High Storage</td>
<td>$4.6</td>
</tr>
</tbody>
</table>

Table 5.1: Amazon ec2 prices based on instance size.
Table 5.2: Comparison of several famous IaaS providers.

<table>
<thead>
<tr>
<th>Providers</th>
<th>Certification</th>
<th>Protection</th>
<th>Service Age</th>
<th>Support</th>
</tr>
</thead>
<tbody>
<tr>
<td>Amazon EC2</td>
<td>Yes</td>
<td>Medium</td>
<td>5+ Years</td>
<td>Poor</td>
</tr>
<tr>
<td>Rackspace</td>
<td>Yes</td>
<td>Poor</td>
<td>5+ Years</td>
<td>Extensive</td>
</tr>
<tr>
<td>Microsoft</td>
<td>Yes</td>
<td>Poor</td>
<td>&lt;1 Year</td>
<td>Average</td>
</tr>
<tr>
<td>IBM</td>
<td>Yes</td>
<td>Medium</td>
<td>1-2 Years</td>
<td>Poor</td>
</tr>
<tr>
<td>Google</td>
<td>Yes</td>
<td>Poor</td>
<td>&lt;1 Year</td>
<td>Poor</td>
</tr>
</tbody>
</table>

In this Table, by "certification" we meant if the vendor has compliance and security related certification. Also "protection" implies the possibility of protecting servers with firewalls and other means. Service age is the time that the service has ben around and support indicates if the provider offers free or in the base price support for the service like on-line forums.
Chapter 6

Restrictions of the Approach

To preserve data confidentiality of the organizations, each organization should develop their own database of decision trees and services within their private servers (or private cloud). While this is a daunting task for the organizations, providing sample design, code structure and accurate documentation will assist them significantly in this task.

In this chapter, we discuss the difficulties we encountered during the design and implementation of the proposed approach. In Section 6.1 we discuss the cloud complexities. Client-side service representative is explained in Section 6.2. In Section 6.3 the application of concept lattice analysis in our research will be discussed. Finally in Section 6.4 we explain the challenges we had in using Python and connecting Xcode to the Python script with regard to the text processing part of the application.

6.1 Cloud Computing

In the design phase of our approach, we first decided to use Platform-as-a-Service (PaaS) technology which in some cases (for example Google App Engine) is free to charge and provides a plug-in for a programming language to write the server-side of the application.
However, the problem we encountered was that there is no adequate mobile back-end support by such cloud providers for sending and receiving data to a mobile device. Although Google provides a mobile back end to use for iOS and Android devices, such solutions are not yet finalized and stable.

Because of the following reasons we moved from PaaS to IaaS (Infrastructure-as-a-Service) to test our server-side on the cloud which is very simple to achieve since we only needed to install the server on a virtual machine and upload it to the cloud. However, this approach is expensive when the server is working on the cloud.

6.2 Client-Side Service Computation

The advantage of this approach is that it allows us to use a “client-side service representative” to execute the application on the mobile device, which enables the application to perform the service for the user locally. Service representative is the code which processes the service at the client side (rather than server side) using a task message that the service representative receives from the server. This preserves the user’s data privacy and reduces the communication traffic. The problem of this technique in the context of our approach was that we needed to design the task message (containing provider’s business details) which should be sent from the server to client application in order to perform the service for the client. Due to the confidentiality aspects of the provider’s business secrets, organizations require that the users apply to their services through their own portals so that the organizations do not need to provide detailed information which is needed to design the task message.
6.3 Concept Lattice and Large Databases

In Section 5.3, we have used concept lattice analysis on a database of 35 customers (as objects) and 28 attributes to extract “concepts” to reduce time complexity and efficiently find relevant data. The concept lattice tool generated more than 500 concepts and as it is shown in the theory of concept lattice, the number of produced concepts increases exponentially when the number of objects increases. As a result of this study we observed that while the concept lattice technique reduces time complexity and increases efficiency of the searching, it obviously creates huge data and needs large data space for analyzing the produced concepts.

6.4 Use Of Python

During the development phase we noticed that there is no practical solution to directly connect iPhone applications (or generally Xcode) with Python scripts (text processing part of the project). After lengthy investigations, we could solved the problem by using PHP to work as a bridge to connect them. Hence we wrote PHP programs to wrap and execute the Python scripts and send the result to iPhone application.

Moreover, in the Python script we need to search several times through the WordNet database; each time we run the python script it takes about 5 seconds (with a MacBook Pro core i5 and 4 GB memory). However, the execution time will drastically reduces if the database and Python scripts are run on a cloud platform with much more processing power.
Chapter 7

Discussion And Conclusion

In this thesis, we presented a novel service selection technique for large business organizations that interactively guides the user who is not familiar with complicated services of public organizations to select services that optimally match with their intended tasks. We employed techniques from decision support systems (DSS) and semantic analysis to develop a new concept which has been implemented as an iPhone application. Moreover, we have used concept lattice analysis from data mining domain to match the user’s attributes with those of previous customers of the organization to provide to the new users the best possible recommendations of services at each point in the decision making path. The application is designed to customize itself to match with the type of questions it receives from the server to make it simpler for the user to respond. It means if databases of decision trees change due to new business rules and regulations, there is no need to modify the client application.
7.1 Future Work

As the next step, we plan to continue working on enhancing the proposed decision support service to make it more sophisticated and support more features. Also by exporting the server side of the application to Platform-as-a-service, we will enhance the usability of the application in real world. Also in the text processing part we will use better algorithms to better understand the user responses and to accelerating the process.
Bibliography


Appendix A

Taxonomy

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DSS</td>
<td>Decision Support Systems</td>
</tr>
<tr>
<td>CaaS</td>
<td>Consultant as a Service</td>
</tr>
<tr>
<td>CDSS</td>
<td>Clinical Decision Support System</td>
</tr>
<tr>
<td>ODSS</td>
<td>Organizational Decision Support System</td>
</tr>
<tr>
<td>GDSS</td>
<td>Group Decision Support System</td>
</tr>
<tr>
<td>PHP</td>
<td>Hypertext Preprocessor</td>
</tr>
<tr>
<td>NLTK</td>
<td>Natural Language Tool Kit</td>
</tr>
<tr>
<td>WN</td>
<td>WordNet</td>
</tr>
<tr>
<td>IaaS</td>
<td>Infrastructure as a Service</td>
</tr>
<tr>
<td>SaaS</td>
<td>Software as a Service</td>
</tr>
<tr>
<td>PaaS</td>
<td>Platform as a Service</td>
</tr>
<tr>
<td>ARPANET</td>
<td>Advanced Research Projects Agency Network</td>
</tr>
<tr>
<td>HuaaS</td>
<td>Human as a Service</td>
</tr>
</tbody>
</table>
Appendix A. Taxonomy

LCH Leacock and Chodorow, 1998
WUP Wu and Palmer, 1994
IDE Integrated Development Environment
SQL Structured Query Language
RDBMS Relational Database Management Systems
ISO International Organization for Standards
UI User Interface
Appendix B

Sample Code Snippets

In this chapter we explain functions and algorithms we have used to design our application. In section B.1, we present the use of Objective-C to design our client side of the application and after that in section B.2 we explains PHP scripts which is used in the server side and finally Python script is explained in section B.3.

B.1 Objective-C

JSON or JavaScript Object Notation is an open standard designed for data exchange which is derived form JavaScript. It is language independet and compilers and parsers are available for many programming languages including Objective-C.

We reguarly send and receive data from our PHP server scripts in the application. JSON is used as the communication protocol in this thesis. Code below is the function we wrote in Objective C to take care of our communications with the server. This function gets the unknown answer and the known inputs and return the result which is a string received from the server script.
In the code you can see the setup for the HTTP POST message which sends a string to a url and returns the data which we parse as a JSON object.

When a service retrieved from the server attached with the name and the service info, the url to that web service is gotten by the module. Moreover, the url will be sent to a web view to direct the user to that specific website where user can apply for that service.
The code doing so is displayed below.

```
- (IBAction)web:(id)sender {
    del.string = webstr;
    UIViewController *controller = [self.storyboard
        instantiateViewControllerWithIdentifier:@"tabbar"];
    [self.navigationController pushViewController:controller animated:YES
        ];
}
```

del.string is the global variable we have used to send the url to the other class we have designed to show the website. When the web view pops up, the function attached to it will send the request to the web view to display the url. The code performing this feature is provided below.

```
- (void)viewDidLoad{
    [super viewDidLoad];
    NSString *str = del.string;
    NSURL *url = [NSURL URLWithString:str];
    NSURLRequest *request = [NSURLRequest requestWithURL:url];
    [webview loadRequest:request];
}
```

**B.2 PHP**

In the server there are two main PHP script which would be accessed from the application. The first PHP script is used as a wrapper to get the data from the application and execute the Python script with those arguments. This script is provided below.

```
<?php
    $known = $_GET["known"];
    $sentence = $_GET["sentence"];?
    $tmp[] = exec("python similarity.py ".$known." ".$sentence." ");
    echo json_encode($tmp);?
```
As you can see these few lines of codes receives the known inputs and the user answer and execute the similarity.py script and send back the result of this script in JSON format which is understandable for Objective-C.

The second PHP script is used to access the TD database, which contains the decision trees for the TD organization, with the defined context and retrieve the appropriate element in this database. Code below is used for performing such a function.

```php
<?php
$status = $_GET["status"];
$occupation = $_GET["occupation"]; $typeofservice = $_GET["typeofservice"];
$age = $_GET["age"]; $amountofmoney = $_GET["amountofmoney"]; $useofmoney = $_GET["useofmoney"]; $cre dithistory = $_GET["credithistory"]; $degree = $_GET["degree"];
$username = "root";
$database = "consultantdb";
$mysqli = new mysqli(localhost, $username, "", $database);
$query = 'SELECT * FROM TDdb';
if ($status) {
    $query = $query . " WHERE status = " . $status . " ";
}
if ($occupation) {
    $query = $query . " AND occupation = " . $occupation . " ";
}
if ($typeofservice) {
    $query = $query . " AND typeofservice = " . $typeofservice . " ";
}
```
Appendix B. Sample Code Snippets

```php
if ($age) {
    if (((int) $age) > 18) {
        $query = $query . " AND age = 19";
    }
    if (((int) $age) < 19) {
        $query = $query . " AND age = 18";
    }
}

if ($amountofmoney) {
    if (((int) $amountofmoney) > 4999) {
        $query = $query . " AND amountofmoney = 5000";
    }
    if (((int) $amountofmoney) < 5000) {
        $query = $query . " AND amountofmoney = 4000";
    }
}

if ($useofmoney) {
    $query = $query . " AND useofmoney = '" . $useofmoney . "'";
}

if ($credithistory) {
    $query = $query . " AND credithistory = '" . $credithistory . "'";
}

if ($degree) {
    $query = $query . " AND degree = '" . $degree . "'";
}

if ($result = $mysqli->query($query)) {
    while ($r = $result->fetch_object()) {
        $rows[] = $r;
    }
    $result->close();
```
The above script declares an object from the specific database we want to have access to and then it and runs a query with the context we received from the application and sends back the results of this query.

In our research, we faced a problem in which, we couldn’t find any lattice software or tool to produce concepts for us as a relational database. Most of these tools produce the xml database of concepts in a lattice generated from an object-attribute table such as Lattice Miner. To solve such a problem we decided to write a php script to convert the xml database, generated from the lattice, to MySQL table which is a relational database management system.

```php
<?php

$numOfObj = 19;
$numOfAtt = 17;
$numOfCon = 99;
$nameOfFile = 'usAtt.xml';
$nameOfTable = 'userAttribute';
$username = "root";
$database = "consultantdb";
mysql_connect(localhost, $username);
@mysql_select_db($database) or die("Unable to find database");
$query = "CREATE TABLE $nameOfTable (objects CHAR(80), attributes CHAR(80),
                           numOfObj INT, numOfAtt INT)";
mysql_query($query) or die (mysql_error("error"));
// finds numeric characters in a string
function get_numerics ($str) {
```
Appendix B. Sample Code Snippets

```
preg_match_all('/\d+/', $str, $matches);
return $matches[0];

// Loads the xml file
$xmlstr = file_get_contents($nameOfFile);
$LAT = new SimpleXMLElement($xmlstr);

// gets the ids of objects
for($i=0; $i<$numOfObj; $i++){
  $objArray[$i] = $LAT->OBJS->OBJ[$i];
}

// gets the ids of attributes
for($i=0; $i<$numOfAtt; $i++){
  $attArray[$i] = $LAT->ATTS->ATT[$i];
}

for($i=1; $i<($numOfCon-1); $i++){
  // gets the object of node
  $nodeStr = $LAT->NODS->NOD[$i]->EXT->asXML();
  $nodeObj = get_numerics($nodeStr);
  print "\n";
  foreach($nodeObj as $key=>$value){
    $nodeObj[$key] = $objArray[$value];
  }

  // gets the attributes of node
  $nodeStr = $LAT->NODS->NOD[$i]->INT->asXML();
  $nodeAtt = get_numerics($nodeStr);
  print "\n";
  foreach($nodeAtt as $key=>$value){
    $nodeAtt[$key] = $attArray[$value];
  }
  $Obj = implode(",", $nodeObj);
  $Att = implode(",", $nodeAtt);
  $numOfObj = count($nodeObj);
```
The algorithm we wrote, gets the number of objects and attributes and the location of the xml file along with the address of the MySQL database which in default is the localhost. Then it generates a table with a desired name in that database containing all the concepts from the lattice except the first and last concepts. Because first concept contains all the objects without any attribute and the last concept contains all the attributes without any object. Hence these two concepts are not useful in our data mining since they do not contain any useful information.

B.3 Python

Python is used in this application to run the unknown user’s answer against the WordNet database to get a meaning of it which is understandable for the consultant of the organization. In the Python script we first import the WordNet and packages we need from NLTK.

```python
from nltk.corpus import wordnet as wn
import nltk.tag

tokens = nltk.word_tokenize(sentence)
```
Appendix B. Sample Code Snippets

```python
2 tagged = nltk.pos_tag(tokens)

For example when we tag the sentence "I am a clerk in BMW", we will get:
('I', 'PRP'), ('am', 'VBP'), ('a', 'DT'), ('clerk', 'NN'), ('in', 'IN'), ('BMW', 'NN')]

Moreover we keep nouns, verbs and adjectives and omit all the other part of the sentence.

```python
1 for x in tagged:
2     if i > 1:
3         if 'NN' in x:
4             array.insert(1, re.sub(r'\bNN\s?', '', nltk.tag.tuple2str(x)))
5         elif 'NNS' in x:
6             array.insert(1, re.sub(r'\bNNS\s?', '', nltk.tag.tuple2str(x)))
7         elif 'VBG' in x:
8             array.insert(1, re.sub(r'\bVBG\s?', '', nltk.tag.tuple2str(x)))
9         elif 'VBN' in x:
10        array.insert(1, re.sub(r'\bVBN\s?', '', nltk.tag.tuple2str(x)))
11        elif 'VBP' in x:
12        array.insert(1, re.sub(r'\bVBP\s?', '', nltk.tag.tuple2str(x)))
13        elif 'JJ' in x:
14        array.insert(1, re.sub(r'\bJJ\s?', '', nltk.tag.tuple2str(x)))
15     i = i + 1
```

After tokenizing and tagging, next we will use wup.similarity on the WordNet database to find the best match between the tagged words and the known inputs.

```python
1 for y in array:
```
In this script we try to find the synset for the each tagged word. If it exist we continue with it but we will try to speel check the word if a synset did not exist for the word. and then we try to match all the tagged words with the known words and we keep the best similarity found. If a similarity better than a fixed percentage (80%) wasn’t found the script would send back a NoAnswerFound message. At the end we send back the results of this script to the application to proceed.